University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

College of Arts and Sciences – Laboratory Inventory Monitoring System (CAS-LIMS)

A Special Problem in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science

By:

Miguelino C. San Miguel

2008-03367

April 2014

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "College of Arts and Sciences – Laboratory Inventory Monitoring System" prepared and submitted by Miguelino San Miguel in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____

Avegail D. Carpio, M.Sc.

Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. *(candidate)*	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Aldrich Colin K. Co, M.Sc. *(candidate)*	_____	_____
4. Ma. Sheila A. Magboo, M.Sc.	_____	_____
5. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
6. Geoffrey A. Solano, M.Sc.	_____	_____
7. Bernie B. Terrado, M.Sc. *(candidate)*	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

_____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

_____

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

The College of Arts and Sciences needs a means to monitor laboratory chemicals and laboratory equipments in its various units especially in laboratories. The College of Arts and Science – Laboratory Inventory Monitoring System provides an efficient use of a monitoring system to prevent human errors in recording the inventory of the chemicals and laboratory equipments. Various Users have been designated to help in the processing of data and accessing information. The significance attained are monitoring the accountability of students, establishing an alert system for replenishment order of chemicals and laboratory equipments, and establishing an alert system for orders of condemnation and repair of machine in laboratories.

Keywords: monitoring system, laboratory chemicals and equipments, alert system, accountability transactions

Contents

List of Figures

Figure 28 – Inventory of Labwares

List of Tables

I. Introduction

A. Background of the study

University of the Philippines Manila (UPM) is known for its quality education especially in Medicine and Sciences. UPM offers a wide variety of degree programs in the field of Medicine and Sciences such as INTARMED, Doctor of Dental Medicine, BS Public Health, BS Pharmacy, BS Nursing, BS Biology, BS Applied Physics, BS Biochemistry, and BS Computer Science whose curriculums include foundation courses in Natural and Physical Sciences. College of Arts and Sciences (CAS) is one of the important pillars of the university that absolutely provides the foundation of knowledge for the different colleges. Laboratory is a prerequisite environment on learning the world of these subjects. It is important for the university to provide the necessary tools and actual procedures in simulating the real processes so as to go beyond book learning. Some equipment/machines which CAS really utilized in its Lab Classes are computers and laboratory modules.

A laboratory attendant is responsible in monitoring the incoming of the supplies, outgoing of the supplies and use of these items. It include tasks such as maintenance of machines/equipment, and monitoring of the quantity of disposable items so as to prevent shortage. It is possible that there is more than one laboratory class held each day. Lab attendants and coordinators must employ an efficient monitoring method of gadgets, instruments and tools present in a laboratory room such as laboratory vessels and instruments. The current method of items inventory consists of tables of information maintained in one computer and then updated regularly by the laboratory administrator. Microsoft Word is the application being used to store this information.

B. Statement of the Problem

The manual method of keeping files using Microsoft Word lacks data consistency, security and reliability. File storage such as Microsoft Word is not sufficient since this kind of storage is not centralized, accessible only to one computer and prone to inconsistencies. In accessing files, this method is also tedious and prone to human error. Also access of information is only accessible to one computer, so laboratory admins would always need to access the same computer to get information. Laboratory faculty coordinators always need to reach the laboratory admin to get information about the availability of items, gadgets, tools and instruments in the inventory.

C. Objectives

Develop a centralized and secured inventory system that would provide an efficient and reliable way of automating different processes of monitoring of laboratory items.

- Allow the Lab Attendant to:
    - update "consumable" items in each laboratory discipline
    - indicate machine as out of order or due for repair
    - set a critical level of quantity deemed as need to request for replenishment
    - declare an equipment as condemned
    - generate report at the end of semester regarding students with lab deficiencies
- Allow the Lab Faculty Coordinator to:
    - view alerts regarding need to order replenishment
    - request for supplies for each laboratory discipline
    - request for equipment to be condemned
    - request for repairs of equipment/ machine
- Allow the Supply Officer to:
    - approve/ disapprove lab order request
    - approve request to condemn equipment
    - approve request for repair of equipment by service provider
- Allow the System Admin to:
    - manage user accounts such as create, delete, view, and update access rights.
    - update accounts including adding/removing associated roles to a user.
    - create backups of the database.

D. Significance of the Study

Through the system monitoring simplified, we will expect speedy responses from request of repair; we will prevent scarcity of laboratory items used in classes; and we will easily monitor the consumption of laboratory consumables.

The system will improve data consistency and reliability through web and database technology. The system would be able to prevent the users from committing errors that are common to file storage. It will also open the chance of not just improving the system but also the process itself in managing laboratory equipment if expansion of responsibility would happen. The use of these software technologies is a future proof response to these needs especially when evolving requirements and methodology is common. If required later, making the data centralized as well to CAS or even to the university level would make data delivery to various concerned departments faster if needed.

E. Scope and Limitations

- Only system admin could manage user accounts and its access rights.
- There are three various kinds of items: machines, lab equipment and chemicals.
- The system is limited only on creating/updating items of laboratory items. Any unusable item would be put into a condemned/unusable status, but not deleted.
- Local network is the initial target environment of this system.
- This system only includes selected laboratory of CAS: math/CS, biology, chemistry and physics. It does not support laboratories used in the Department of Arts and Communication (DAC), Department of Behavioral Science (DBS), and Department of Social Science.

F. Assumptions

This paper assumes that:

- Internet/Ethernet connection is provided.
- the lab faculty coordinator acts as a Department Head in requesting for replenishment, requesting for condemnation and requesting for repair.
- the memorandum receipt which is found on each laboratory items makes the lab attendant accountable since the lab attendant will record the memorandum receipt on the system.
- the accounts are on a semestral basis.

II. Review of Related Literature

Inventory management is a prime task for an organization to achieve its goals of maintaining appropriate level of inventory and minimizing waste. One way of managing inventory is to have a web-based system in place that can instantly track and update the information about the products, tools or equipment. A report by Yankee Group (2005) reveals that product information management (PIM) improves inventory management by 25%. The study highlights the fact that the retailers and suppliers using web-based product information management system realize an average of 25% improvement in operations related to inventory control. As an organization grows, it is required to deal with a lot of paper-based records for each transaction, necessitating a lot of documentation in hardcopies. As an alternative, a web-based inventory system's records can be digitally archived, thus reducing filing activity at the end of each term's end. Chaffee, A. (2009) states that the web-based application has been found to be essential in recent years as more organization started to realize the importance of applying new technology to assist current inventory practices. In the last 10 years, a barcode technology has been replaced with an RFID (Radio Frequency Identification) technology, and then evolved rapidly to being able to track any item by using GPS-based device. The importance of implementing a web based inventory system is thus becoming vital as most of the time the information are accessible instantly, thereby making the details of the usage of the equipment available, and improving the movement and anticipation of their demand as well as the productivity of the system as a whole. The implementation of technology-mediated learning can help institutions develop the skills to cope with their operational environments. [1]

Selecting web 2.0 as the platform for implementing the system as web application is considerably future proof. In Web 2.0, the programming models and concepts differ from those used in the enterprise, though they are services-based and underpinned with the concept of messaging passing; they use Representational State Transfer (REST) protocols, and focus on simplicity and ease of use.

Web 2.0 programming is based on separation of concerns using a loosely coupled, message- passing-based model on top of an Internet-based, standard set of communications protocols (HTTP) which is often called RESTful programming. It entails the acts of syndication and composition where services are delivered exclusively to knowledge of their use. This is different from a usual tightly coupled, transactional and object-oriented system. It has a different set of benefits, such as flexibility and speed of implementation, and challenges, such as integrity and management" (O'Really T, 2007). The languages used in Web 2.0 like Perl, Python and Ruby frameworks are simple and dynamic and thus providing a low blockade to entry, reuse and high productivity.[2] In comparison with the languages enumerated by the former citation, it is safe to say that Java is much more scalable, mature, secured, and popular as well. Java has been always part of the top 5

popular languages in Tiobe index. Many industries even in Philippines use Java all the more in developing systems on top of web 2.0. It is cross platform and lots of frameworks has been created to ease the web development such as Spring, Struts, Wicket, Play, etc. One proof of its scalability is when tweeter experienced server traffic bottleneck because of millions of simultaneous user access, it migrated its implementation from Ruby framework to a Java derivative/framework which is Scala. There are also specialized IDE such as Eclipse, IDEA and Netbeans that provide out of the box features that could ease and speed up the application development. Therefore choosing the web platform and Java as our backend language would allow the system to be future proof when it comes to evolving requirements and scalable in the context of increasing system usage.

One of the main objectives of inventory management is to minimize the inventory carrying cost. Therefore, it is very important to determine the optimal stock and optimal time of replenishment of inventory to meet the future demand. In classical inventory models the demand rate is assumed to be a constant. But demand for physical goods may be time-dependent, price dependent and stock dependent. [3] Thus, a common problem that arises in the management of an inventory is demand uncertainty. It is difficult to predict the future demand of a new seasonal product because of insufficient historical data. [4] When it comes to inventory system, we should only consider not only the system technical implementation itself but also the business process which it is rooted. The biggest profit in the supply chain of this process is to meet the customers demand quickly with the lowest cost. Inevitably, by this way, it will have to accumulate a large number of stocks to ensure that the final consumer can get the products which they want to buy. However, this kind of management is also apt to accumulate too many stocks in the points of sales which lead the entire supply chain expose to high risk, from the rising of inventory costs to the dull sales or products sold at a loss. Therefore, another method of inventory management is to prepare fewer stocks to avoid the cost loss of inventory owing to market changes.[5] But the reaction speed to demand could be affected. In Musara Mazanai's research, his paper presents the findings of the study that was conducted to investigate the impact of application of Just-In-Time (JIT) inventory management system in the manufacturing sector SMEs. The study revealed that the majority of SMEs in the manufacturing sector were not applying the JIT inventory management principles. Furthermore, statistically significant positive correlations between the application of JIT inventory management principles and cost efficiency, quality and flexibility were found. It is therefore deduced that manufacturing sector SMEs can benefit significantly in terms of improved quality of products, increased operational cost cuts and increased flexibility by applying the JIT inventory management principles. As an illustration, Procter & Gamble drove out non value-adding supply chain costs to save the company over $200 million by using an optimization model with an interactive approach [6].

One of the approaches which have long been proven effective in the manufacturing sector in cutting costs, improving quality, productivity, efficiency and decreasing waste is the just in time (JIT) management approach. JIT is a management approach which originated in Japan in the 1950s. According to Chase et al. (2006), waste in Japan, as defined by Toyota's Fujio Cho, is 'anything other than the minimum amount of equipment, materials, parts, and workers (working time) which are absolutely essential to production'. JIT has the effect of releasing the much needed capital and lessen the burden of SMEs of the need to keep excessive inventories.[7] In College of Arts and Sciences – Laboratory Inventory Monitoring System(CAS-LIMS), a requirement has been employed in monitoring the minimum quantity for each item of every involved laboratory discipline. But training for involved employees is also very important. Since the system is on its starting stage, we could still consider that it is not that smart enough to understand statistical usage reports and demand rate and be able to adjust its replenishment automation according to these parameters. It is very important though that the system users should be able to adjust according to the said factors that could affect dynamically the frequency of students order and the deterioration of certain items. They should also provide feedbacks to the administration to assist in improving the system thus will open the chance for further enhancements and optimizations.

Another factor to consider in this kind of inventory is deterioration of items. Deterioration is defined as decay, damage, spoilage, evaporation, obsolescence, pilferage, and loss of entity or loss of marginal value of a commodity that results in decreasing usefulness from the original one.[8] The system has addressed this need by allowing laboratory attendants to set the minimum quantity level for certain items to trigger alerts for the need to send a request for replenishment order. The request is officially send by the laboratory faculty coordinator based on these alerts. The supply officer would be responsible to approve or reject the request. Another type of this request is the maintenance of repairable machines/equipment. The same request flow is used. The advantage of this requirement, it could open up more opportunity for the university to optimize and cut cost. In Hui-Ming Wee research, he stated that previous researchers have developed ways of managing forward-oriented supply chains, and gave insight to solve single-stage inventory systems. In his study, they analysed an inventory system with traditional forward-oriented material flow as well as a reverse material flow supply chain. In the reverse material flow, the used products are returned, remanufactured and shipped to the retailer for resale.[9] Inventory holding and warehousing continue to play an important role in modern supply chains. A survey of logistics costs in Europe identified the cost of inventory as being 13 per cent of total logistics costs, whilst warehousing accounted for a further 24 per cent However, determination to control purchase and inventory cost in academic institutions is not as strong as it is in production enterprise. They in many cases have not been motivated to control purchasing costs in the same way that competitive industry has. Academic institutions focus more on keeping the consumer more satisfied and

comfortable so that their teaching and research activities do not suffer. In this sense, quality of material and timeliness is kept in mind. Supply chain management offers great potential for organizations to reduce costs and improve customer service performance However, it should not be forgotten that money saved by cutting inventory cost could be used for academic development of the organization. Therefore, while quality material should be provided in time to facilitate the teaching and research activities, minimization of inventory cost must be ensured.[10]

III. Theoretical Framework

A. College of Arts and Sciences (CAS)

This college provides the common subjects enrolled by students from different courses. This is the primary college that will use the system. The manual system of monitoring the machines and other laboratory resources comprise of the following methods: recording in tables the laboratory chemicals which will include the name, molecular formula, description, acquired date, expiry date, threshold, amount and note in columns, and recording in tables the labwares which will include the type, size, description, the quantity that is in good condition, the quantity that is in broken condition, the quantity that is in stained condition and the total quantity in columns. Both tables have also an indication of the current semester and the current academic year.

B. Database Management System

All businesses have to keep this type of data and much more; and just as importantly, they must have those data available to decision makers when they need them. Information is the result of processing raw data to reveal its meaning. Data processing can be as simple as organizing data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modeling. To reveal meaning, information requires context. It can be argued that the ultimate purpose of all business information systems is to help businesses use information as an organizational resource. At the heart of all of these systems are the collection, storage, aggregation, manipulation, dissemination, and management of data. [1]

The database is now such an integral part of our day-to-day life that often we are not aware we are using one. We consider a database to be a collection of related data and the Database Management System (DBMS) to be the software that manages and controls access to the database. A database application is simply a program that interacts with the database at some point in its execution. We also use the more inclusive term database system to be a collection of application programs that interact with the database along with the DBMS and database itself. [2]

C. MySQL

MySQL is the most popular open source database system available. Although it's free, it's still very dependable and fast, and is being employed increasingly in areas that used to be the province of Oracle or MS SQL Server. MySQL is an open source, multithreaded, relational database management system created by Michael "Monty" Widenius in 1995. In 2000, MySQL was released under a dual-license model that permitted the public to use it for free under the GNU General Public License (GPL); this caused its popularity to soar.

The company that owns and develops MySQL is MySQL AB (the AB stands for aktiebolag, the Swedish term for stock company), which is now a subsidiary of Sun Microsystems. Currently, MySQL AB estimates that there are more than 6 million installations of MySQL worldwide, and reports an average of 50,000 downloads a day of MySQL installation software from its site and from mirror sites. The success of MySQL as a leading database is due not only to its price—after all, other cost-free and open source databases are available—but also its reliability, performance, and features. [3]

D. Java (Backend)

The Java programming language, developed at Sun Microsystems under the guidance of Net luminaries James Gosling and Bill Joy, is designed to be a machine-independent programming language that is both safe enough to traverse networks and powerful enough to replace native executable code. Of even more importance in the past few years, Java has become the premier platform for web-based applications and web services. These applications use technologies including the Java Servlet API, Enterprise JavaBeans, and many popular open source and commercial Java application servers and frameworks. Java's portability and speed make it the platform of choice for modern business applications. [4]

E. Spring Framework (MVC)

Spring is the most popular application development framework for enterprise Java. Spring is an environment within which your code can operate, a set of libraries for solving certain types of problems, and a set of libraries for assisting you in interacting with numerous other frameworks [in java]. At its heart, Spring is primarily a framework for enabling the use of IOC (also known as dependency injection). This is not to diminish Spring's other features, but rather to highlight the importance of IOC in addressing the problem of tangled dependencies. This is exactly the problem that Spring IOC solves: it makes the problem of supplying dependencies to classes so wonderfully simple that we can take full advantage of the benefits of decoupling. [5]

F. Hibernate (ORM)

Hibernate Object Relational Mapping (ORM) framework provides a solution to the complex technical problem of persisting objects from your object model into a relational database.[6]

Most significant development projects involve a relational database. The mainstay of most commercial applications is the large-scale storage of ordered information, such as catalogs, customer lists, contract details, published text, and architectural designs. With the advent of the World Wide Web, the demand for databases

has increased. Though they may not know it, the customers of online bookshops and newspapers are using databases. Somewhere in the guts of the application a database is being queried and a response is offered. While the demand for such applications has grown, their creation has not become noticeably simpler. Some standardization has occurred—the most successful being the Enterprise JavaBeans (EJB) standard of Java 2 Enterprise Edition (J2EE), which provides for container and bean-managed persistence of entity bean classes. Unfortunately, this and other persistence models all suffer to one degree or another from the mismatch between the relational model and the object-oriented model. In short, database persistence is difficult. There are solutions for which EJBs are appropriate, some for which some sort of object relational mapping (ORM) like Hibernate is appropriate, and some for which the traditional approach of direct access via the Java Database Connectivity (JDBC) API is appropriate. We think that Hibernate represents a good first choice, as it does not preclude the simultaneous use of these alternative approaches.[7]

G. HTML5, CSS3

HTML5 and CSS3 help lay the groundwork for the next generation of web applications. They let us build sites that are simpler to develop, easier to maintain, and more user-friendly. HTML5 has new elements for defining site structure and embedding content, which means we don't have to resort to extra markup or plug-ins. CSS3 provides advanced selectors, graphical enhancements, and better font support that make our sites more visually appealing without using font image re- placement techniques, complex JavaScript, or graphics tools. Improved accessibility support will improve Ajax applications for people with disabilities, and offline support lets us start building working applications that don't need an Internet connection. [8]

HTML5 offers support for Web Sockets, which give you a persistent connection to a server. Instead of constantly polling a back end for progress updates, your web page can subscribe to a socket, and the back end can push notifications to your users. With the addition of the Web Storage and Web SQL Database APIs, we can build applications in the browser that can persist data entirely on the client's machine. One of the best reasons for you to embrace HTML5 today is that it works in most existing browsers. Right now, even in Internet Explorer 6, you can start using HTML5 and slowly transition your markup. It'll even validate with the W3C's validation service ([9]

H. JavaScript and jQuery

JavaScript is ubiquitous on the World Wide Web. You can use JavaScript both to make your Web pages more interactive, so that they react to a viewer's actions, and to give your Web pages some special effects. JavaScript often gets thrown in with Hypertext mark-up Language (HTML) as one of the recommended languages for beginning Web developers (whether you build Web sites for business or pleasure). Of course,

you can build a Web page by using only HTML, but JavaScript allows you to add additional features that a static page of HTML can't provide without some sort of scripting or programming help.[10]

The jQuery JavaScript Library can enhance your websites regardless of your background. It provides a wide range of features, an easy-to-learn syntax, and robust cross-platform compatibility in a single compact file. What's more, hundreds of plug-ins have been developed to extend jQuery's functionality, making it an essential tool for nearly every client-side scripting occasion.[11]

IV. Design and Implementation

A. System Design

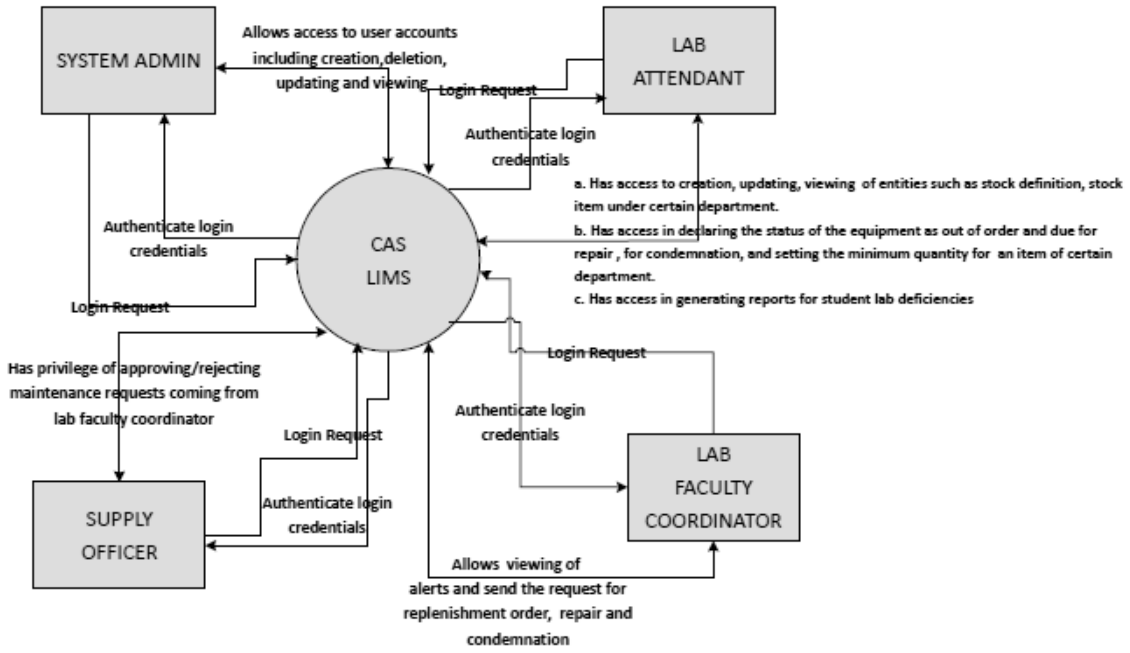

Figure 1 – System Design

The above illustration depicts the top level view of how various users could access functionalities the system provides relatively.

B. Context-Level DFD

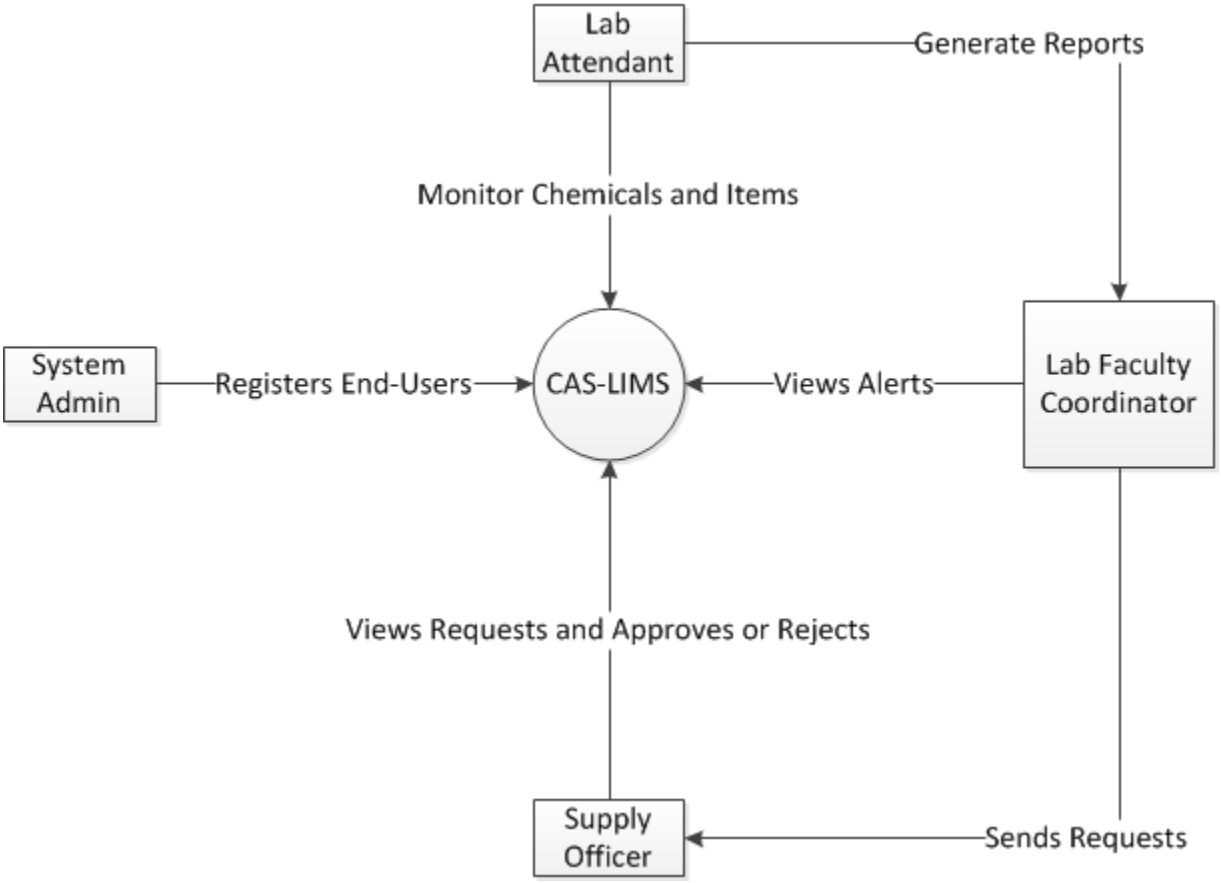

Figure 2 – Context-Level DFD

The illustration above shows various processes accessible to registered users. Only logged in users can access various management actions of LIMS system.

C. DFD Sub-Explosions

Level 1 - 1.0 Manage User Account

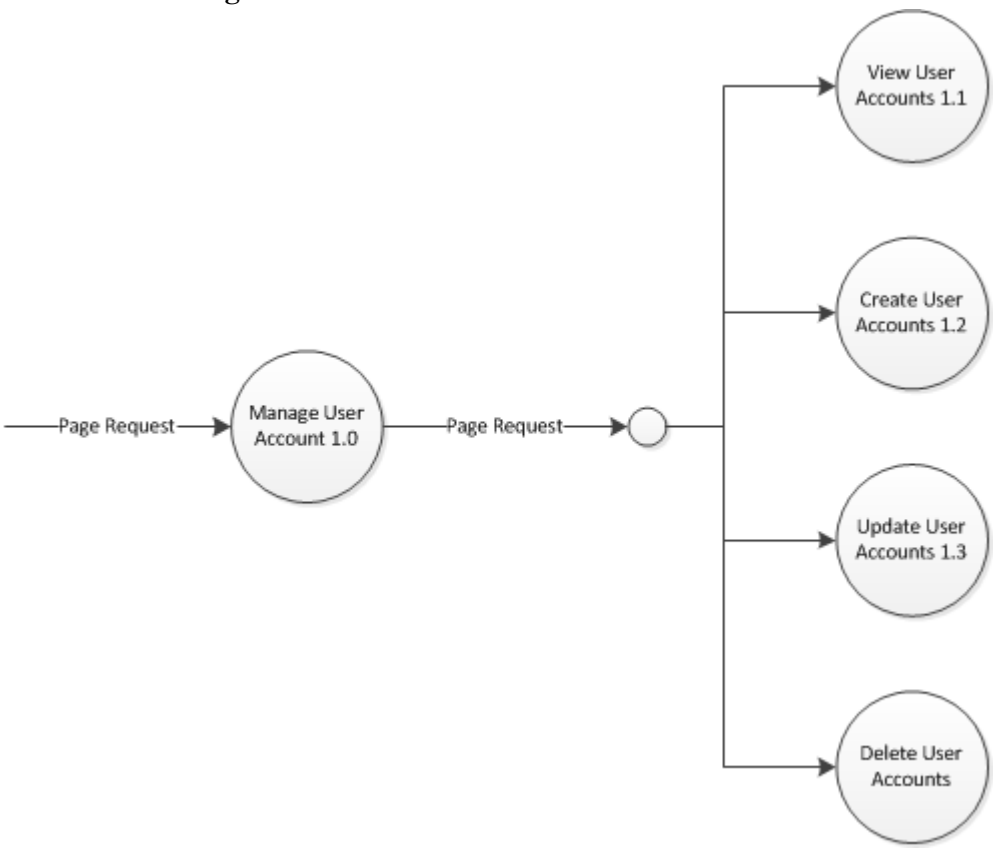

Figure 3 – Manage User Accounts

Management of user accounts is accessible only to system administrator. The admin can view the list of users in a table. Upon viewing, he could execute a delete or update action on a selected user. The admin should have the access on creation of user accounts as well. It is very important to note as well, that updating the accounts as an admin also includes the associating a user to a specific access rights or roles.

These roles play important part in the security access API the system provides and it adheres to the common strategy called Role-based Access Control (RBAC). A user can be associated to a role recognized by the end users such as system admin, lab attendant, etc. But a user can be also associated with a specific access right to a specific operation in a module without affecting the role itself.

Level 1 - 2.0 Manage LIMS Stocks

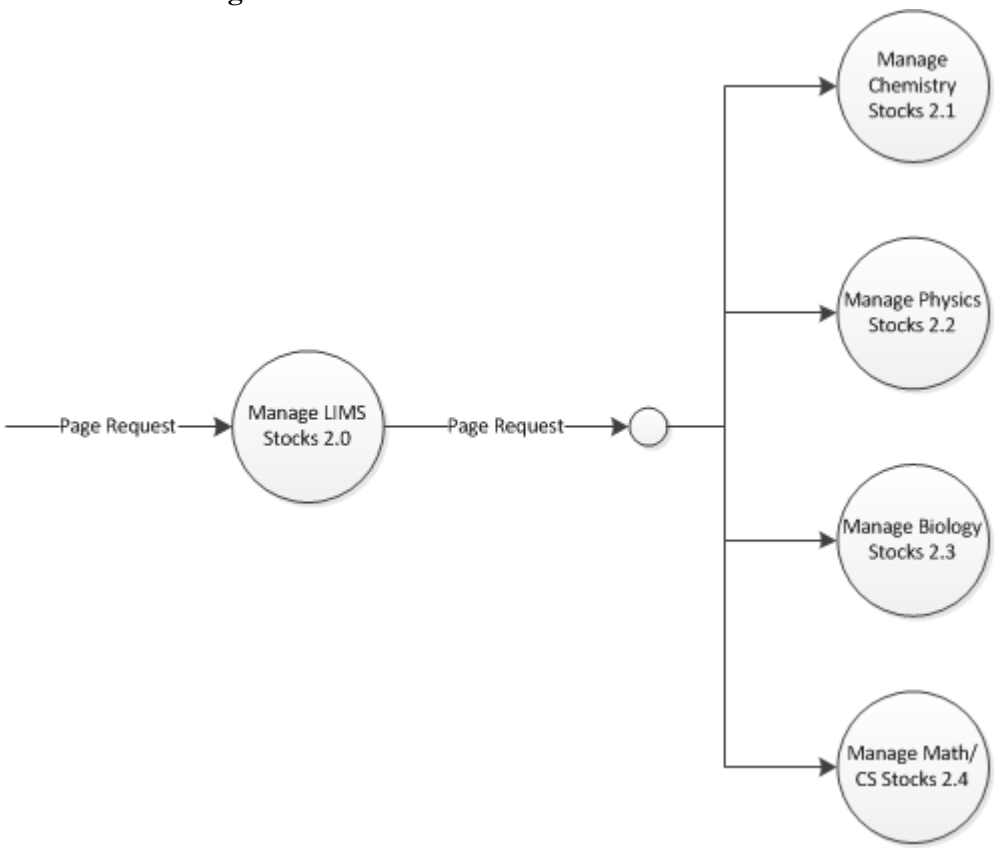

Figure 4 – Manage LIMS Stocks

Figure 4 shows the module of managements of LIMS stocks. It involves 4 main entities including Chemistry stocks, Biology stocks, Physics stocks and Math/CS lab equipment. The chemical and lab equipment entities under physics, biology, math/cs and chemistry departments are identical in nature except that math/cs module doesn't have corresponding chemical stocks. We will see in the proceeding diagrams that processes are basically the same.

Level 2 Sub-Explosion

Level 2 – 2.1 Manage Chemistry Stocks

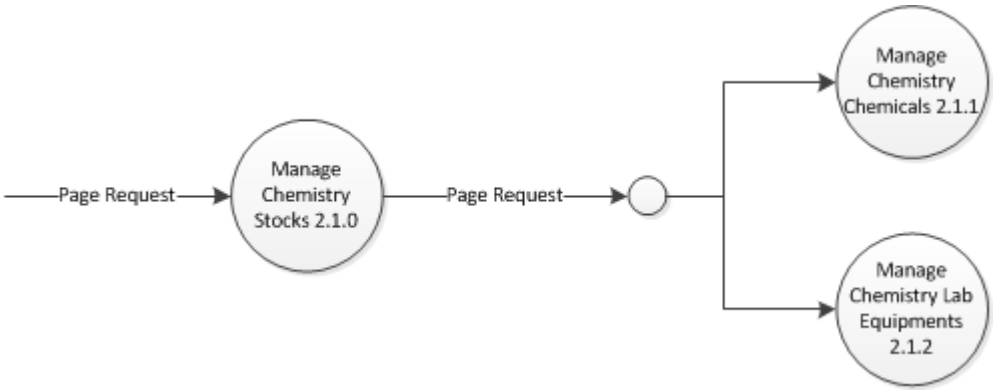

Figure 5 – Manage Chemistry Stocks

The diagram above shows the overview of the two major stock entities the chemistry module has which can be applied to a general module.

Level 3 Sub-Explosion

Level 3 – 2.1.1 Manage Chemicals

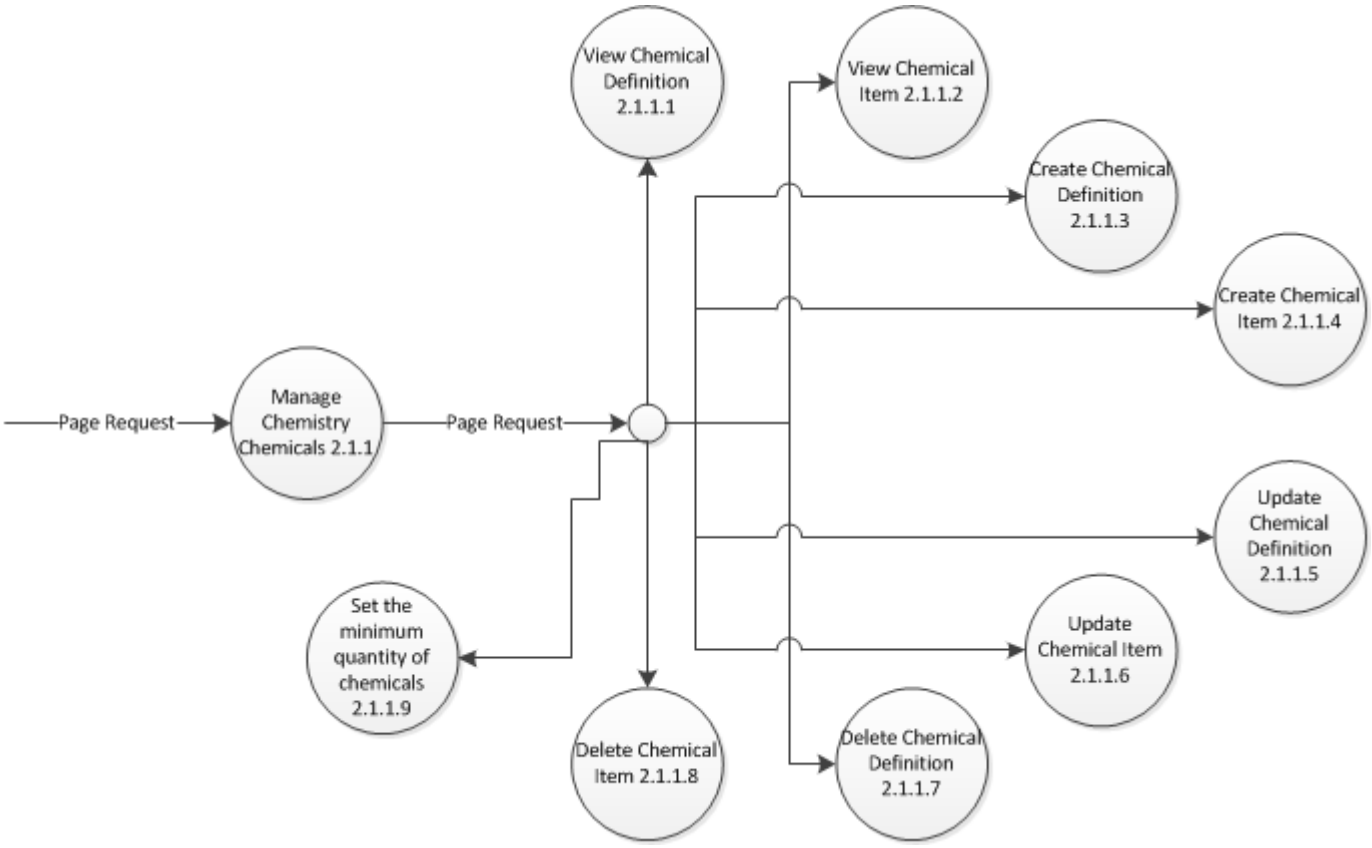

Figure 6 – Manage Chemicals

Note: This diagram applies to Chemistry, Physics and Biology laboratories.

The diagram above illustrates how management of the chemical entity is done. Only lab attendant has complete access to these processes. It involves creation, updating, deletion and viewing of chemical definitions and items. Following the normalization standard of database design, management of chemical entity is divided into two tables which include a chemical definition and chemical item. Chemical definition refers to the definition of a specific chemical instance, while chemical item refers to record status of a specific chemical category. Each chemical instance or item doesn't need to store its own definition pertaining to its category and instead this is stored on its corresponding chemical definition. Thus the chemical definition acts as parent table and the chemical item acts as child table. A chemical item can't be created unless its chemical definition already exists. The user would be allowed to have the option of creating the new definition as the

same time adding the chemical item. In this way, the system will make it convenient for the user by sending just one request for this use case scenario.

Level 3 – 2.1.2 Manage Lab Equipments

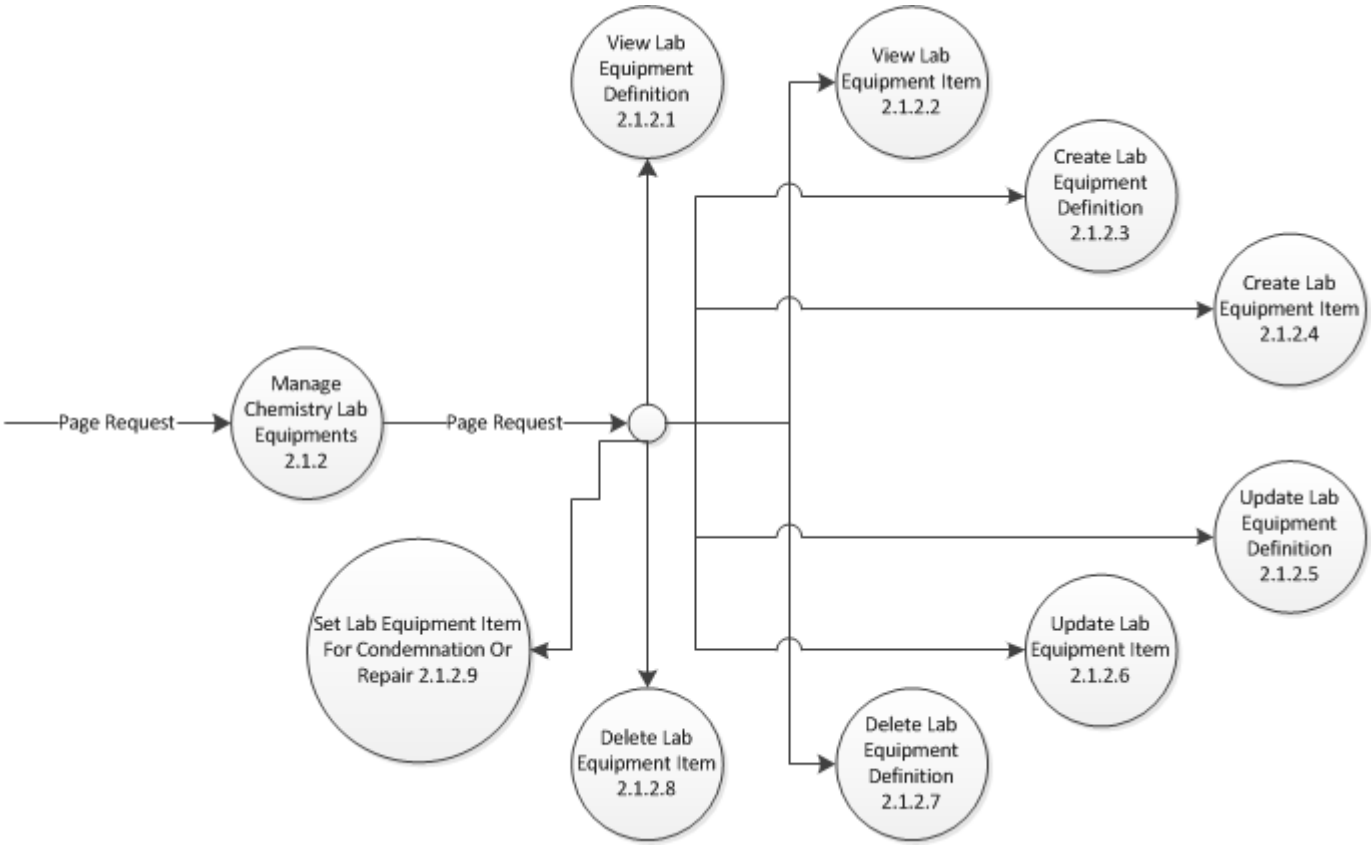

Figure 7 – Manage Lab Equipments

Note: This diagram applies to Chemistry, Physics, Biology, and Math/CS laboratories.

Figure 7 also illustrates the same management flow just like in chemical entity. Only lab attendant has complete access to these processes. It involves creation, updating, deletion and viewing of lab equipment definitions and items. The lab attendant can also declare equipment status for condemnation or repair. Following the normalization standard of database design, management of lab equipment entity is divided into two tables which include the definition and item. Lab equipment definition refers to the definition of a specific lab equipment category, while lab equipment item refers to record status of a specific lab equipment instance. Each lab equipment instance or item doesn't need to store its own definition pertaining to its category and instead this is stored on its corresponding lab equipment definition. Thus the lab equipment definition acts as parent table and the lab equipment item acts as child table. A lab equipment item can't be created unless its lab equipment definition already exists. The user would be allowed to have the option of

creating the new definition as the same time adding the lab equipment item. In this way, the system will make it convenient for the user by sending just one request for this use case scenario.

Level 1 – 3.0 Dispatch Maintenance Requests

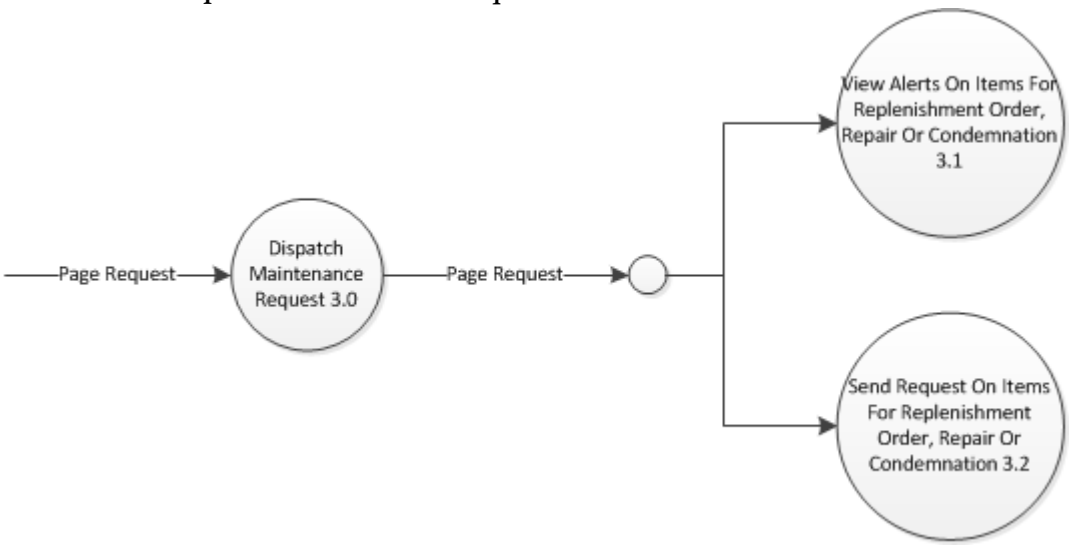

Figure 8 – Dispatch Maintenance Requests

The illustration above presents the access rights of dispatching a request for stock maintenance purposes such as request for order replenishment for items that has reached its minimum quantity deemed by the lab attendant, the request for repair if an equipment is out of order and the request for condemnation of equipment as these statuses are declared by lab attendants. This privilege is available only for lab faculty coordinator.

Level 1 – 4.0 Manage Approval for Maintenance Requests

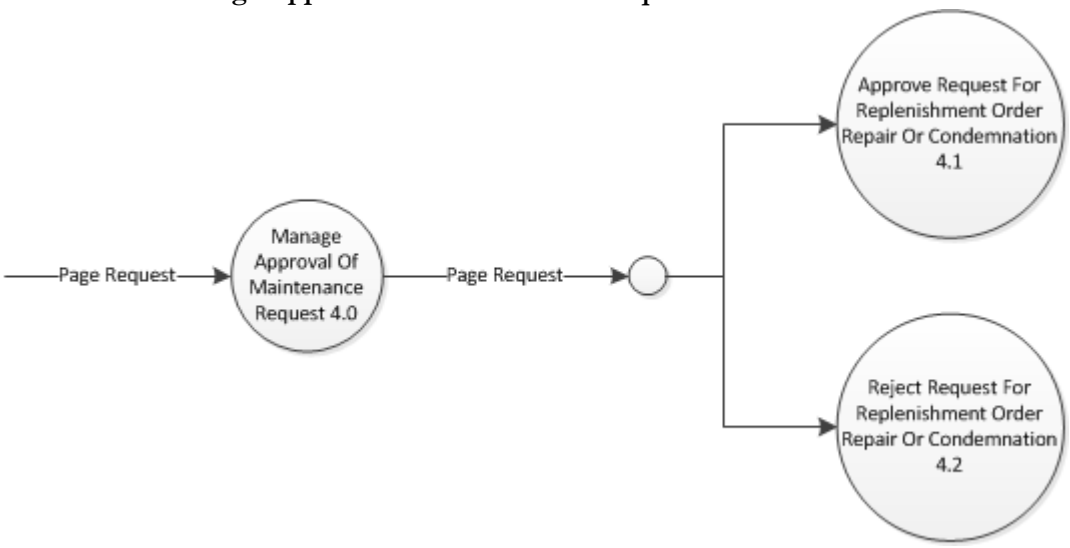

Figure 9 – Manage Approval for Maintenance Requests

The diagram above shows how maintenance requests are received by the supply officer and allows only the supply officer of the privilege to approve or reject any request coming from the lab faculty coordinator.

D. Entity Relationship Diagram

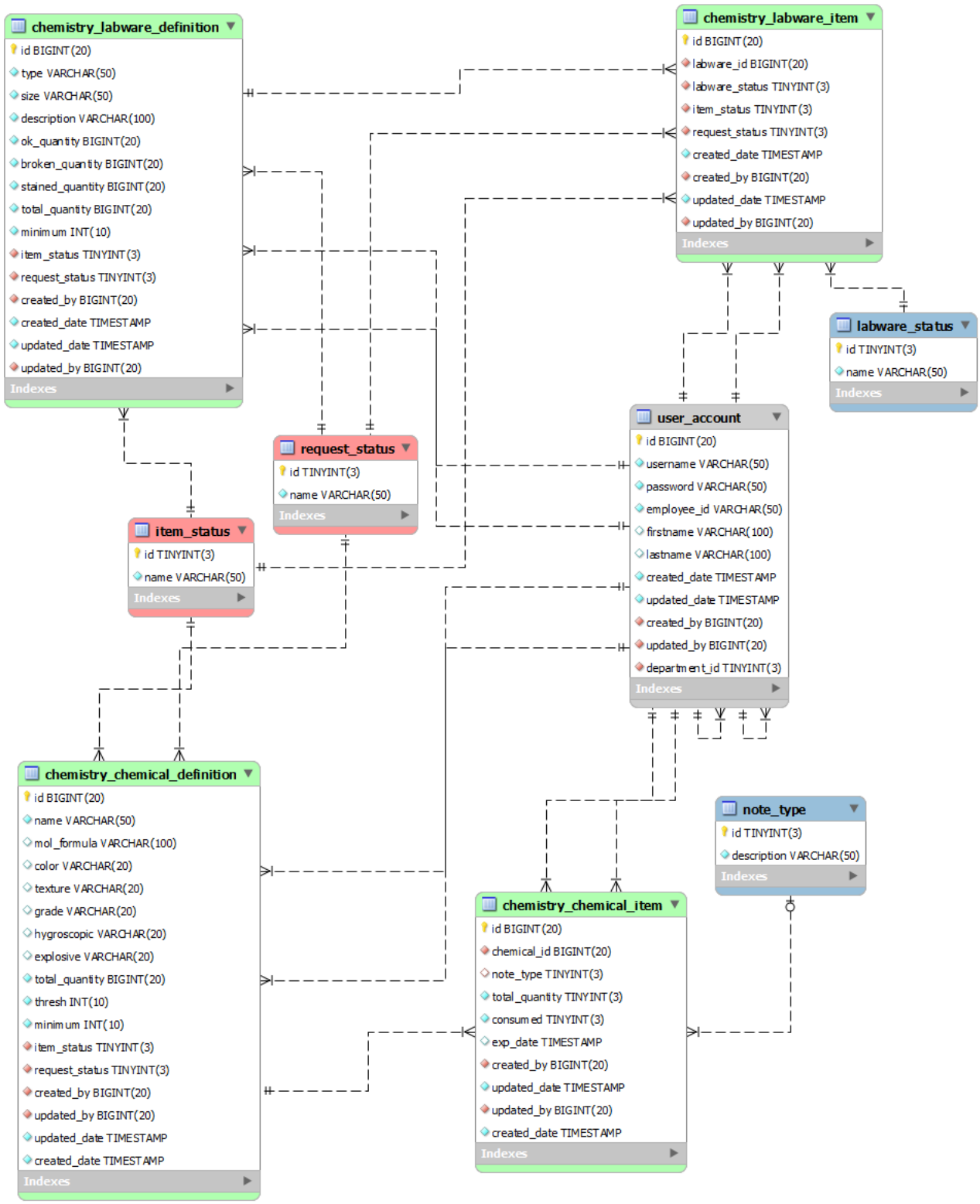

Figure 10 – Chemical and Labware ERD

***Note:** Chemistry, Math/CS, Biology and Physics divisions have their own table for entities such as Chemical definitions, Chemical Items, Lab equipment Definitions and Lab equipment Items, except for Math/CS which is only Machine tables corresponding to Lab equipment tables.

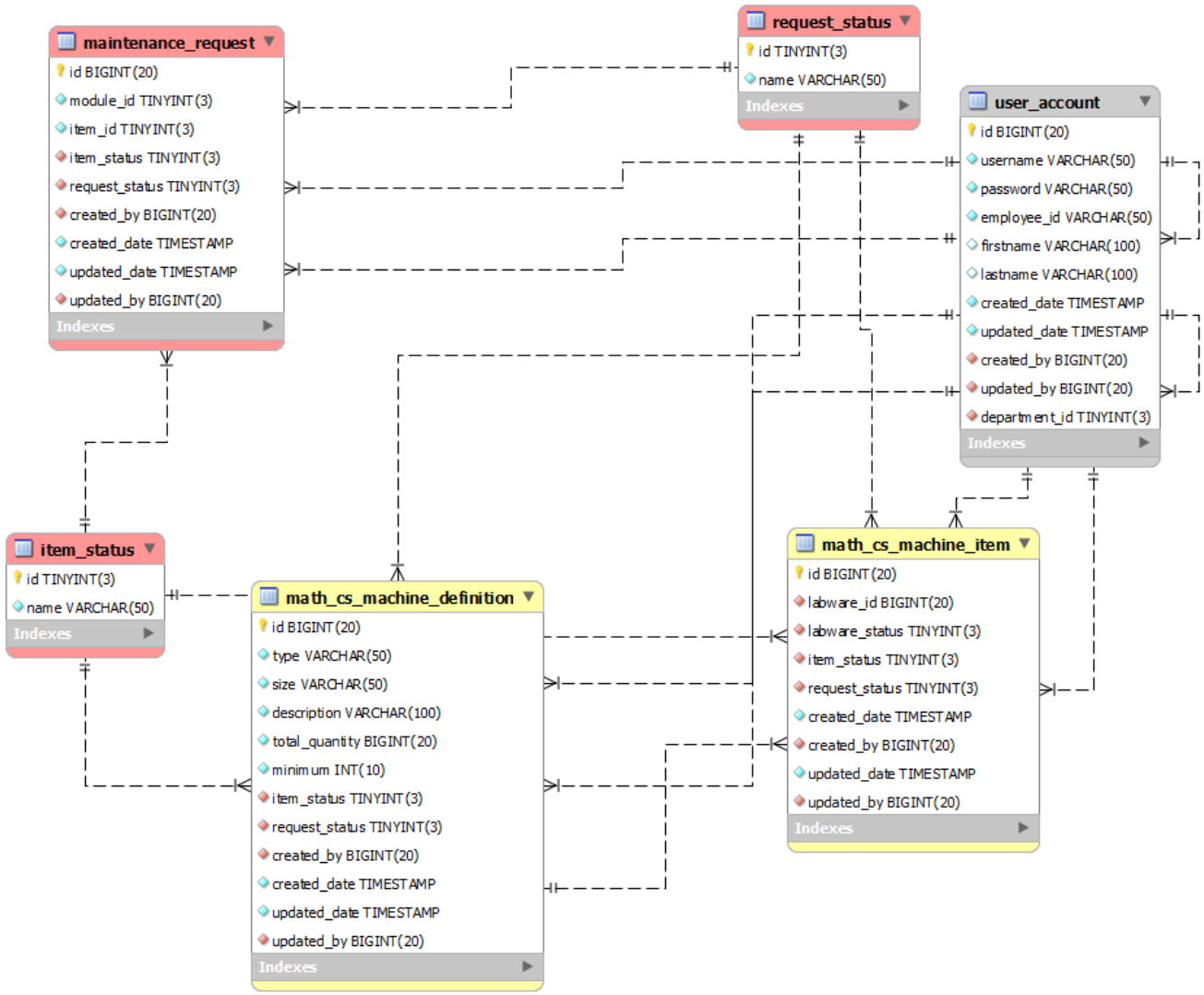

Figure 11 – Maintenance Request ERD

This diagram shows the related tables used in managing maintenance requests.

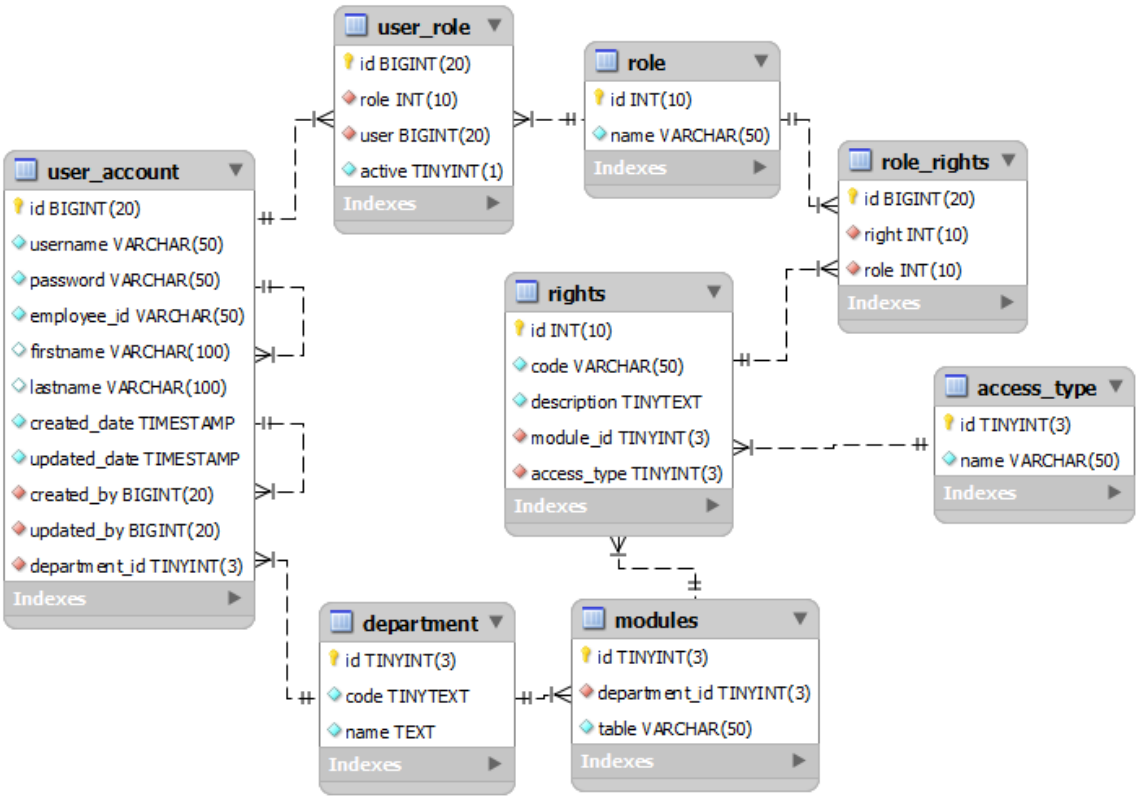

Figure 12 - User Access Rights ERD

The diagram above shows the related tables that comprised the structure needed to implement Role-Based Access Control. This is used in controlling the user privileges in the system.

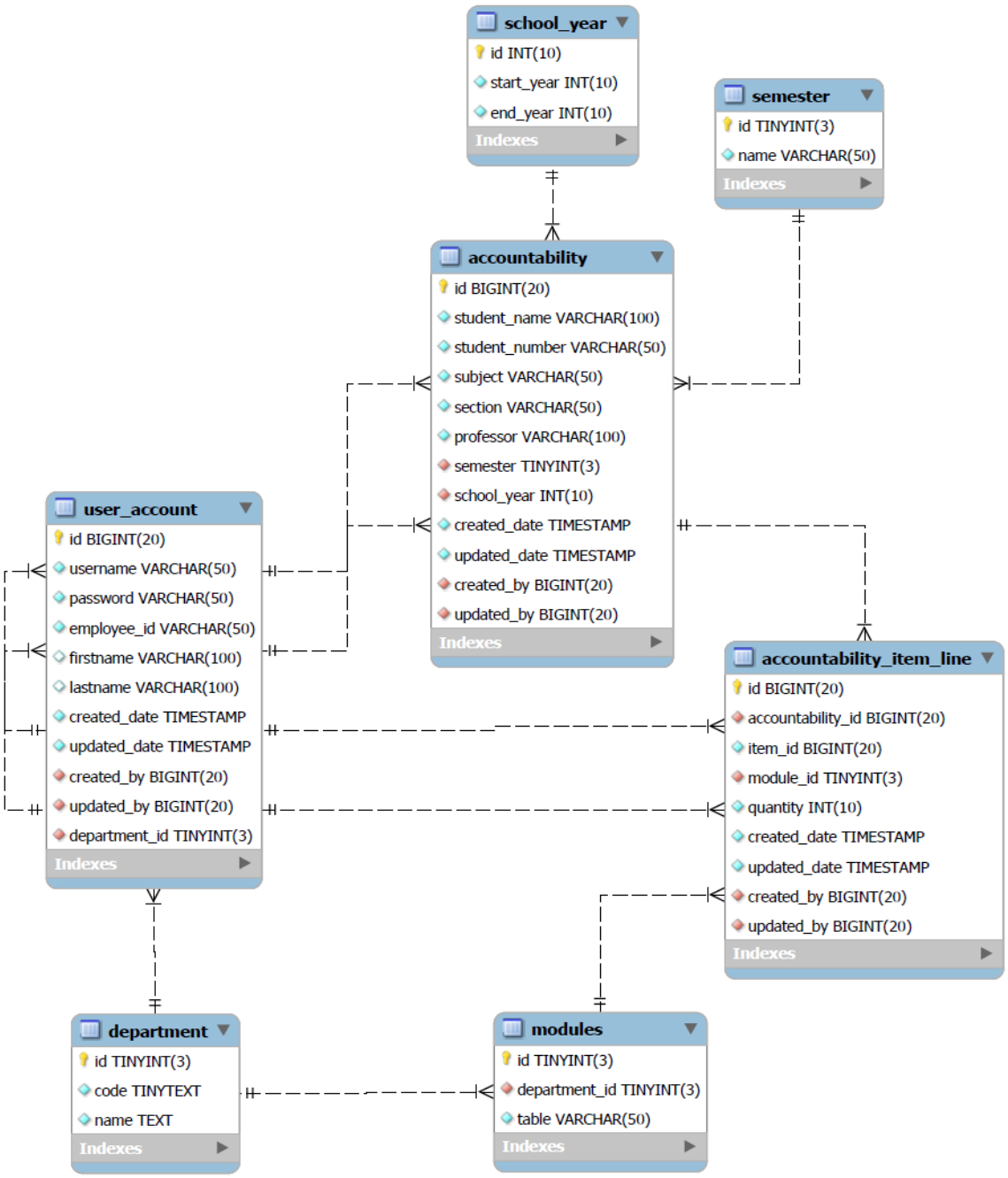

Figure 13 - Accountability ERD

These tables in the diagram above are used to store data of the accountability records for students. This is important to track down student lab deficiencies in every academic semester.

E. Data Dictionary

'[PK]' means Primary key and '[FK]' means Foreign key.

Table 1 - Chemical Definition

The detailed description of the chemicals such as name, molecular formula, color and texture are defined here. It only applies to Chemistry, Physics, and Biology.

Field Name	Field Type	Description
Id [PK]	bigint	Id of chemical definition
Name	varchar	Chemical name
mol_formula	varchar	Molecular formula
Color	varchar	Color of chemical
Texture	varchar	texture of chemical
Grade	varchar	grade of chemical
hygroscopic	varchar	hygroscopic definition
explosive	varchar	explosive feature
total_quantity	bigint	total quantity of chemical item of this definition
Thresh	int	Status id of the request
minimum	int	Minimum quantity for this item
Item_status [FK]	Tinyint	Status id of the concerned item
Request_status [FK]	Tinyint	Status id of the request
created_by [FK]	int	user id who created the record
updated_by [FK]	int	id the last user that updated this record

created_date	timestamp	the creation date of the record
updated_date	timestamp	the latest update date

Table 2 - Chemical Item

The individual chemical items are defined here. It uses the Chemical Definition table as its definition. It only applies to Chemistry, Physics, and Biology

Field Name	**Field Type**	**Description**
Id [PK]	autoint	Id of batch chemical item
chemical_id [FK]	int	Id of parent chemical definition
consumed	tinyint	Status whether this is already consumed
note_type [FK]	tinyint	Note id for this record
Total_quantity	tinyint	Total quantity for this batch items
exp_date	varchar	Expiration date of this item
created_by [FK]	int	user id who created the record
updated_by [FK]	int	id the last user that updated this record
created_date	timestamp	the creation date of the record
updated_date	timestamp	the latest update date

Table 3 - Note Type

It is basically the reference table of table Chemical Item on column note_type. It stores description on the chemical item.

Field Name	Field Type	Description
Id [PK]	tinyint	identifier
description	varchar	note type description

Table 4 - Machine/Lab Equipment Definition

The detailed description of the lab equipments such as type, size and description are defined here. It applies to Chemistry, Physics, and Biology, Math/CS

Field Name	Field Type	Description
Id [PK]	bigint	Id of lab equipment definition
Type	varchar	lab equipment type
Size	varchar	lab equipment size
description	varchar	lab equipment description
ok_quantity	bigint	total count of usable lab equipment
broken_quantity	bigint	total count of broken lab equipment
stained_quantity	bigint	total count of stained lab equipment
Total_quantity	bigint	Total quantity present for this item
minimum	int	Minimum quantity for this item
Item_status [FK]	tinyint	Status id of the concerned item

request_status [FK]	tinyint	Status id of the request
created_by [FK]	bigint	user id who created the record
updated_by [FK]	bigint	id the last user that updated this record
created_date	timestamp	the creation date of the record
updated_date	timestamp	the latest update date

Table 5 - Machine/Lab Equipment Item

The individual lab equipments are defined here. It uses the Lab Equipment Definition table as its definition. It applies to Chemistry, Physics, and Biology, Math/CS

Field Name	**Field Type**	**Description**
Id [PK]	bigint	Id of lab equipment item
labware_id [FK]	bigint	Id of parent lab equipment definition
Labware_status [FK]	tinyint	Id refers to the status of the item
Item_status [FK]	tinyint	Status id of the concerned item
request_status [FK]	tinyint	Status id of the request
created_date	timestamp	the creation date of the record
updated_date	timestamp	the latest update date
created_by [FK]	bigint	user id who created the record
updated_by [FK]	bigint	id the last user that updated this record

Table 6 - Lab Equipment Status

It is basically the reference table of table Lab Equipment Item on column labware_status. It stores description on the lab equipment.

Field Name	Field Type	Description
Id [PK]	tinyint	identifier
Name	varchar	Status name

Table 7 - Maintenance Request

The request used in declaring order of replenishment, declaring equipment as to be repaired or condemned is defined here.

Field Name	Field Type	Description
Id [PK]	bigint	identifier
module_id	tinyint	Module identifier this request is associated
item_id	bigint	Identifier of associated laboratory item of the module id specified in the previous column
item_status [FK]	tinyint	Status id of the concerned item
Request_status [FK]	tinyint	Status id of the request
Created_by [FK]	bigint	Account id of the creator
Created_date	timestamp	Timestamp when this record is created
Updated_by [FK]	bigint	Account id of the last user updated this record

Updated_date	timestamp	Timestamp when this is last updated

Table 8 - Request Status

It is basically the reference table of table Maintenance Request on column request_status. It stores description on the request.

Field Name	Field Type	Description
Id [PK]	tinyint	Status id
Name	varchar	Status name

Table 9 - Item Status

It is basically the reference table of table Maintenance Request on column item_status. It stores description on the item.

Field Name	Field Type	Description
Id [PK]	tinyint	identifier
Name	varchar	Item Status name

Table 10 - Modules

It is basically the reference table of table Rights on column module_id. It stores description on the module.

Field Name	Field Type	Description
Id [PK]	tinyint	identifier
department_id [FK]	tinyint	department id
Table name	varchar	Actual table name from Database, used in security Implementation design

Table 11 - Departments

It is basically the reference table of table Modules on column department_id. It stores the department in text.

Field Name	Field Type	Description
Id [PK]	tinyint	identifier
Code	varchar	Short unique code such as DPSM
Name	varchar	department name

Table 12 - User Account

The user account which contains basic information on the user such as the username, password and employee id are defined here.

Field Name	Field Type	Description
Id [PK]	bigint	identifier
username	varchar	Chemical name
password	varchar	Molecular formula
employee_id	varchar	Color of chemical
firstname	varchar	texture of chemical
Lastname	varchar	grade of chemical
user_type_id	tinyint	hygroscopic definition
department_id [FK]	tinyint	Department identifier
created_by [FK]	bigint	user id who created the record
updated_by [FK]	bigint	account id the last user that updated this record

created_date	timestamp	the creation date of the record
updated_date	timestamp	the latest update date

Table 13 - Role

It is the referenced table used as storage for the role of each user.

Field Name	**Field Type**	**Description**
Id [PK]	bigint	identifier
Name	varchar	Description for role

Table 14 - Rights

It is the referenced table used as storage for the rights of each user. It usually depends on the module a user is in.

Field Name	**Field Type**	**Description**
Id [PK]	bigint	identifier
Code	varchar	Short unique code for this rights
Description	tinytext	Description of this rights
Module id [FK]	tinyint	The module id which this rights would be applied
Access type [FK]	tinyint	Id of the access type of this operation

Table 15 - Accountability

It registers student with accountability to the system. One registry may contain information such as name, student number, subject, section and professor.

Field Name	Field Type	Description
Id [PK]	tinyint	identifier
Student name	varchar	Student full name
Student number	varchar	Student number
Subject	varchar	Subject of the class
Section	varchar	Class section
professor	varchar	Class professor full name
Semester [FK]	tinyint	Id of the semester when this accountability is produced
academic year [FK]	int	Id of the academic year when this accountability is produced

Table 16 - Accountability Item

The Accountability table is used here as a definition. It stores the item which is involved in the accountability of the student.

Field Name	Field Type	Description
Id [PK]	bigint	identifier
Accountability_id [FK]	varchar	Id of the parent accountability record
module_id [FK]	bigint	id of the module where this item is belong
Item_id	tinyint	id of the tem concerned

Quantity	int	Quantity for this item
created_by [FK]	bigint	user id who created the record
updated_by [FK]	bigint	account id the last user that updated this record
created_date	timestamp	the creation date of the record
updated_date	timestamp	the latest update date

Table 17 - Semester

The current semester the accountability is in.

Field Name	**Field Type**	**Description**
Id [PK]	tinyint	identifier
Name	varchar	Semester name

Table 18 - Academic Year

The current academic year the accountability is in. It comprises of the start year and the end year of the accountability.

Field Name	**Field Type**	**Description**
Id [PK]	int	identifier
Start year	int	Start year
End year	int	End year

V. Architecture

A. System Architecture

The presentation layer is created using HTML5 and made use of JavaScript and jQuery for some on-page functionalities. For the service layer, Java is used. The Model-View-Controller Framework Spring is used for implementing user interfaces while employing Java in the processing of data. Finally, the database layer is composed of the relational database management system MySQL. All information that needs to be stored for future reference made use of MySQL for recording.

B. Technical Architecture

- The client in this context is the computer browser used by the user.
- The server is the remote computer that runs the server application and database.
- When the client needs data (a HTML page, for instance) it sends a request to the server, and the server responds by sending the requested data.
- After getting this response, the client may send a new request, which leads to a new response, and so on.
- CAS-LIMS is online-based and made use of the following:
    1. Spring Framework
    2. Java
    3. MySQL database server
    4. Javascript
    5. JQuery
    6. HTML5

C. System Requirements

Client

Minimum Hardware Requirements:

- Pentium/AMD 1 GHz processor
- 512MB RAM
- At least 120GB hard drive
- Existing LAN/WAN and Internet connection

Minimum Software Requirements:

- Platform - Linux/Windows
- Web Browsers: Mozilla Firefox 16+/Opera 10+/IE 8+/Chrome 15+

Server

Minimum Hardware Requirements:

- Pentium/AMD 1 GHz processor
- 1GB RAM
- At least 120GB hard drive
- Existing LAN/WAN and Internet connection

Minimum Software Requirements:

- Platform - Linux/Windows
- DBMS: MySQL version 5.6.12
- Server Container : Jetty 7.6/ Apache Tomcat 6
- JRE 6

VI. Results

A. General View

Login Page

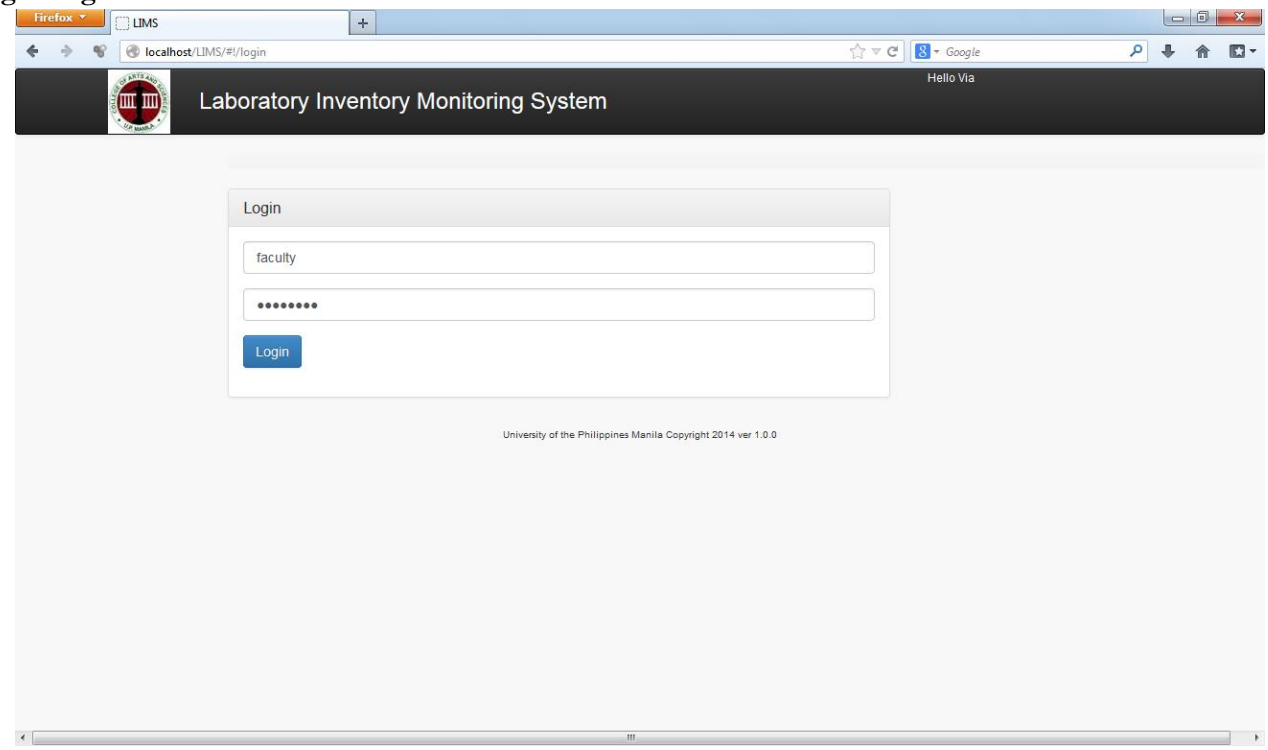

Figure 14: Login Page

This is the home page of CAS-LIMS. See Figure 14. The System Administrator, Lab Attendant, Lab Faculty Coordinator, and Supply Officer can login to their respective accounts through this page.

B. System Administrator View

Manage Users

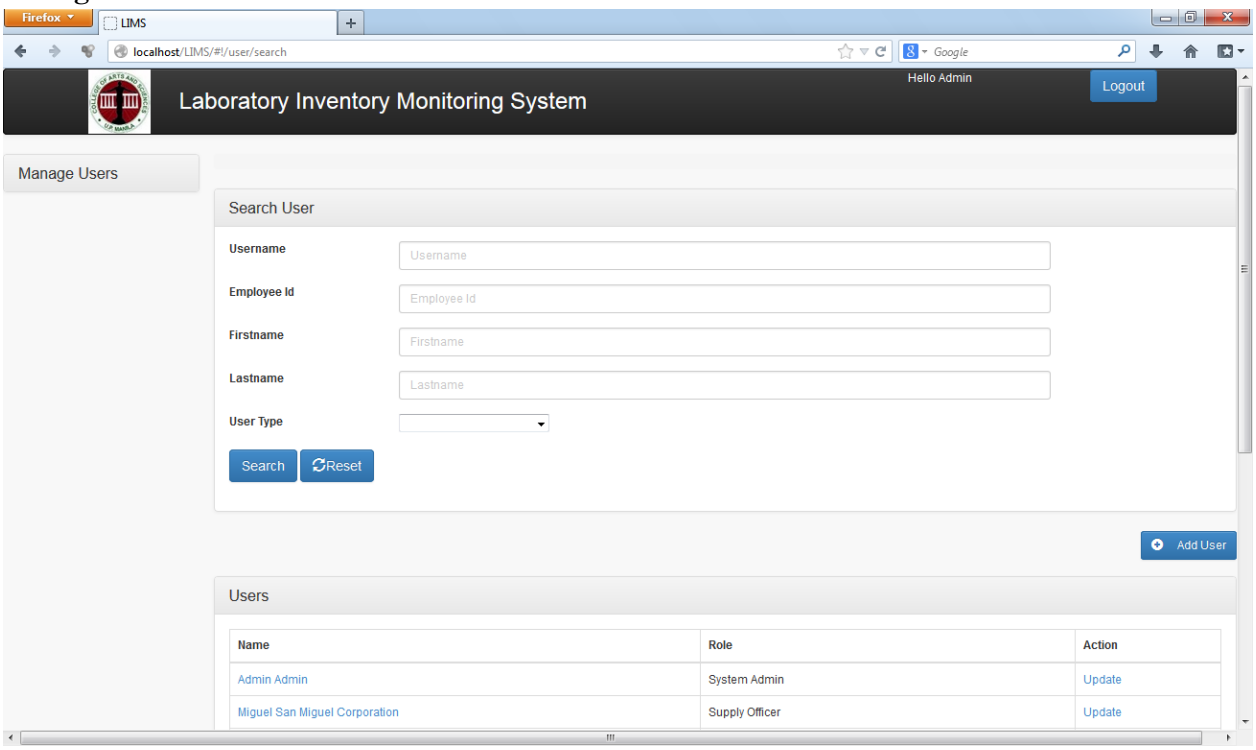

Figure 15: Search User

The System Admin can search for a user. See Figure 15. The search result will be displayed at the table at the bottom.

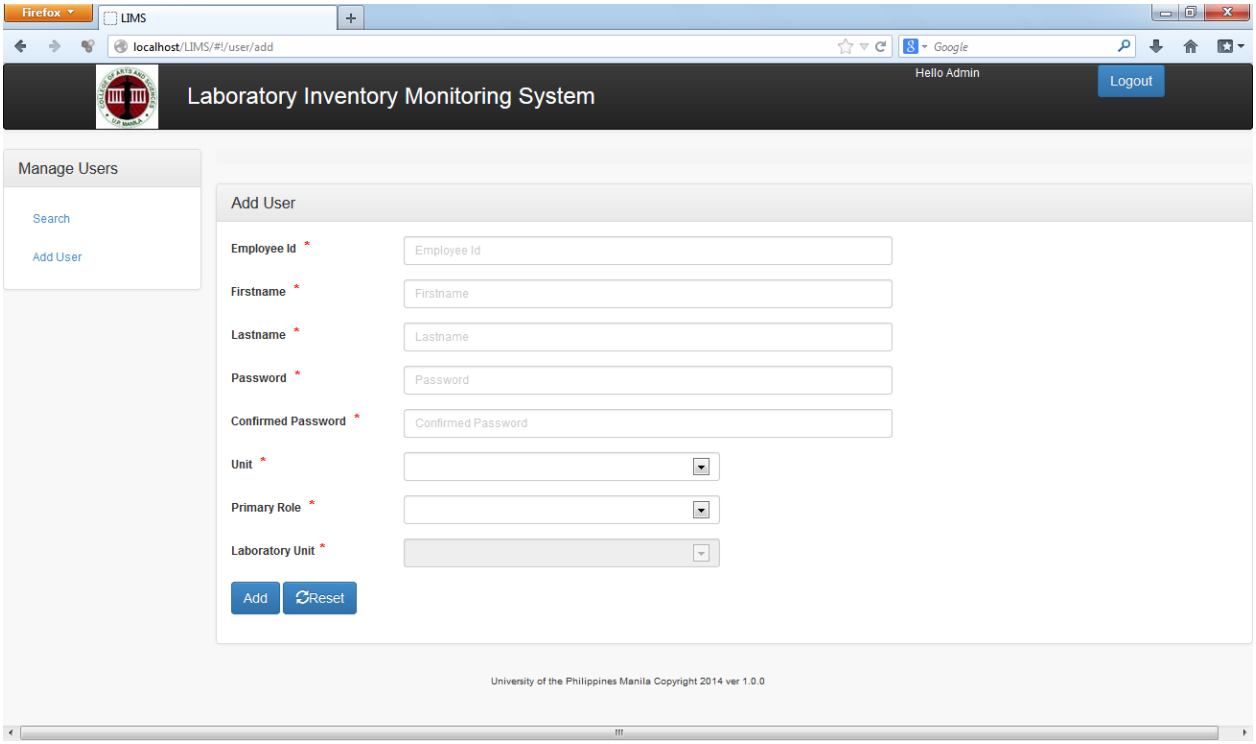

Figure 16: Add User

The System Admin can also add user to the database. See Figure 16. The entries with a red asterisk after it is indicated as required.

C. Lab Attendant View

View Dashboard

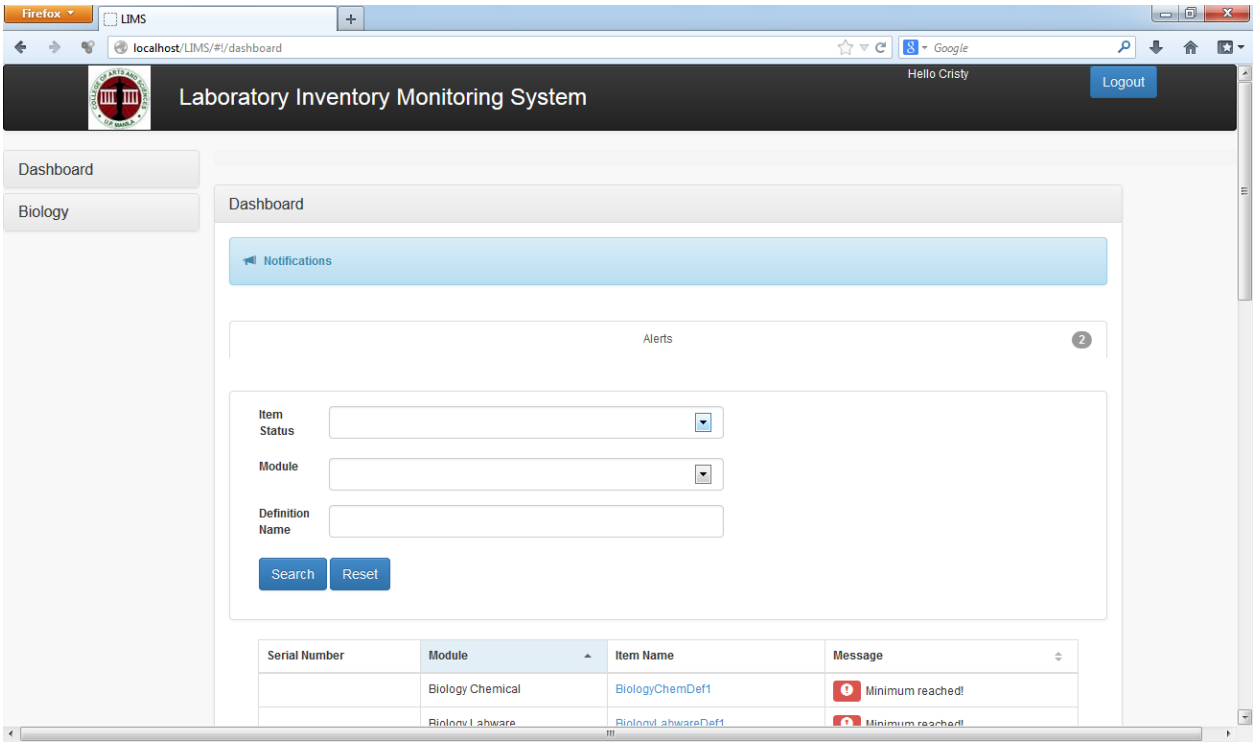

Figure 17: View Lab Attendant DashBoard

The lab attendant can view the alerts on the dashboard. See Figure 17.

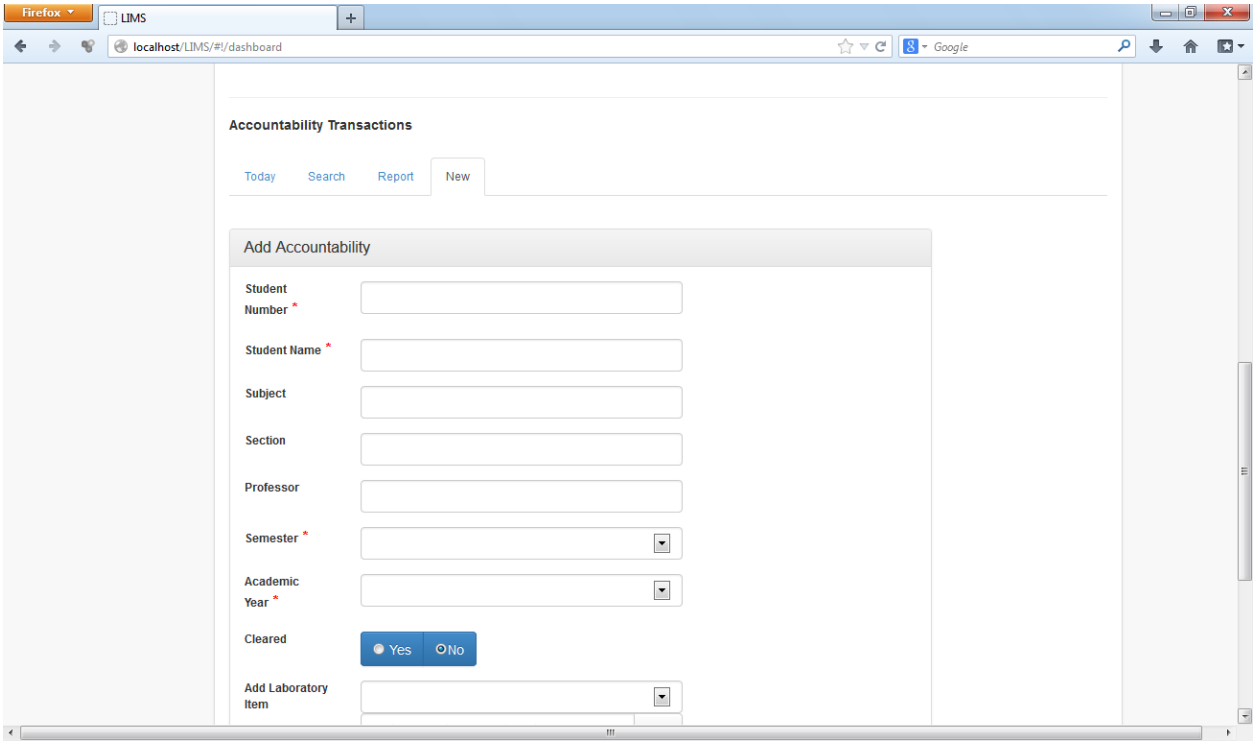

Figure 18: Add Accountability

The lab attendant can add accountability whether it be a chemical or a labware/machine. See Figure 18.

View Chemistry/Physics/Biology/MathCS Panel

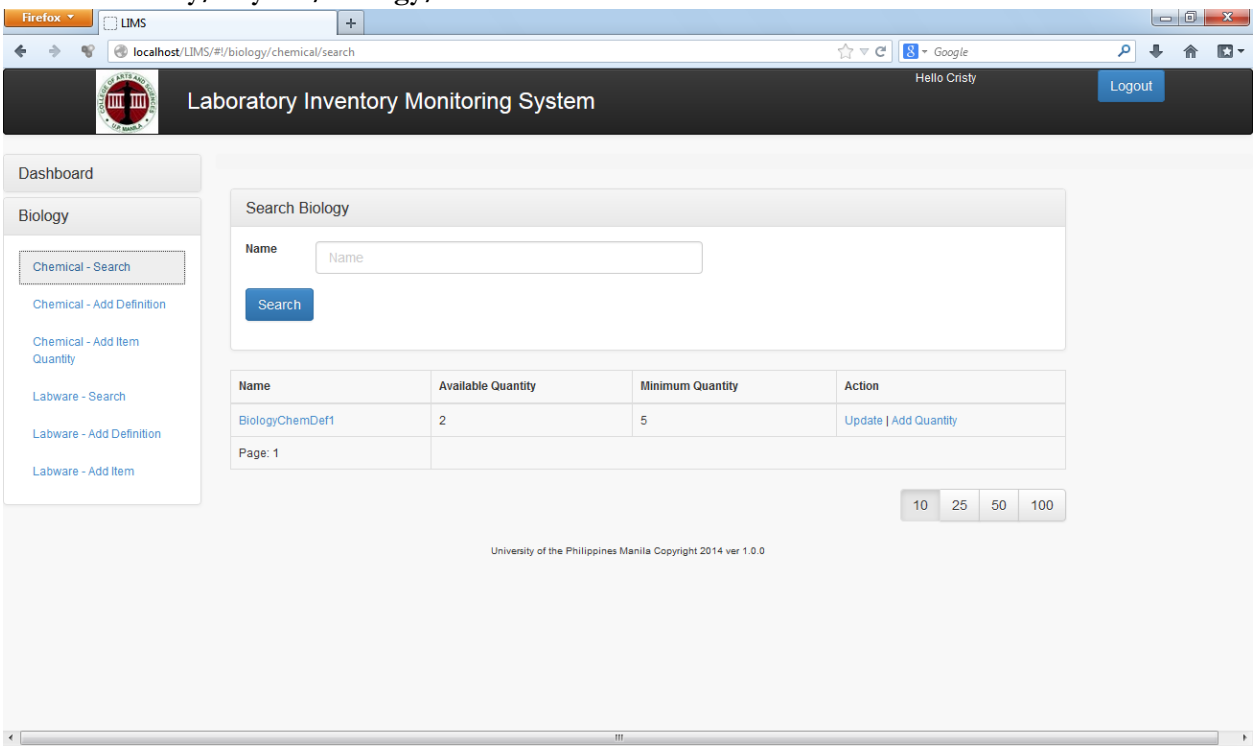

Figure 19: Chemical-Search

The lab attendant can search for a chemical item, update the information regarding the definition of the item and add quantity of the item. See Figure 19.

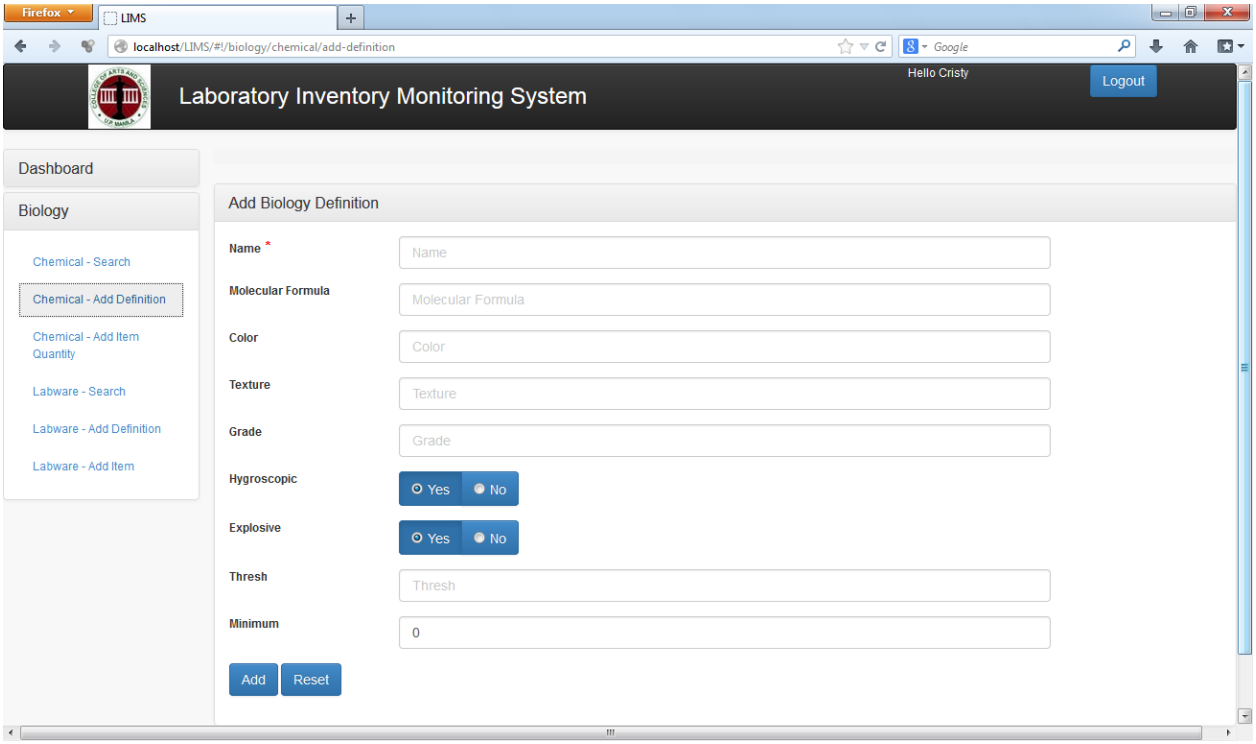

Figure 20: Chemical-Add Definition

Showing is the page for the addition of a chemical definition. See Figure 20. Remember that an asterisk after an entry indicates that the field is required.

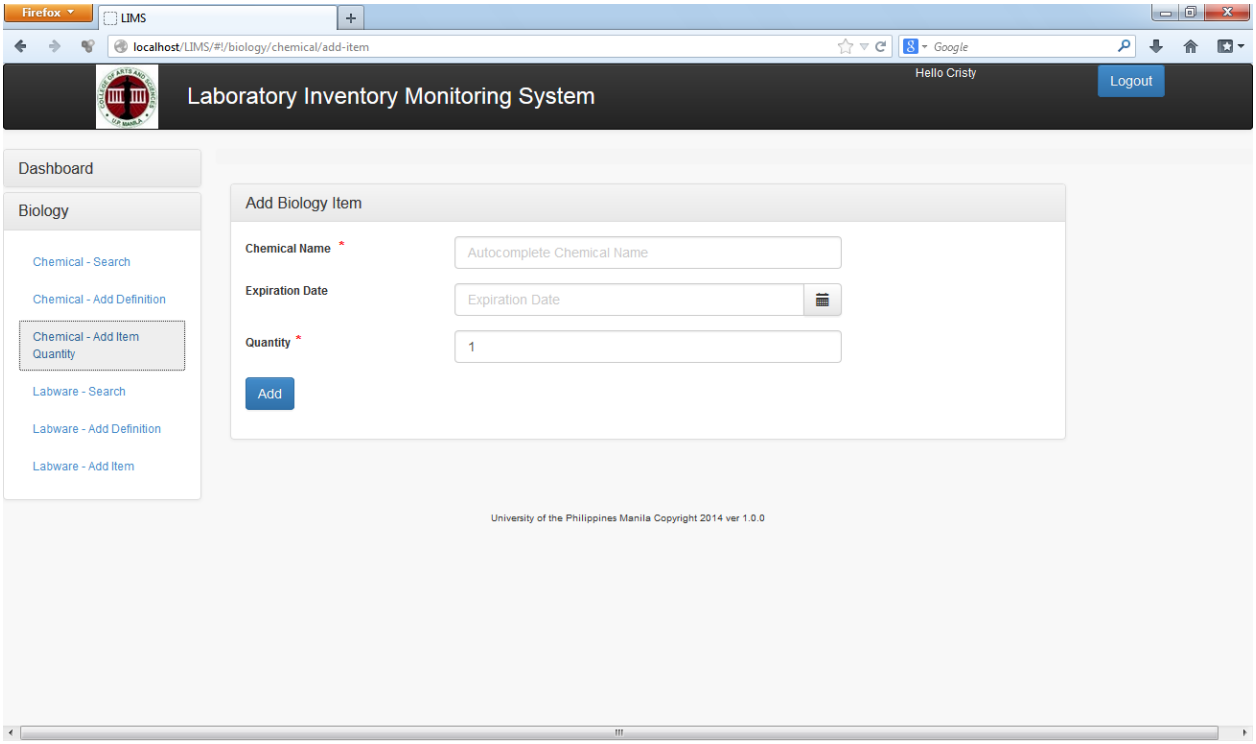

Figure 21: Chemical-Add Item Quantity

The lab attendant can add a chemistry item quantity. See Figure 21.

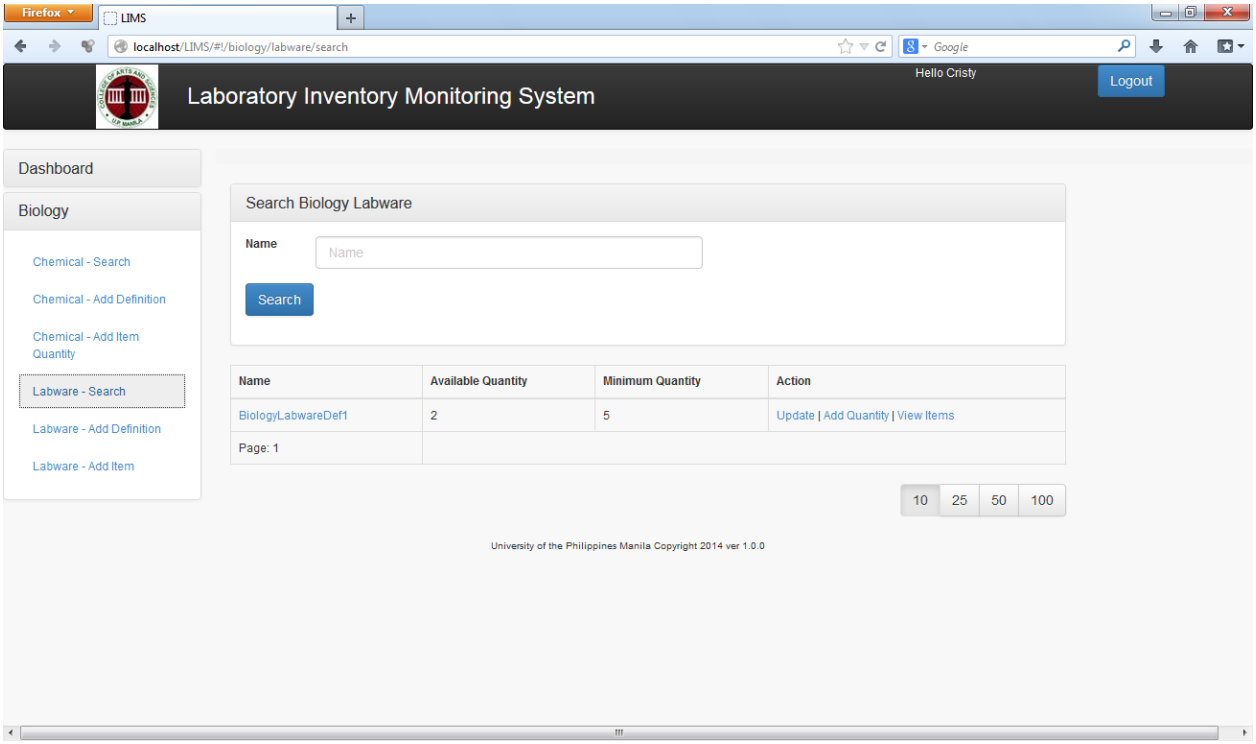

Figure 22: Labware Search

The Lab attendant can search for a labware item, update its definition, and add the respective quantity. See Figure 22.

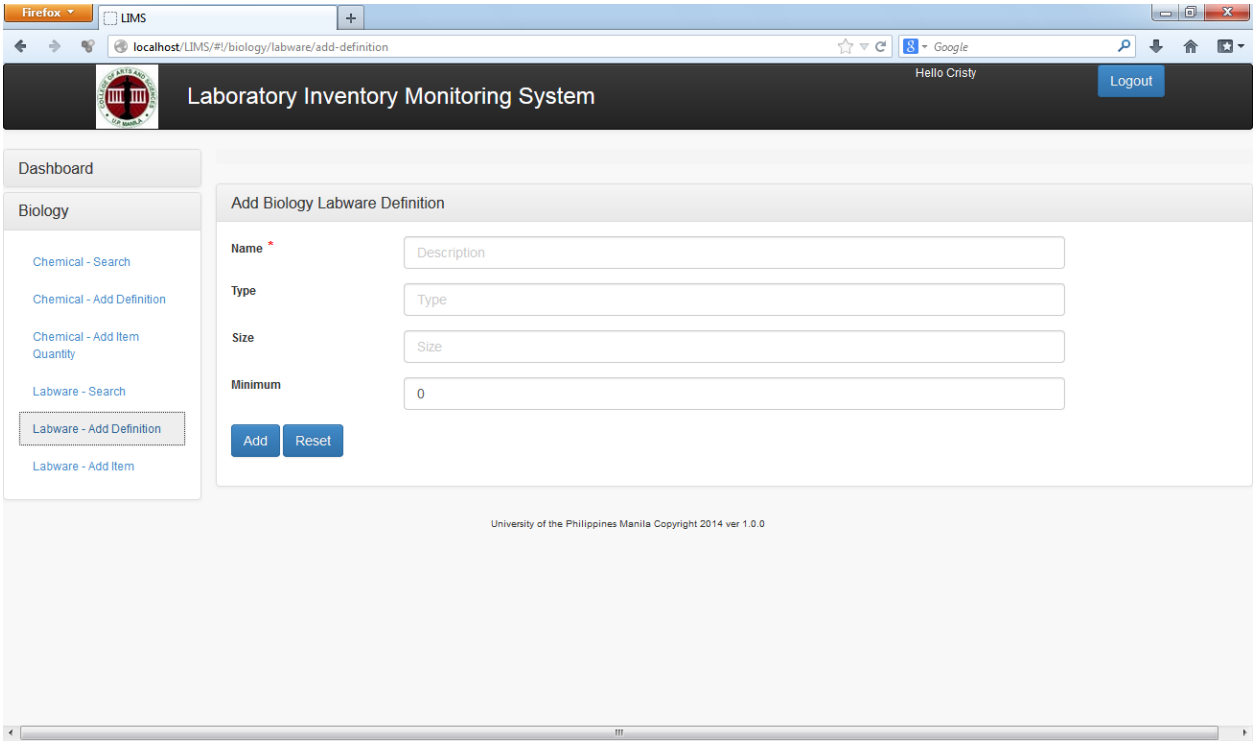

Figure 23: Labware: Add Definition

The Lab Attendant is also responsible for the adding of the definition of a labware. See Figure 23. Remember that the asterisk indicates that the field is required.

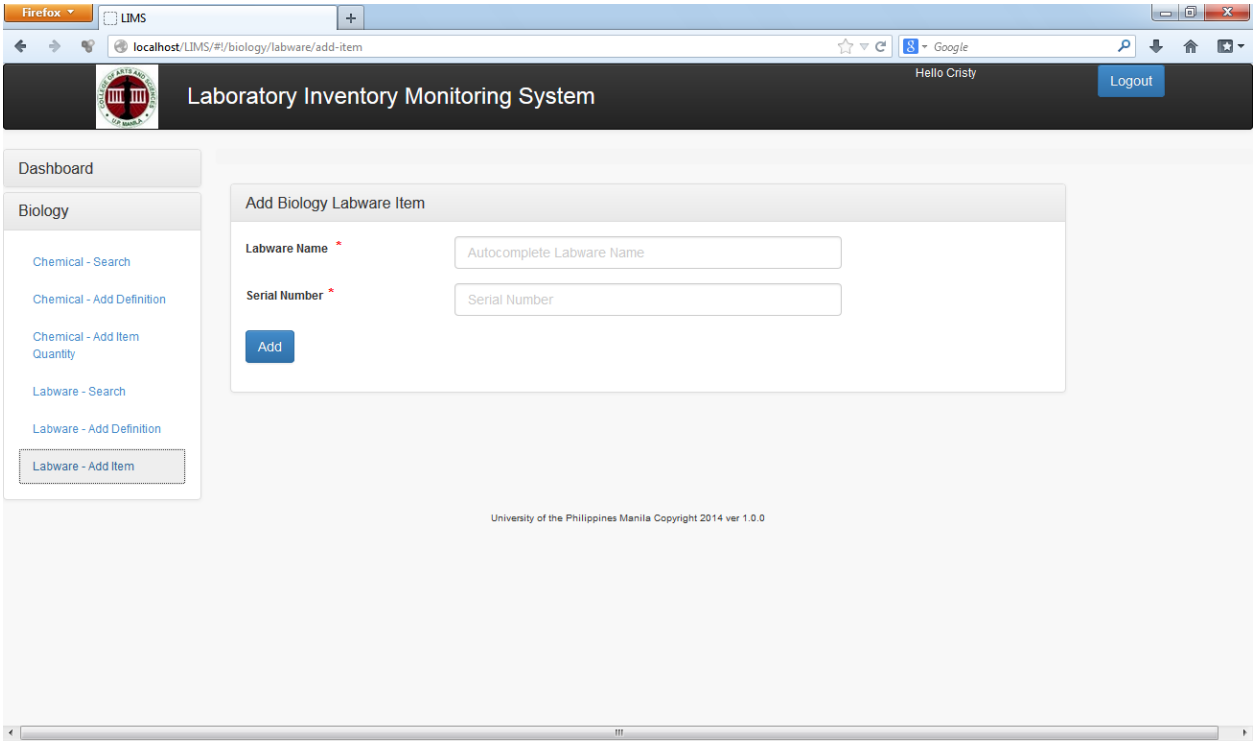

Figure 24: Labware: Add Item Quantity

The lab attendant can update the number of labware items. See Figure 24.

D. Lab Faculty Coordinator View

View Dashboard

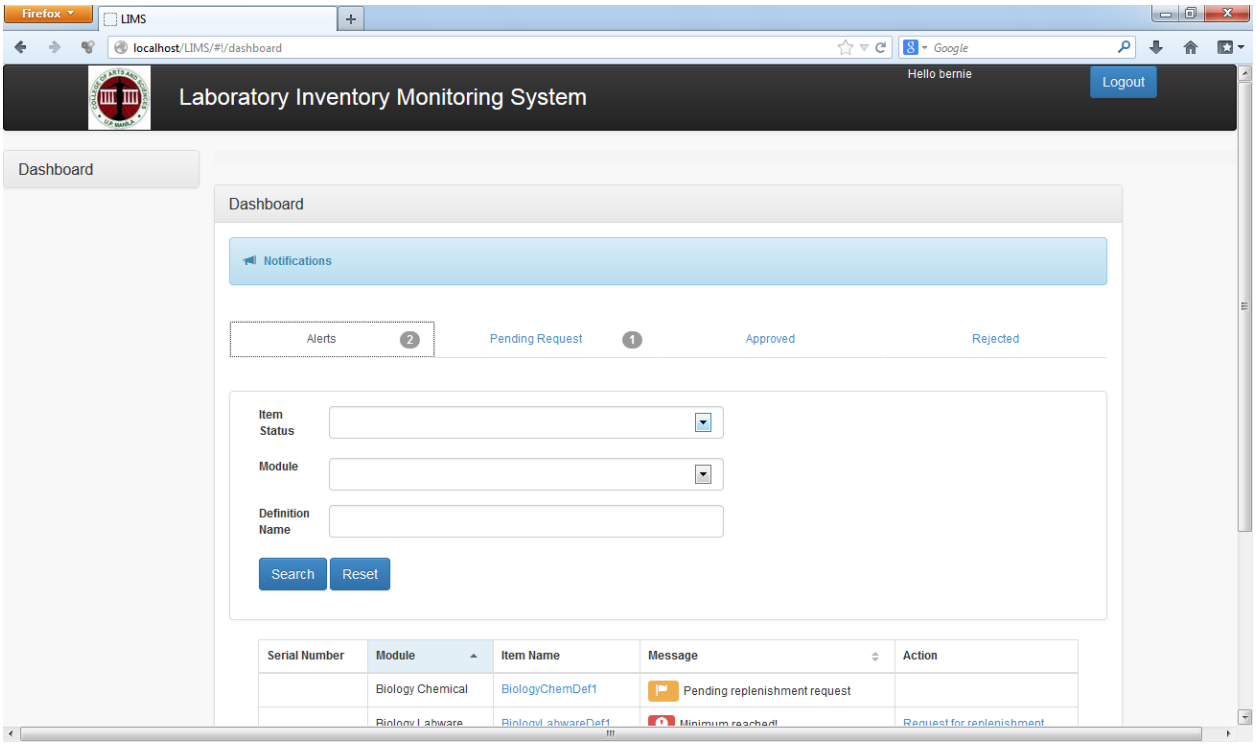

Figure 25: View Lab Faculty Coordinator Dashboard

The Lab Faculty Coordinator can view alerts, request for replenishment, can view pending requests, approved requests and rejected requests for its respective discipline. See Figure 25.

E. Supply Officer View

View Dashboard

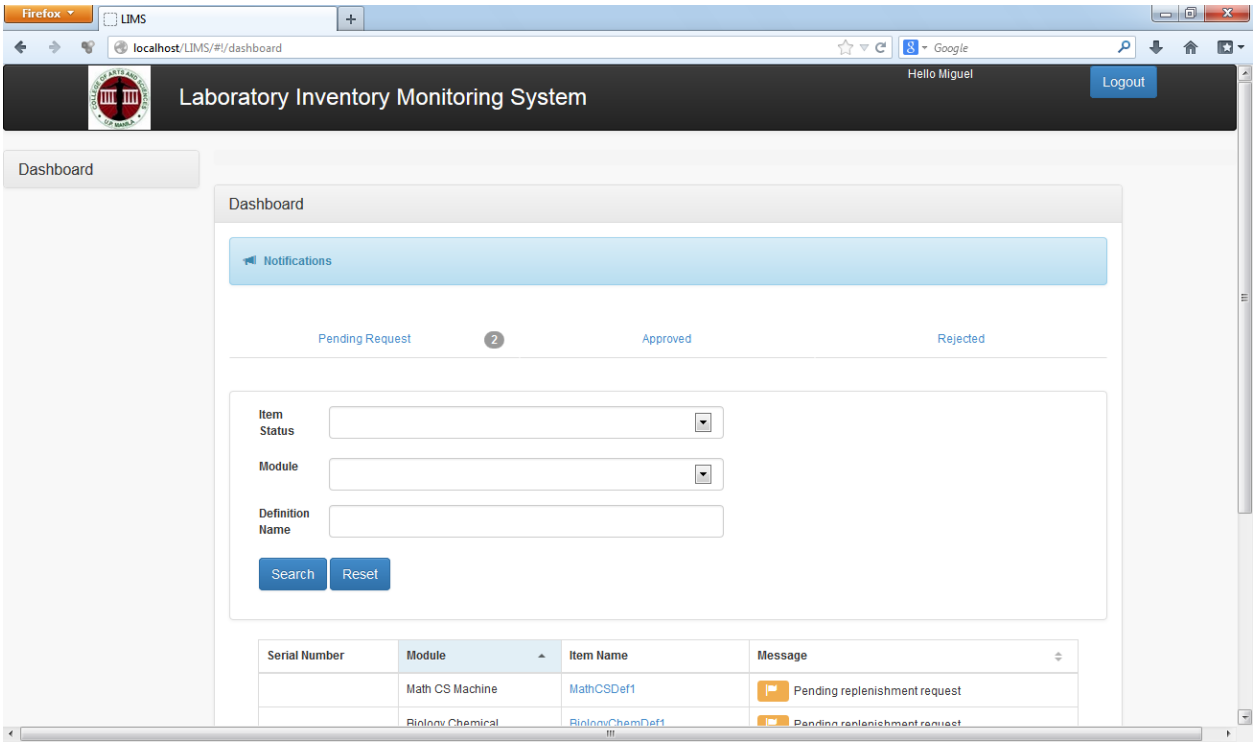

Figure 26: View Supply Officer Dashboard

The Supply Officer can view alerts, can approve or reject pending requests. See Figure 26.

VII. Discussions

The CAS-LIMS is an automation of the procedures done in monitoring the supplies and behavior of transactions within the different laboratory disciplines. To accomplish separation of concerns, one must be able to integrate knowledge of Spring MVC framework with that of Angular JS framework. Angular JS is a helpful tool in minimizing the code required in implementing the separation of views for the different modules of lab attendant. It automates the changes made from one page to another by means of a simple Javascript protocol. It can also help in the specification of the pages accessible to one user. Because of this, lab attendant users can only view alerts related to their own unit and the lab faculty coordinator can view its own dashboard. The user accounts were also created by means of the Javascript protocol which is handled by the AngularJS.

VIII. Conclusion

The main goal of CAS-LIMS is to provide automation for the monitoring of the laboratory supplies and labwares and this goal has been met. The current method which lacks data consistency, security and reliability has been changed to a more consistent, secure and reliable web interface. The resulting method is also not tedious and not vulnerable to human error anymore since accessing information is done over the monitoring system by using the search functionalities. Also the definition of the chemicals and laboratory equipments are done over the system so the information could persist throughout the entire recording season. Laboratory instructors would only have to check the system for the quantity of the laboratory items to get information of the availability of such item.

IX. Recommendations

CAS-LIMS is flexible enough with its inventory monitoring system, however it is recommended that the addition of other functionalities such as creation of new College Admin to monitor or create accounts for his/her respective department faculty coordinator of lab attendant be implemented. It will include an interface to add another monitoring method that is unique in one college. Inclusion of this feature would make CAS-LIMS more flexible.

X. Bibliography

1 - A. N. Mustafizul Karim, et. al., Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS) 2 (1): 36

2 - Mathaba, S, Dlodlo,N, Smith, A and Adigun, M. "The use of RFID and Web 2.0 Technologies to Improve Inventory Management in South African Enterprises" The Electronic Journal Information Systems Evaluation Volume 14 Issue 2 2011, (pp228-241), available online at www.ejise.com

3 - S.K. Yadav et. al "EOQ model for deteriorating items with exponential demand rate and shortages" Uncertain Supply Chain Management 1 (2013)

4 – G. K. Yang "Discussion of arithmetic defuzzifications for fuzzy production inventory models" African Journal of Business Management Vol. 5(6), pp. 2336-2344, 18 March, 2011 Available online at http://www.academicjournals.org/AJBM

5 – M-Y Liao , et. al "A study of supply chain replenishment system of theory of constraints for thin film transistor liquid crystal display (TFT-LCD) plants" African Journal of Business Management Vol. 5(21), pp. 8617-8633, 23 September, 2011 Available online at http://www.academicjournals.org/AJBM

6 – C. X. Lou,W. Dai "Optimal Supply Chain Services Management for SMEs through Integrated Model-driven Service System" ICIW 2012 : The Seventh International Conference on Internet and Web Applications and Service. ISBN: 978-1-61208-200-4

7 – M. Mazanai "Impact of just-in-time (JIT) inventory system on efficiency, quality and flexibility among manufacturing sector, small and medium enterprise (SMEs) in South Africa" African Journal of Business Management Vol. 6(17), pp. 5786-5791, 2 May, 2012 Available online at http://www.academicjournals.org/AJBM

8 - P.C. Yang, H.M. Wee "A collaborative inventory system with permissible delay in payment for deteriorating items" Mathematical and Computer Modelling 43 (2006) 209–221

9 - H.-M. Wee**,** S.-L. Chung et al. "Optimal policy for a closed-loop supply chain inventory system with remanufacturing" Mathematical and Computer Modelling 48 (2008) 867–881

10 - Sharif "A Comparison of Purchase and Inventory Management System of two Educational Institutes"

Other References

1. Coronel, C., &Morris, S., & Rob, P. (2010). Database Systems: Design, Implementation and Management, 9th Edition, 5, Joe Sabatino.

2. Connolly T., Begg C. , Database Systems : A Practical Approach to Design, Implementation and Management, Pearson Education Limited, Fourth Edition, p. 5, 2005

3. Russel D. MySQL in a nutshell, O'Reilly Media, Inc., 2nd Edition, p. 3, 2008

4. Niemeyer P. ,& Knudsen J. , Learning Java?, O'Reilly Media, Inc., Third Edition, 2, 2005

5. MINTER ,D.(2008) Beginning Spring 2: From Novice to Professional ,2,4-5, Apress.

6. MINTER ,D. Beginning Spring 2: From Novice to Professional ,1- 2, Apress., 2008

7. MINTER ,D. Beginning Hibernate: From Novice to Professional , Apress, p.1,2006

8. Hogan B., HTML5 and CSS3: Develop with Tomorrow's Standards Today, Pragmatic Programmers, LLC , p. 10, 2010

9. Hogan B., HTML5 and CSS3: Develop with Tomorrow's Standards Today, Pragmatic Programmers, LLC , p. 15 -17, 2010

10. Pollock, J. Javascript A Beginner's Guide, Third Edition, 2, McGraw-Hill Companies.

11. Chaffer, J., & Swedberg, K. (2009) Learning jQuery 1.3 , 1, Packt Publishing Ltd.

XI. Appendix

A. Forms

Inventory of Chemicals

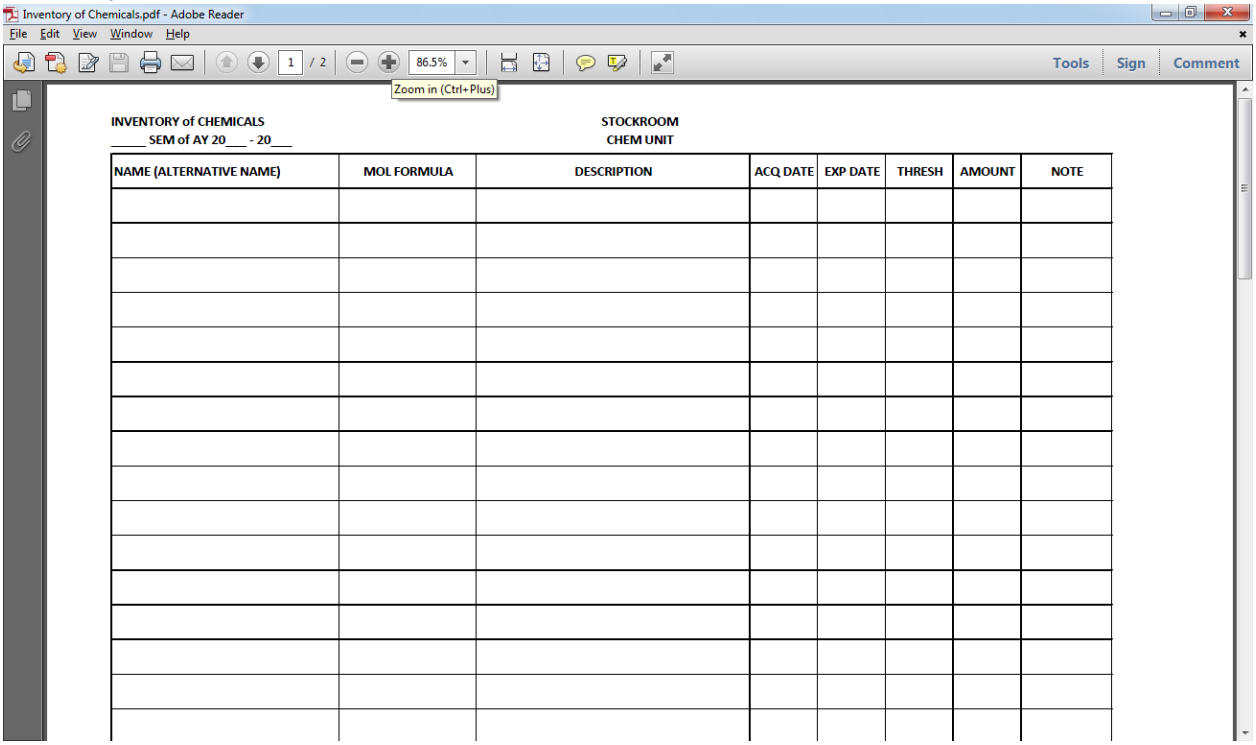

Figure 27 – Inventory of Chemicals

Inventory of Labwares

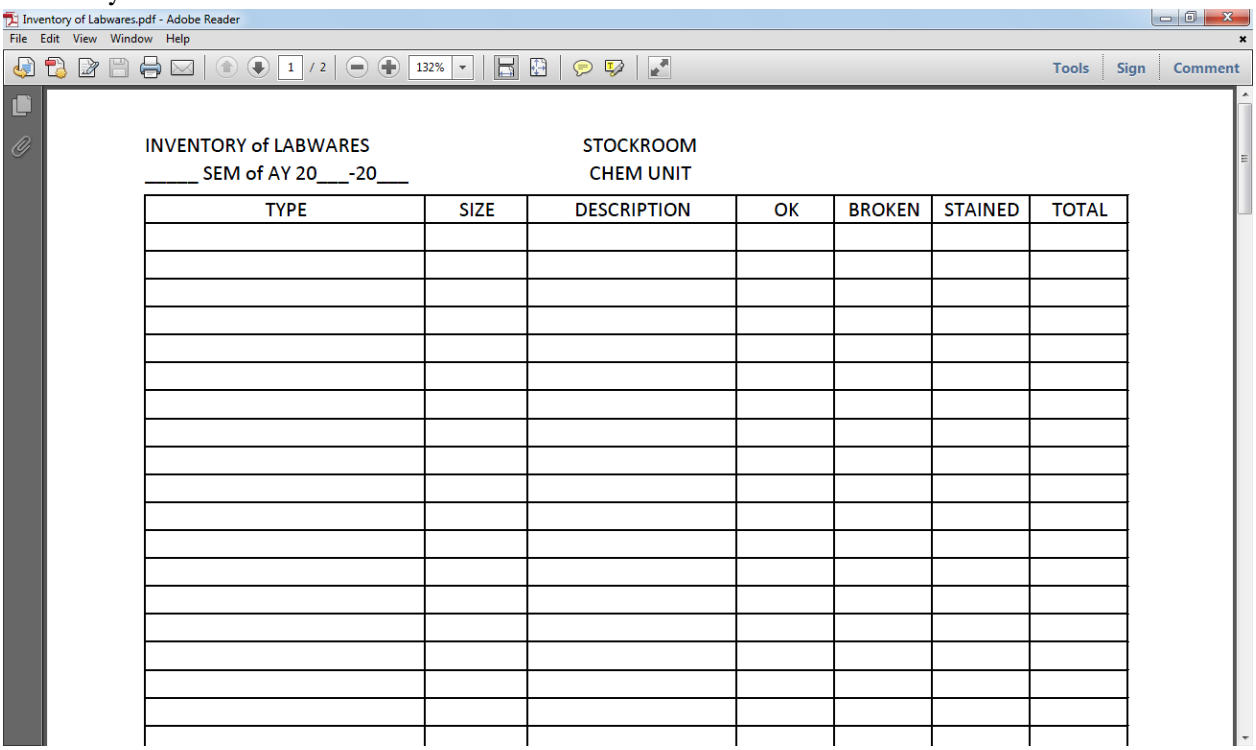

Figure 28 – Inventory of Labwares

B. Source Code

```
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.dto.AutocompleteLineItemDTO;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * AutocompleteNewLineItemController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class AutocompleteNewLineItemController {

  @Autowired
  private AccountabilityService accountabilityService;

  @RequestMapping(value = "accountability/lineitem/autocomplete", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody AutocompleteLineItemDTO autocomplete,
HttpServletRequest request) throws Exception {

    return accountabilityService.autocompleteLineItem(autocomplete);
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
```

```
 */
package edu.upm.dpsm.cims.accountability.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * ClearAccountabilityController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class ClearAccountabilityController {
  @Autowired
  private AccountabilityService accountabilityService;

  @RequestMapping(value = "accountability/clear", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody Accountability accountability, HttpServletRequest
request) throws Exception {
    Accountability toUpdateAccountability =
accountabilityService.get(accountability);
    toUpdateAccountability.setIsCleared(true);
    accountabilityService.update(toUpdateAccountability);
    return
JSONMapUtils.createJSONSuccess(EMessages.ACCOUNTABILITY_CLEAR_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.controller;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * CreateAccountabilityController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateAccountabilityController {
  @Autowired
  private AccountabilityService accountabilityService;

  @RequestMapping(value = "accountability/new", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody @Valid Accountability accountability,
HttpServletRequest request) throws Exception {
    accountabilityService.save(accountability);
    return
JSONMapUtils.createJSONSuccess(EMessages.ACCOUNTABILITY_CREATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.controller;

import java.io.File;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.model.AccountabilityLineItem;
import edu.upm.dpsm.cims.accountability.service.AccountabilityReportService;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.accountability.validation.group.ReportValidationGroup;
import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * GenerateAccountabilityReportController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GenerateAccountabilityReportController {
  @Autowired
  private AccountabilityService accountabilityService;

  @Autowired
  private AccountabilityReportService accountabilityReportService;

  @RequestMapping(value = "accountability/report/generate", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody @Validated(ReportValidationGroup.class) Accountability
accountability, HttpServletRequest request) throws Exception {
    accountability.setIsCleared(false);
    List<Accountability> accountabilityList =
accountabilityService.search(accountability);

    List<Accountability> accountabilityListDTO = DTOUtil.copy(Accountability.class,
accountabilityList, new String[] {
        Accountability.CREATED_BY, Accountability.UPDATED_BY,
Accountability.ACCOUNTABILITY_LINE_ITEMS });

    for (Accountability item : accountabilityList) {
      int index = accountabilityList.indexOf(item);
      List<AccountabilityLineItem> accountabilityLineItems =
item.getAccountabilityLineItems();
      accountabilityListDTO.get(index).setAccountabilityLineItems(
          DTOUtil.copy(AccountabilityLineItem.class, accountabilityLineItems, new
String[] {
              AccountabilityLineItem.CREATED_BY, AccountabilityLineItem.UPDATED_BY,
              AccountabilityLineItem.ACCOUNTABILITY }));
    }

    //TODO call report service
```

```
    HttpSession session = request.getSession(false);
    String filename = "report_" + session.getId() + ".pdf";
    String filePath = session.getServletContext().getRealPath(File.separator) +
ProjectConstants.TEMP_FOLDER_PATH + File.separator + filename;
    accountabilityReportService.generateReport(accountabilityListDTO, filePath);
    return new JSONSuccessMap().addProp("url", ProjectConstants.TEMP_FOLDER_PATH +
filename).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.SearchTodayAccountability;
import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.model.AccountabilityLineItem;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * GetTodayAccountabilityController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetTodayAccountabilityController {
  @Autowired
  private AccountabilityService accountabilityService;

  @RequestMapping(value = "accountability/today/search", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(HttpServletRequest request) throws Exception {
    Accountability accountability = new Accountability();
    SearchTodayAccountability searchToday = new
SearchTodayAccountability(accountability);
    accountability.setSearchModel(searchToday);
```

```
    List<Accountability> accountabilityList =
accountabilityService.search(accountability);

    List<Accountability> accountabilityListDTO = DTOUtil.copy(Accountability.class,
accountabilityList, new String[] {
        Accountability.CREATED_BY, Accountability.UPDATED_BY,
Accountability.ACCOUNTABILITY_LINE_ITEMS });

    for (Accountability item : accountabilityList) {
      int index = accountabilityList.indexOf(item);
      List<AccountabilityLineItem> accountabilityLineItems =
item.getAccountabilityLineItems();
      accountabilityListDTO.get(index).setAccountabilityLineItems(
          DTOUtil.copy(AccountabilityLineItem.class, accountabilityLineItems, new
String[] {
              AccountabilityLineItem.CREATED_BY, AccountabilityLineItem.UPDATED_BY,
              AccountabilityLineItem.ACCOUNTABILITY }));
    }

    return accountabilityListDTO;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.model.AccountabilityLineItem;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * SearchAccountabilityController
 *
 * Description:
 *
 * author : Miguelino San Miguel
```

```
 */
@Controller
public class SearchAccountabilityController {
  @Autowired
  private AccountabilityService accountabilityService;

  @RequestMapping(value = "accountability/search", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody Accountability accountability, HttpServletRequest
request) throws Exception {
    List<Accountability> accountabilityList =
accountabilityService.search(accountability);

    List<Accountability> accountabilityListDTO = DTOUtil.copy(Accountability.class,
accountabilityList, new String[] {
        Accountability.CREATED_BY, Accountability.UPDATED_BY,
Accountability.ACCOUNTABILITY_LINE_ITEMS });

    for (Accountability item : accountabilityList) {
      int index = accountabilityList.indexOf(item);
      List<AccountabilityLineItem> accountabilityLineItems =
item.getAccountabilityLineItems();
      accountabilityListDTO.get(index).setAccountabilityLineItems(
          DTOUtil.copy(AccountabilityLineItem.class, accountabilityLineItems, new
String[] {
              AccountabilityLineItem.CREATED_BY, AccountabilityLineItem.UPDATED_BY,
              AccountabilityLineItem.ACCOUNTABILITY }));
    }

    return accountabilityListDTO;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
```

```
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * UnclearAccountabilityController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class UnclearAccountabilityController {
  @Autowired
  private AccountabilityService accountabilityService;

  @RequestMapping(value = "accountability/unclear", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody Accountability accountability, HttpServletRequest
request) throws Exception {
    Accountability toUpdateAccountability =
accountabilityService.get(accountability);
    toUpdateAccountability.setIsCleared(false);
    accountabilityService.update(toUpdateAccountability);
    return
JSONMapUtils.createJSONSuccess(EMessages.ACCOUNTABILITY_UNCLEAR_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.accountability.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.accountability.dao.AccountabilityDAO;
import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * AccountabilityServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class AccountabilityDAOImpl extends JPADao<Accountability> implements
AccountabilityDAO {
```

```
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.accountability.dao;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * AccountabilityDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface AccountabilityDAO extends DAO<Accountability> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.dto;

/**
 * AutocompleteLineItemDTO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class AutocompleteLineItemDTO {

  private String laboratoryItemName;
  private Long   moduleId;

  public String getLaboratoryItemName() {
    return laboratoryItemName;
  }

  public void setLaboratoryItemName(String laboratoryItemName) {
    this.laboratoryItemName = laboratoryItemName;
  }
```

```
  /**
   * @return the moduleId
   */
  public Long getModuleId() {
    return moduleId;
  }

  /**
   * @param moduleId the moduleId to set
   */
  public void setModuleId(Long moduleId) {
    this.moduleId = moduleId;
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.accountability.model;

import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.groups.Default;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.accountability.validation.group.ReportValidationGroup;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.semester.model.AcademicYear;
import edu.upm.dpsm.cims.semester.model.Semester;

/**
 * Accountability
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "accountability")
```

```
public class Accountability extends JPAEntityModel {
  public final static String           ACCOUNTABILITY_LINE_ITEMS =
"accountabilityLineItems";
  @NotEmpty
  @Column(name = "student_name")
  private String                       studentName;
  @NotEmpty
  @Column(name = "student_number")
  private String                       studentNumber;
  @Column(name = "subject")
  private String                       subject;
  @Column(name = "section")
  private String                       section;
  @Column(name = "professor")
  private String                       professor;
  @NotNull(groups={Default.class,ReportValidationGroup.class})
  @Column(name = "semester_id")
  private Long                         semesterId;
  @ManyToOne
  @JoinColumn(name = "semester_id", updatable = false, insertable = false)
  private Semester                     semester;

  @NotNull(groups={Default.class,ReportValidationGroup.class})
  @Column(name = "academic_year_id")
  private Long                         academicYearId;

  @ManyToOne
  @JoinColumn(name = "academic_year_id", updatable = false, insertable = false)
  private AcademicYear                 academicYear;

  @NotNull
  @Column(name = "is_cleared")
  private Boolean                      isCleared;

  @NotNull
  @OneToMany(mappedBy = "accountability", cascade = { CascadeType.ALL }, fetch =
FetchType.EAGER)
  private List<AccountabilityLineItem> accountabilityLineItems;

  public String getStudentName() {
    return studentName;
  }

  public void setStudentName(String studentName) {
    this.studentName = studentName;
  }

  public String getStudentNumber() {
    return studentNumber;
  }

  public void setStudentNumber(String studentNumber) {
    this.studentNumber = studentNumber;
```

```
}

public String getSubject() {
  return subject;
}

public void setSubject(String subject) {
  this.subject = subject;
}

public String getSection() {
  return section;
}

public void setSection(String section) {
  this.section = section;
}

public String getProfessor() {
  return professor;
}

public void setProfessor(String professor) {
  this.professor = professor;
}

public Long getSemesterId() {
  return semesterId;
}

public void setSemesterId(Long semesterId) {
  this.semesterId = semesterId;
}

public Semester getSemester() {
  return semester;
}

public void setSemester(Semester semester) {
  this.semester = semester;
}

public Long getAcademicYearId() {
  return academicYearId;
}

public void setAcademicYearId(Long academicYearId) {
  this.academicYearId = academicYearId;
}

public AcademicYear getAcademicYear() {
  return academicYear;
}
```

```
  public void setAcademicYear(AcademicYear academicYear) {
    this.academicYear = academicYear;
  }

  public Boolean getIsCleared() {
    return isCleared;
  }

  public void setIsCleared(Boolean isCleared) {
    this.isCleared = isCleared;
  }

  public List<AccountabilityLineItem> getAccountabilityLineItems() {
    return accountabilityLineItems;
  }

  public void setAccountabilityLineItems(List<AccountabilityLineItem>
accountabilityLineItem) {
    this.accountabilityLineItems = accountabilityLineItem;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;

/**
 * AccountabilityLineItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "accountability_line_item")
public class AccountabilityLineItem extends JPAEntityModel{
  public final static String ACCOUNTABILITY ="accountability";
  @ManyToOne(fetch = FetchType.EAGER)
```

```
  @JoinColumn(name = "accountability_id",referencedColumnName = "id",insertable=true,
updatable=true)
  private Accountability accountability;

  @Column(name = "definition_id")
  private Long definitionId;

  @Column(name = "item_id")
  private Long           itemId;

  @Column(name = "item_name")
  private String itemName;

  @Column(name = "module_id")
  private Long           moduleId;

  @Column(name = "quantity")
  private Integer        quantity;

  @Column(name = "is_consumable")
  private Boolean        isConsumable;

  public String getItemName() {
    return itemName;
  }

  public void setItemName(String itemName) {
    this.itemName = itemName;
  }

  public Boolean getIsConsumable() {
    return isConsumable;
  }

  public Accountability getAccountability() {
    return accountability;
  }

  public void setAccountability(Accountability accountability) {
    this.accountability = accountability;
  }

  public Long getItemId() {
    return itemId;
  }

  public void setItemId(Long itemId) {
    this.itemId = itemId;
  }

  public Long getModuleId() {
    return moduleId;
  }
```

```
  public void setModuleId(Long moduleId) {
    this.moduleId = moduleId;
  }

  public Integer getQuantity() {
    return quantity;
  }

  public void setQuantity(Integer quantity) {
    this.quantity = quantity;
  }

  public void setIsConsumable(Boolean isConsumable) {
    this.isConsumable = isConsumable;
  }

  public Long getDefinitionId() {
    return definitionId;
  }

  public void setDefinitionId(Long definitionId) {
    this.definitionId = definitionId;
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 10, 2014
 */
package edu.upm.dpsm.cims.accountability.service.impl;

import java.sql.Timestamp;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.accountability.dao.AccountabilityDAO;
import edu.upm.dpsm.cims.accountability.dto.AutocompleteLineItemDTO;
import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.model.AccountabilityLineItem;
import edu.upm.dpsm.cims.accountability.service.AccountabilityService;
import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
```

```
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.exception.LIMSException;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.core.util.DateUtil;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.core.util.UserAccountUtil;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;
import edu.upm.dpsm.cims.mathcs.labware.definition.services.MathCSLabwareService;
import edu.upm.dpsm.cims.modules.model.EModules;
import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;

/**
 * AccountabilityServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class AccountabilityServiceImpl extends JPAService<Accountability> implements
AccountabilityService {

  @Autowired
  private AccountabilityDAO        dao;
  @Autowired
  private ChemistryChemicalService chemistryChemicalService;
  @Autowired
  private ChemistryLabwareService  chemistryLabwareService;

  @Autowired
  private BiologyChemicalService   biologyChemicalService;
  @Autowired
  private BiologyLabwareService    biologyLabwareService;

  @Autowired
  private PhysicsChemicalService   physicsChemicalService;
  @Autowired
  private PhysicsLabwareService    physicsLabwareService;

  @Autowired
  private MathCSLabwareService     mathcsLabwareService;
```

```
  @Override
  public DAO<Accountability> getDAO() {

    return dao;
  }

  @Override
  public Accountability save(Accountability accountability) throws Exception {
    if (accountability != null) {

      Long userId = UserAccountUtil.getUserAccountId();
      Timestamp currentTime = DateUtil.getCurrentTime();
      List<AccountabilityLineItem> lineItems =
accountability.getAccountabilityLineItems();
      if (!ObjectUtils.isEmpty(lineItems)) {
        for (AccountabilityLineItem item : lineItems) {
          item.setCreatedById(userId);
          item.setUpdatedById(userId);
          item.setCreatedDate(currentTime);
          item.setUpdatedDate(currentTime);
          item.setAccountability(accountability);
          consumeQuantity(item);
        }
      }
    }
    return super.save(accountability);
  }

  protected void consumeQuantity(AccountabilityLineItem lineItem) throws Exception {
    EModules module = EModules.valueOf( lineItem.getModuleId());
    Long totalQuantity = 0L;
    switch (module) {
      case BIOLOGY_CHEMICAL:
        BiologyChemical biologyChemical = new BiologyChemical();
        biologyChemical.setId(lineItem.getDefinitionId());
        biologyChemical = biologyChemicalService.get(biologyChemical);
        totalQuantity = biologyChemical.getTotalQuantity();
        if( totalQuantity == 0){
          throw new LIMSException(EValidationMessages.CHEMICAL_OUT_OF_QUANTITY);
        }else if(totalQuantity < lineItem.getQuantity()){
          throw new LIMSException(EValidationMessages.CHEMICAL_LACKING);
        }

        biologyChemical.setTotalQuantity(biologyChemical.getTotalQuantity() -
lineItem.getQuantity());
        biologyChemicalService.update(biologyChemical);
        break;
      case PHYSICS_CHEMICAL:
        PhysicsChemical physicsChemical = new PhysicsChemical();
        physicsChemical.setId(lineItem.getDefinitionId());
        physicsChemical = physicsChemicalService.get(physicsChemical);
        totalQuantity = physicsChemical.getTotalQuantity();
```

```
        if( totalQuantity == 0){
          throw new LIMSException(EValidationMessages.CHEMICAL_OUT_OF_QUANTITY);
        }else if(totalQuantity < lineItem.getQuantity()){
          throw new LIMSException(EValidationMessages.CHEMICAL_LACKING);
        }
        physicsChemical.setTotalQuantity(physicsChemical.getTotalQuantity() -
lineItem.getQuantity());
        physicsChemicalService.update(physicsChemical);
        break;
      case CHEMISTRY_CHEMICAL:
        ChemistryChemical chemistryChemical = new ChemistryChemical();
        chemistryChemical.setId(lineItem.getDefinitionId());
        chemistryChemical = chemistryChemicalService.get(chemistryChemical);
        totalQuantity = chemistryChemical.getTotalQuantity();
        if( totalQuantity == 0){
          throw new LIMSException(EValidationMessages.CHEMICAL_OUT_OF_QUANTITY);
        }else if(totalQuantity < lineItem.getQuantity()){
          throw new LIMSException(EValidationMessages.CHEMICAL_LACKING);
        }
        chemistryChemical.setTotalQuantity(chemistryChemical.getTotalQuantity() -
lineItem.getQuantity());
        chemistryChemicalService.update(chemistryChemical);
        break;
      default:
        ;
    }
  }

  @Override
  public List<?> autocompleteLineItem(AutocompleteLineItemDTO dto) throws Exception {
    EModules module = EModules.valueOf(dto.getModuleId());
    switch (module) {
      case BIOLOGY_CHEMICAL:
        BiologyChemical biologyChemical = new BiologyChemical();
        biologyChemical.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(BiologyChemical.class,
biologyChemicalService.search(biologyChemical), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
BiologyChemical.BIOLOGY_CHEMICAL_ITEMS });
      case BIOLOGY_LABWARE:
        BiologyLabware biologyLabware = new BiologyLabware();
        biologyLabware.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(BiologyLabware.class,
biologyLabwareService.search(biologyLabware), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
BiologyLabware.BIOLOGY_LABWARE_ITEMS });
      case PHYSICS_CHEMICAL:
        PhysicsChemical physicsChemical = new PhysicsChemical();
        physicsChemical.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(PhysicsChemical.class,
physicsChemicalService.search(physicsChemical), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
PhysicsChemical.PHYSICS_CHEMICAL_ITEMS });
```

```
      case PHYSICS_LABWARE:
        PhysicsLabware physicsLabware = new PhysicsLabware();
        physicsLabware.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(PhysicsLabware.class,
physicsLabwareService.search(physicsLabware), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
PhysicsLabware.PHYSICS_LABWARE_ITEMS });
      case CHEMISTRY_CHEMICAL:
        ChemistryChemical chemistryChemical = new ChemistryChemical();
        chemistryChemical.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(ChemistryChemical.class,
chemistryChemicalService.search(chemistryChemical), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
ChemistryChemical.CHEMISTRY_CHEMICAL_ITEMS });
      case CHEMISTRY_LABWARE:
        ChemistryLabware chemistryLabware = new ChemistryLabware();
        chemistryLabware.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(ChemistryLabware.class,
chemistryLabwareService.search(chemistryLabware), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
ChemistryLabware.CHEMISTRY_LABWARE_ITEMS });
      case MATH_CS_MACHINE:
        MathCSLabware mathcsLabware = new MathCSLabware();
        mathcsLabware.setName(dto.getLaboratoryItemName());
        return DTOUtil.copy(MathCSLabware.class,
mathcsLabwareService.search(mathcsLabware), new String[] {
            JPAEntityModel.CREATED_BY, JPAEntityModel.UPDATED_BY,
MathCSLabware.MATH_CS_LABWARE_ITEMS });
      default:
        ;
    }
    return null;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.service;

import static net.sf.dynamicreports.report.builder.DynamicReports.col;
import static net.sf.dynamicreports.report.builder.DynamicReports.export;
import static net.sf.dynamicreports.report.builder.DynamicReports.grp;
import static net.sf.dynamicreports.report.builder.DynamicReports.report;
import static net.sf.dynamicreports.report.builder.DynamicReports.type;
import static net.sf.dynamicreports.report.builder.DynamicReports.cmp;

import java.util.Date;
import java.util.List;
```

```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.jasper.builder.export.JasperPdfExporterBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

import org.springframework.stereotype.Service;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.accountability.model.AccountabilityLineItem;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * AccountabilityReportService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
public class AccountabilityReportService {
  private final static String ID                 = "id";
  private final static String ID_LABEL           = "ID";
  private final static String DATE_FORMAT        = "mm/dd/yyyy";

  private final static String CREATED_DATE       = "createdDate";
  private final static String CREATED_DATE_LABEL = "Created Date";
  private final static String ITEM_NAME          = "itemName";
  private final static String ITEM_NAME_LABEL    = "Item";
  private final static String QUANTITY           = "quantity";
  private final static String QUANTITY_LABEL     = "Quantity";

  TextColumnBuilder<String>   idColumn           = col.column(ID_LABEL, ID,
type.stringType());
  TextColumnBuilder<Date>     createdDateColumn  = col.column(CREATED_DATE_LABEL +
"(" + DATE_FORMAT + ")",
                                                     CREATED_DATE, type.dateType());
  TextColumnBuilder<String>   itemNameColumn     = col.column(ITEM_NAME_LABEL,
ITEM_NAME, type.stringType());
  TextColumnBuilder<Integer>  quantityColumn     = col.column(QUANTITY_LABEL,
QUANTITY, type.integerType());

  public void generateReport(List<Accountability> accountabilityList, String
filename) {
    JasperPdfExporterBuilder pdfExporter = export.pdfExporter(filename);
    ColumnGroupBuilder itemGroup =
grp.group(idColumn).setTitleWidth(30).setHeaderLayout(GroupHeaderLayout.VALUE)
        .showColumnHeaderAndFooter();
    try {
```

```
      JasperReportBuilder reportBuilder =
report().setTemplate(Templates.reportTemplate).setShowColumnTitle(false)
          .columns(itemNameColumn, quantityColumn,
createdDateColumn).groupBy(itemGroup)
          .title(Templates.createTitleComponent("Accountability
Report")).pageFooter(Templates.footerComponent);
          if(!ObjectUtils.isEmpty(accountabilityList)){
            reportBuilder.setDataSource(createDataSource(accountabilityList));
          }else{
            reportBuilder.noData(Templates.createTitleComponent("Accountability
Report"), cmp.text("There is no data"));
          }
      reportBuilder.toPdf(pdfExporter);
    } catch (DRException e) {
      e.printStackTrace();
    }
  }

  private JRDataSource createDataSource(List<Accountability> accountabilityList) {
    DRDataSource dataSource = new DRDataSource(ID, ITEM_NAME, QUANTITY,
CREATED_DATE);
    if (!ObjectUtils.isEmpty(accountabilityList)) {
      for (Accountability accountability : accountabilityList) {
        extractRow(dataSource, accountability);
      }
    }

    return dataSource;
  }

  /**
   * @param dataSource
   * @param accountability
   */
  private void extractRow(DRDataSource dataSource, Accountability accountability) {
    String groupId = accountability.getStudentNumber() + " - " +
accountability.getStudentName();
    List<AccountabilityLineItem> lineItems =
accountability.getAccountabilityLineItems();
    for (AccountabilityLineItem lineItem : lineItems) {
      dataSource.add(groupId, lineItem.getItemName(), lineItem.getQuantity(),
lineItem.getCreatedDate());
    }
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 10, 2014
 */
```

```
package edu.upm.dpsm.cims.accountability.service;

import java.util.List;

import edu.upm.dpsm.cims.accountability.dto.AutocompleteLineItemDTO;
import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * AccountabilityService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface AccountabilityService extends BusinessService<Accountability> {
  public List<?> autocompleteLineItem(AutocompleteLineItemDTO dto) throws Exception;
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability.validation.group;

/**
 * ReportValidationGroup
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface ReportValidationGroup {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.accountability;

import edu.upm.dpsm.cims.accountability.model.Accountability;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.util.DateUtil;
import static edu.upm.dpsm.cims.accountability.model.Accountability.*;
/**
```

```
 * SearchTodayAccountability
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchTodayAccountability extends JPASearchModel<Accountability> {

  public SearchTodayAccountability(Accountability model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(Accountability model) throws Exception {
    setSearch(new SearchCondition()
        .add(CREATED_DATE,ERestrictionType.GE,DateUtil.getTodayStartTime())
        .add(CREATED_DATE,ERestrictionType.LE,DateUtil.getTodayEndTime()));
  }



}
package edu.upm.dpsm.cims.biology.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddBiologyChemicalController {
  @Autowired
  public BiologyChemicalService biologyChemicalService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }
```

```
  @RequestMapping(value = "biology/chemical/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated BiologyChemical biologyChemical,
HttpServletRequest request)
      throws Exception {

    biologyChemicalService.save(biologyChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_DEFINITION_CREATE_SUCCESS).getMap()
;
  }



}
package edu.upm.dpsm.cims.biology.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class GetBiologyChemicalController {

  @Autowired
  private BiologyChemicalService biologyChemicalService;

  @RequestMapping(value = "biology/chemical/get", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody BiologyChemical biologyChemical,
HttpServletRequest request)
      throws Exception {
    BiologyChemical chemicalItem = biologyChemicalService.get(biologyChemical);

    return DTOUtil.copy(BiologyChemical.class, chemicalItem, new String[] {
BiologyChemical.CREATED_BY,
        BiologyChemical.UPDATED_BY, BiologyChemical.BIOLOGY_CHEMICAL_ITEMS });
  }
```

```
}
package edu.upm.dpsm.cims.biology.chemical.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchBiologyChemicalController {

  @Autowired
  private BiologyChemicalService biologyChemicalService;

  @RequestMapping(value = "biology/chemical/search", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody BiologyChemical biologyChemical,
HttpServletRequest request)
      throws Exception {
    List<BiologyChemical> chemicalList =
biologyChemicalService.search(biologyChemical);

    return DTOUtil.copy(BiologyChemical.class, chemicalList, new String[] {
BiologyChemical.CREATED_BY,
        BiologyChemical.UPDATED_BY, BiologyChemical.BIOLOGY_CHEMICAL_ITEMS });
  }

}
package edu.upm.dpsm.cims.biology.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import
edu.upm.dpsm.cims.biology.chemical.definition.validation.group.UpdateBiologyChemical;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class UpdateBiologyChemicalController {
  @Autowired
  public BiologyChemicalService biologyChemicalService;

  @RequestMapping(value = "biology/chemical/update-definition", method =
RequestMethod.POST)
  public @ResponseBody
  Object updateChemical(@RequestBody @Validated(UpdateBiologyChemical.class)
BiologyChemical biologyChemical,
      HttpServletRequest request) throws Exception {

    BiologyChemical ccd = new BiologyChemical();
    ccd.setId(biologyChemical.getId());
    ccd = biologyChemicalService.get(ccd);
    BeanUtils.copyProperties(biologyChemical, ccd, new String[] { BiologyChemical.ID,
BiologyChemical.TOTAL_QUANTITY,
        BiologyChemical.CREATED_BY_ID, BiologyChemical.CREATED_DATE });
    biologyChemicalService.update(ccd);
    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_DEFINITION_UPDATE_SUCCESS).getMap()
;
  }
}
package edu.upm.dpsm.cims.biology.chemical.definition.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.FetchMode;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;

import
edu.upm.dpsm.cims.biology.chemical.definition.dao.BiologyChemicalDefinitionDAO;
import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * @author Miguelino San Miguel
 */
@Repository
```

```
public class BiologyChemicalDefinitionDAOImpl extends JPADao<BiologyChemical>
implements BiologyChemicalDefinitionDAO {

  @Override
  public List<BiologyChemical> search(
      BiologyChemical chemical) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session.createCriteria(
        BiologyChemical.class).setFetchMode(
        "createdBy",
        FetchMode.JOIN);
    return super.search(chemical, criteria);
  }
}
package edu.upm.dpsm.cims.biology.chemical.definition.dao;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * @author Miguelino San Miguel
 */
public interface BiologyChemicalDefinitionDAO extends DAO<BiologyChemical> {
}
package edu.upm.dpsm.cims.biology.chemical.definition.models;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.validation.groups.Default;

import org.hibernate.validator.constraints.NotEmpty;

import
edu.upm.dpsm.cims.biology.chemical.definition.validation.group.UpdateBiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;

@Entity
@Table(name = "biology_chemical_definition")
```

```
@Access(AccessType.FIELD)
public class BiologyChemical extends JPAEntityModel implements Serializable,
MonitorItem {

  public final static String          NAME                      = "name";
  public final static String          MOL_FORMULA               = "molFormula";
  public final static String          COLOR                     = "color";
  public final static String          TEXTURE                   = "texture";
  public final static String          GRADE                     = "grade";
  public final static String          HYGROSCOPIC               = "hygroscopic";
  public final static String          EXPLOSIVE                 = "explosive";
  public final static String          TOTAL_QUANTITY            = "totalQuantity";
  public final static String          THRESH                    = "thresh";
  public final static String          MINIMUM                   = "minimum";
  public final static String          BIOLOGY_CHEMICAL_ITEMS =
"biologyChemicalItems";

  private static final long           serialVersionUID         =
3529801466283219405L;

  @NotEmpty
  @Size(min = 2, max = 50)
  @Column(updatable = true, name = "name")
  private String                      name;

  @Size(min = 2, max = 100, groups = { Default.class, UpdateBiologyChemical.class })
  @Column(updatable = true, name = "mol_formula")
  private String                      molFormula;

  @Column(updatable = true, name = "color")
  private String                      color;

  @Column(updatable = true, name = "texture")
  private String                      texture;

  @Column(updatable = true, name = "grade")
  private String                      grade;

  @Column(updatable = true, name = "hygroscopic")
  private Boolean                     hygroscopic;

  @Column(updatable = true, name = "explosive")
  private Boolean                     explosive;

  @Column(updatable = true, name = "total_quantity")
  private Long                        totalQuantity            = Long.valueOf(0L);

  @Column(updatable = true, name = "thresh")
  private Integer                     thresh;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
```

```
  private Integer                     minimum;

  @OneToMany(mappedBy = "biologyChemical", cascade = { CascadeType.REMOVE },fetch =
FetchType.LAZY)
  private List<BiologyChemicalItem> biologyChemicalItems;

  @Transient
  private String                      username;

  public Integer getMinimum() {
    return minimum;
  }

  public void setMinimum(Integer minimum) {
    this.minimum = minimum;
  }

  public Boolean getHygroscopic() {
    return hygroscopic;
  }

  public void setHygroscopic(Boolean hygroscopic) {
    this.hygroscopic = hygroscopic;
  }

  public Boolean getExplosive() {
    return explosive;
  }

  public void setExplosive(Boolean explosive) {
    this.explosive = explosive;
  }

  public String getColor() {
    return color;
  }

  public void setColor(String color) {
    this.color = color;
  }

  public String getGrade() {
    return grade;
  }

  public void setGrade(String grade) {
    this.grade = grade;
  }

  public String getMolFormula() {
    return molFormula;
  }
```

```
  public void setMolFormula(String molFormula) {
    this.molFormula = molFormula;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getTexture() {
    return texture;
  }

  public void setTexture(String texture) {
    this.texture = texture;
  }

  public Integer getThresh() {
    return thresh;
  }

  public void setThresh(Integer thresh) {
    this.thresh = thresh;
  }

  public Long getTotalQuantity() {
    return totalQuantity;
  }

  public void setTotalQuantity(Long totalQuantity) {
    this.totalQuantity = totalQuantity;
  }

  public List<BiologyChemicalItem> getBiologyChemicalItems() {
    return biologyChemicalItems;
  }

  public void setBiologyChemicalItems(List<BiologyChemicalItem> biologyChemicalItems)
{
    this.biologyChemicalItems = biologyChemicalItems;
  }

  public String getUsername() {
    return username;
  }

  public void setUsername(String username) {
    this.username = username;
  }
}
```

```
package edu.upm.dpsm.cims.biology.chemical.definition.search;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import static edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical.*;
/**
 * @author Miguelino San Miguel
 */
public class SearchBiologyChemicals extends JPASearchModel<BiologyChemical> {

  /**
   * @param request
   * @param model
   * @throws Exception
   */
  public SearchBiologyChemicals(
      BiologyChemical model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(
      BiologyChemical model)
      throws Exception {
    setSearch(new SearchCondition().add(SearchCondition
        .conjunction()
        .add(
            SearchCondition
                .disjunction()
                .add(NAME, ERestrictionType.LIKE, model.getName(),
MatchMode.ANYWHERE)
                .add(MOL_FORMULA, ERestrictionType.LIKE, model.getMolFormula(),
MatchMode.ANYWHERE))
        .and(CREATED_BY_ID, ERestrictionType.EQ, model.getCreatedById())
        .and(ID, ERestrictionType.EQ, model.getId())
         )
     );
  }
}
package edu.upm.dpsm.cims.biology.chemical.definition.services.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import
edu.upm.dpsm.cims.biology.chemical.definition.dao.BiologyChemicalDefinitionDAO;
```

```
import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.search.SearchBiologyChemicals;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorSupply;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional(rollbackFor = Exception.class)
public class BiologyChemicalServiceImpl extends JPAService<BiologyChemical>
implements
    BiologyChemicalService {

  @Override
  public BiologyChemical update(BiologyChemical t) throws Exception {
    return super.update(t);
  }

  @Override
  public BiologyChemical save(BiologyChemical t) throws Exception {

    return super.save(t);
  }

  @Autowired
  private BiologyChemicalDefinitionDAO dao;

  @Transactional(readOnly = true)
  @Override
  public List<BiologyChemical> search(BiologyChemical chemical) throws Exception {
    Pagination pagination = chemical.getPagination();
    pagination = (pagination == null) ? (new Pagination()) : pagination;
    pagination.setTotalSize(dao.getTotalSize(chemical));
    pagination.setEnable(true);
    chemical.setPagination(pagination);
    SearchBiologyChemicals scc = new SearchBiologyChemicals(chemical);
    chemical.setSearchModel(scc);
    return dao.search(chemical);
  }

  @Override
  public DAO<BiologyChemical> getDAO() {
    return this.dao;
  }

}
package edu.upm.dpsm.cims.biology.chemical.definition.services;

import org.springframework.transaction.annotation.Transactional;
```

```
import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface BiologyChemicalService extends BusinessService<BiologyChemical> {

}
package edu.upm.dpsm.cims.biology.chemical.definition.validation.group;

/**
 * @author Miguelino San Miguel
 */
public interface UpdateBiologyChemical {

}
package edu.upm.dpsm.cims.biology.chemical.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.biology.chemical.item.services.BiologyChemicalItemService;
import
edu.upm.dpsm.cims.biology.chemical.item.validation.validator.ExistingBiologyChemicalD
efinitionValidator;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddBiologyChemicalItemController {
  @Autowired
  private BiologyChemicalItemService biologyChemicalDefinitionService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
    binder.addValidators(new ExistingBiologyChemicalDefinitionValidator());
  }
```

```
  @RequestMapping(value = "biology/chemical/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemicalItem(@RequestBody @Validated BiologyChemicalItem
biologyChemicalItem,
      HttpServletRequest request) throws Exception {

    biologyChemicalDefinitionService.saveItems(biologyChemicalItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_ITEM_ADD_SUCCESS).getMap();
  }


}
```

```
package edu.upm.dpsm.cims.biology.chemical.item.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.FetchMode;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;

import edu.upm.dpsm.cims.biology.chemical.item.dao.BiologyChemicalItemDAO;
import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * @author Miguelino San Miguel
 */
@Repository
public class BiologyChemicalItemDAOImpl extends JPADao<BiologyChemicalItem>implements
BiologyChemicalItemDAO {

  @Override
  public List<BiologyChemicalItem> search(BiologyChemicalItem chemicalItem)
      throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session
        .createCriteria(BiologyChemicalItem.class)
        .setFetchMode(
            "createdBy",
            FetchMode.JOIN)
        .createAlias(BiologyChemicalItem.BIOLOGY_CHEMICAL_DEFINITION,
            BiologyChemicalItem.BIOLOGY_CHEMICAL_DEFINITION)
        .setFetchMode(BiologyChemicalItem.BIOLOGY_CHEMICAL_DEFINITION,
            FetchMode.JOIN);

    // apply search parameters e.g. pagination
    return super.search(chemicalItem, criteria);
  }
```

```
}
package edu.upm.dpsm.cims.biology.chemical.item.dao;

import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * @author Miguelino San Miguel
 */
public interface BiologyChemicalItemDAO extends DAO<BiologyChemicalItem> {

}
package edu.upm.dpsm.cims.biology.chemical.item.models;

import java.io.Serializable;
import java.sql.Timestamp;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.notetype.models.NoteType;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "biology_chemical_item")
@Access(AccessType.FIELD)
public class BiologyChemicalItem extends JPAEntityModel implements Serializable {

  private static final long  serialVersionUID           = -392215716780171378L;

  public final static String CHEMICAL_ID                = "chemicalId";
  public final static String CHEMICAL_NAME              = "chemicalName";
  public final static String STATUS                     = "consumed";
  public final static String EXPIRATION_DATE            = "expDate";
  public final static String NOTE_TYPE_ID               = "noteTypeId";
  public final static String NOTE_TYPE                  = "noteType";
  public static final String BIOLOGY_CHEMICAL_DEFINITION = "biologyChemical";
```

```
@NotNull
@Column(updatable = false, name = "chemical_id")
private Long               chemicalId;

@NotEmpty
@Transient
private String             chemicalName;

@Column(updatable = true, name = "consumed")
private Boolean            consumed;

@Column(name = "note_type")
private Long               noteTypeId;

@ManyToOne
@JoinColumn(name = "note_type", updatable = false, insertable = false)
private NoteType           noteType;

@Column(updatable = true, name = "exp_date")
private Timestamp          expDate;

@Min(value = 1)
@Transient
private Integer            quantity                    = 1;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "chemical_id", updatable = false, insertable = false)
private BiologyChemical    biologyChemical;

public Long getChemicalId() {
  return chemicalId;
}

public void setChemicalId(Long chemicalId) {
  this.chemicalId = chemicalId;
}

public Boolean getConsumed() {
  return consumed;
}

public void setConsumed(Boolean consumed) {
  this.consumed = consumed;
}

public Long getNoteTypeId() {
  return noteTypeId;
}

public void setNoteTypeId(Long noteType) {
  this.noteTypeId = noteType;
}
```

```
  public NoteType getNoteType() {
    return noteType;
  }

  public void setNoteType(NoteType noteType) {
    this.noteType = noteType;
  }

  public Timestamp getExpDate() {
    return expDate;
  }

  /**
   * @param expDate
   *          the expDate to set
   */
  public void setExpDate(Timestamp expDate) {
    this.expDate = expDate;
  }

  public BiologyChemical getBiologyChemicalDefinition() {
    return biologyChemical;
  }

  public void setBiologyChemicalDefinition(BiologyChemical biologyChemical) {
    this.biologyChemical = biologyChemical;
  }

  public String getChemicalName() {
    return chemicalName;
  }

  public void setChemicalName(String chemicalName) {
    this.chemicalName = chemicalName;
  }

  public Integer getQuantity() {
    return quantity;
  }

  public void setQuantity(Integer quantity) {
    this.quantity = quantity;
  }
}
package edu.upm.dpsm.cims.biology.chemical.item.search;

import javax.servlet.http.HttpServletRequest;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
```

```
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;

/**
 * @author Miguelino San Miguel
 */
public class SearchChemicalItems extends JPASearchModel<BiologyChemicalItem> {

  /**
   * @param request
   * @param model
   * @throws Exception
   */
  public SearchChemicalItems(BiologyChemicalItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(BiologyChemicalItem model) throws Exception {

    setSearch(new SearchCondition()
        .add(BiologyChemicalItem.BIOLOGY_CHEMICAL_DEFINITION + "." +
BiologyChemical.NAME,
            ERestrictionType.LIKE, model.getChemicalName(), MatchMode.START)
        .add(BiologyChemicalItem.EXPIRATION_DATE, ERestrictionType.LE,
model.getExpDate())
        .add(BiologyChemicalItem.CREATED_BY_ID, ERestrictionType.EQ,
model.getCreatedById())
        .add(BiologyChemicalItem.NOTE_TYPE_ID, ERestrictionType.EQ,
model.getNoteTypeId()));
  }
}
package edu.upm.dpsm.cims.biology.chemical.item.services.impl;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.biology.chemical.item.dao.BiologyChemicalItemDAO;
import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.biology.chemical.item.search.SearchChemicalItems;
import edu.upm.dpsm.cims.biology.chemical.item.services.BiologyChemicalItemService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;

/**
 * @author Miguelino San Miguel
```

```
 */
@Service
@Transactional
public class BiologyChemicalItemServiceImpl extends JPAService<BiologyChemicalItem>
implements
    BiologyChemicalItemService {

  @Autowired
  private BiologyChemicalItemDAO           dao;

  @Autowired
  private BiologyChemicalService biologyChemicalService;

  @Override
  public List<BiologyChemicalItem> search(BiologyChemicalItem chemicalItem) throws
Exception {
    Pagination pagination = chemicalItem.getPagination();
    if (pagination == null) {
      pagination = new Pagination();
      chemicalItem.setPagination(pagination);
    }

    SearchChemicalItems scc = new SearchChemicalItems(

    chemicalItem);
    chemicalItem.setSearchModel(scc);
    pagination.setTotalSize(dao.getTotalSize(chemicalItem));
    pagination.setEnable(true);
    return this.dao.search(chemicalItem);
  }

  @Override
  public DAO<BiologyChemicalItem> getDAO() {
    return this.dao;
  }

  @Override
  public BiologyChemicalItem saveItems(BiologyChemicalItem biologyChemicalItem)
throws Exception {
    int quantity = biologyChemicalItem.getQuantity();
    biologyChemicalItem.setConsumed(true);
    int row = quantity;
    while (row > 0) {
      BiologyChemicalItem item = new BiologyChemicalItem();
      PropertyUtils.copyProperties(item, biologyChemicalItem);
      super.save(item);
      row--;
    }

    // update parent ChemicalDefinition total quantity
    BiologyChemical ccd = new BiologyChemical();
    ccd.setId(biologyChemicalItem.getChemicalId());
    ccd = biologyChemicalService.get(ccd);
```

```
    ccd.setTotalQuantity(ccd.getTotalQuantity() + quantity);
    biologyChemicalService.update(ccd);

    return biologyChemicalItem;
  }

  @Override
  public BiologyChemicalItem save(BiologyChemicalItem t) throws Exception {
    return super.save(t);
  }
}
package edu.upm.dpsm.cims.biology.chemical.item.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface BiologyChemicalItemService extends
BusinessService<BiologyChemicalItem> {
  public  BiologyChemicalItem saveItems(
      BiologyChemicalItem biologyChemicalItem) throws Exception;
}
package edu.upm.dpsm.cims.biology.chemical.item.validation.group;

/**
 * @author Miguelino San Miguel
 *
 */
public interface UpdateChemicalItem {

}
package edu.upm.dpsm.cims.biology.chemical.item.validation.validator;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.biology.chemical.definition.models.BiologyChemical;
import edu.upm.dpsm.cims.biology.chemical.definition.services.BiologyChemicalService;
import edu.upm.dpsm.cims.biology.chemical.item.models.BiologyChemicalItem;
import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
```

```
 * @author Miguelino San Miguel
 */
@Component
public class ExistingBiologyChemicalDefinitionValidator implements Validator {
      @Autowired
      private MessageSource messageSource;

      @Autowired
      private BiologyChemicalService biologyChemicalService;
  @Override
  public boolean supports(Class<?> clazz) {
    return BiologyChemicalItem.class.equals(clazz);
  }

  @Override
  public void validate(Object target, Errors errors) {
    try {
      BiologyChemicalItem biologyChemicalItem = (BiologyChemicalItem) target;
      Long chemicalId = biologyChemicalItem.getChemicalId();
      if (!ObjectUtils.isEmpty(chemicalId)) {
        BindingResult result = (BindingResult) errors;
        BiologyChemical existingDef = new BiologyChemical();
        existingDef.setId(chemicalId);
        existingDef = biologyChemicalService.get(existingDef);
        if (existingDef == null) {
          ValidationUtils.addBindingError(result,

BiologyChemicalItem.CHEMICAL_NAME,EValidationMessages.NON_EXISTING_CHEMICAL_NAME);
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

}
package edu.upm.dpsm.cims.biology.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
```

```
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddBiologyLabwareController {
  @Autowired
  private BiologyLabwareService biologyLabwareService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "biology/labware/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated BiologyLabware biologyChemical,
HttpServletRequest request)
      throws Exception {

    biologyLabwareService.save(biologyChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_CREATE_SUCCESS).getMap();
  }
}
package edu.upm.dpsm.cims.biology.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class GetBiologyLabwareController {

  @Autowired
  private BiologyLabwareService biologyLabwareService;

  @RequestMapping(value = "biology/labware/get", method = RequestMethod.POST)
  public @ResponseBody
```

```
  Object chemicalListPage(@RequestBody BiologyLabware biologyLabware,
HttpServletRequest request) throws Exception {
    BiologyLabware labwareItem = biologyLabwareService.get(biologyLabware);

    return DTOUtil.copy(BiologyLabware.class, labwareItem, new String[] {
BiologyLabware.CREATED_BY,
        BiologyLabware.UPDATED_BY, BiologyLabware.BIOLOGY_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.biology.labware.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchBiologyLabwareController {

  @Autowired
  private BiologyLabwareService biologyLabwareService;

  @RequestMapping(value = "biology/labware/search", method = RequestMethod.POST)
  public @ResponseBody
  Object labwareListPage(@RequestBody BiologyLabware biologyLabware,
HttpServletRequest request)
      throws Exception {
    List<BiologyLabware> chemicalList = biologyLabwareService.search(biologyLabware);

    return DTOUtil.copy(BiologyLabware.class, chemicalList, new String[] {
BiologyLabware.CREATED_BY,
        BiologyLabware.UPDATED_BY, BiologyLabware.BIOLOGY_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.biology.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class UpdateBiologyLabwareController {
  @Autowired
  public BiologyLabwareService biologyLabwareService;

  @RequestMapping(value = "biology/labware/update-definition", method =
RequestMethod.POST)
  public @ResponseBody
  Object updateChemical(@RequestBody @Validated BiologyLabware biologyLabware,
HttpServletRequest request)
      throws Exception {

    BiologyLabware ccd = new BiologyLabware();
    ccd.setId(biologyLabware.getId());
    ccd = biologyLabwareService.get(ccd);
    BeanUtils.copyProperties(biologyLabware, ccd, new String[] { BiologyLabware.ID,
BiologyLabware.TOTAL_QUANTITY,
        BiologyLabware.OK_QUANTITY, BiologyLabware.CREATED_BY_ID,
BiologyLabware.CREATED_DATE });
    biologyLabwareService.update(ccd);
    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_UPDATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.biology.labware.definition.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
```

```
import edu.upm.dpsm.cims.biology.labware.definition.dao.BiologyLabwareDAO;
import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * BiologyLabwareDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class BiologyLabwareDAOImpl extends JPADao<BiologyLabware> implements
BiologyLabwareDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.biology.labware.definition.dao;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * BiologyLabwareDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface BiologyLabwareDAO extends DAO<BiologyLabware> {

}
package edu.upm.dpsm.cims.biology.labware.definition.models;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.Min;
```

```
import javax.validation.constraints.NotNull;

import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "biology_labware_definition")
@Access(AccessType.FIELD)
public class BiologyLabware extends JPAEntityModel implements MonitorItem,
Serializable {

  private static final long        serialVersionUID      = -7617390104822472391L;
  public final static String       NAME                  = "name";
  public final static String       TYPE                  = "type";
  public final static String       SIZE                  = "size";
  public final static String       OK_QUANTITY           = "okQuantity";
  public final static String       BROKEN_QUANTITY       = "brokenQuantity";
  public final static String       STAINED_QUANTITY      = "stainedQuantity";
  public final static String       TOTAL_QUANTITY        = "totalQuantity";
  public final static String       MINIMUM               = "minimum";
  public final static String       BIOLOGY_LABWARE_ITEMS = "biologyLabwareItems";

  @Column(name = "type")
  private String                   type;

  @Column(name = "size")
  private String                   size;

  @Column(name = "name")
  private String                   name;

  @Column(name = "total_quantity")
  private Long                     totalQuantity         = Long.valueOf(0L);      ;

  @Column(name = "broken_quantity")
  private Long                     brokenQuantity        = Long.valueOf(0L);      ;

  @Column(name = "stained_quantity")
  private Long                     stainedQuantity       = Long.valueOf(0L);      ;

  @Column(name = "ok_quantity")
  private Long                     okQuantity            = Long.valueOf(0L);      ;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
  private Integer                  minimum;
```

```
  @OneToMany(mappedBy = "biologyLabware", cascade = { CascadeType.REMOVE }, fetch =
FetchType.LAZY)
  private List<BiologyLabwareItem> biologyLabwareItems;

  public List<BiologyLabwareItem> getBiologyLabwareItems() {
    return biologyLabwareItems;
  }

  public void setBiologyLabwareItems(List<BiologyLabwareItem> biologyLabwareItems) {
    this.biologyLabwareItems = biologyLabwareItems;
  }

  public String getType() {
    return type;
  }

  public void setType(String type) {
    this.type = type;
  }

  public String getSize() {
    return size;
  }

  public void setSize(String size) {
    this.size = size;
  }

  public Long getTotalQuantity() {
    return totalQuantity;
  }

  public void setTotalQuantity(Long totalQuantity) {
    this.totalQuantity = totalQuantity;
  }

  public Long getBrokenQuantity() {
    return brokenQuantity;
  }

  public void setBrokenQuantity(Long brokenQuantity) {
    this.brokenQuantity = brokenQuantity;
  }

  public Long getStainedQuantity() {
    return stainedQuantity;
  }

  public void setStainedQuantity(Long stainedQuantity) {
    this.stainedQuantity = stainedQuantity;
  }

  public Long getOkQuantity() {
```

```
    return okQuantity;
  }

  public void setOkQuantity(Long okQuantity) {
    this.okQuantity = okQuantity;
  }

  @Override
  public String getName() {
    return this.name;
  }

  @Override
  public Integer getMinimum() {
    return this.minimum;
  }

  public void setName(String name) {
    this.name = name;
  }

  public void setMinimum(Integer minimum) {
    this.minimum = minimum;
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.biology.labware.definition.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.biology.labware.definition.dao.BiologyLabwareDAO;
import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;

/**
 * BiologyLabwareServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
```

```
@Transactional
public class BiologyLabwareServiceImpl extends JPAService<BiologyLabware> implements
BiologyLabwareService {

  @Autowired
  private BiologyLabwareDAO dao;

  @Override
  public DAO<BiologyLabware> getDAO() {
    return dao;
  }

}
package edu.upm.dpsm.cims.biology.labware.definition.services;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */

public interface BiologyLabwareService extends BusinessService<BiologyLabware> {

}
package edu.upm.dpsm.cims.biology.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.biology.labware.item.services.BiologyLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddBiologyLabwareItemController {
  @Autowired
  private BiologyLabwareItemService biologyLabwareItemService;

  @InitBinder
```

```
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "biology/labware/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody @Validated BiologyLabwareItem biologyLabwareItem,
      HttpServletRequest request) throws Exception {

    biologyLabwareItemService.saveItems(biologyLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_ITEM_ADD_SUCCESS).getMap();
  }
}

/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.biology.labware.item.services.BiologyLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateBiologyLabwareItemCondemnationAlertController {
  @Autowired
  private BiologyLabwareItemService biologyLabwareItemService;

  @MonitorRequest(status=EItemStatus.CONDEMNATION)
```

```
  @RequestMapping(value = "biology/labware/item/condemnation-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody BiologyLabwareItem biologyLabwareItem,
HttpServletRequest request)
      throws Exception {

    biologyLabwareItemService.createCondemnationAlert(biologyLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.CONDEMNATION_REQUEST_CREATE_SUCCESS).getMap(
);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.biology.labware.item.services.BiologyLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateBiologyLabwareItemRepairAlertController {
  @Autowired
  private BiologyLabwareItemService biologyLabwareItemService;

  @MonitorRequest(status=EItemStatus.REPAIR)
```

```
  @RequestMapping(value = "biology/labware/item/repair-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody BiologyLabwareItem biologyLabwareItem,
HttpServletRequest request)
      throws Exception {

    biologyLabwareItemService.createRepairAlert(biologyLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.REPAIR_REQUEST_CREATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.biology.labware.item.services.BiologyLabwareItemService;

/**
 * SearchChemicalLabwareItemController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class SearchBiologyChemicalLabwareItemController {
  @Autowired
  private BiologyLabwareItemService biologyLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
```

```
  }

  @RequestMapping(value = "biology/labware/item/search", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody BiologyLabwareItem biologyLabwareItem,
HttpServletRequest request)
      throws Exception {

    List<BiologyLabwareItem> list =
biologyLabwareItemService.searchLabwareItem(biologyLabwareItem);

    return DTOUtil.copy(BiologyLabwareItem.class, list, new String[] {
BiologyLabwareItem.CREATED_BY,
        BiologyLabwareItem.UPDATED_BY, BiologyLabwareItem.BIOLOGY_LABWARE });
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.biology.labware.item.dao.BiologyLabwareItemDAO;
import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * BiologyLabwareItemDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class BiologyLabwareItemDAOImpl extends JPADao<BiologyLabwareItem> implements
BiologyLabwareItemDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.dao;
```

```
import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * BiologyLabwareItemDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface BiologyLabwareItemDAO extends DAO<BiologyLabwareItem> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.models;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;

/**
 * BiologyLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "biology_labware_item")
public class BiologyLabwareItem extends JPAEntityModel implements Serializable,
Maintainable {
  public final static String LABWARE_ID        = "labwareId";
  public final static String BIOLOGY_LABWARE = "biologyLabware";
  private static final long  serialVersionUID  = 3011167584400262957L;
```

```
@Column(name = "labware_id")
private Long               labwareId;

@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "labware_id", updatable = false, insertable = false)
private BiologyLabware   biologyLabware;

@Column(name = "labware_status")
private Integer            labwareStatus;

@NotEmpty
@Transient
private String             labwareName;

@Transient
private Integer            quantity;

@Column(name="serial_number")
private String serialNumber;

public String getSerialNumber() {
  return serialNumber;
}

public void setSerialNumber(String serialNumber) {
  this.serialNumber = serialNumber;
}

public Integer getQuantity() {
  return quantity;
}

public void setQuantity(Integer quantity) {
  this.quantity = quantity;
}

public String getLabwareName() {
  return labwareName;
}

public void setLabwareName(String labwareName) {
  this.labwareName = labwareName;
}

public Long getLabwareId() {
  return labwareId;
}

public void setLabwareId(Long labwareId) {
  this.labwareId = labwareId;
}
```

```
  public BiologyLabware getBiologyLabware() {
    return biologyLabware;
  }

  public void setBiologyLabware(BiologyLabware biologyLabware) {
    this.biologyLabware = biologyLabware;
  }

  public Integer getLabwareStatus() {
    return labwareStatus;
  }

  public void setLabwareStatus(Integer labwareStatus) {
    this.labwareStatus = labwareStatus;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.search;

import static
edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem.LABWARE_ID;
import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
/**
 * SearchBiologyLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchBiologyLabwareItem extends JPASearchModel<BiologyLabwareItem> {

  public SearchBiologyLabwareItem(BiologyLabwareItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(BiologyLabwareItem model) throws Exception {
   setSearch(new SearchCondition()
     .add(LABWARE_ID,ERestrictionType.EQ, model.getLabwareId()));
  }


}
/**
```

```
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.biology.labware.item.services.impl;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.biology.labware.definition.models.BiologyLabware;
import edu.upm.dpsm.cims.biology.labware.definition.services.BiologyLabwareService;
import edu.upm.dpsm.cims.biology.labware.item.dao.BiologyLabwareItemDAO;
import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.biology.labware.item.search.SearchBiologyLabwareItem;
import edu.upm.dpsm.cims.biology.labware.item.services.BiologyLabwareItemService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.labstatus.model.ELabwareStatus;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;
import edu.upm.dpsm.cims.modules.model.EModules;

/**
 * BiologyLabwareItemServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class BiologyLabwareItemServiceImpl extends JPAService<BiologyLabwareItem>
implements BiologyLabwareItemService {

  @Autowired
  private BiologyLabwareItemDAO     dao;

  @Autowired
  private BiologyLabwareService     biologyLabwareService;

  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @Override
  public DAO<BiologyLabwareItem> getDAO() {
    return dao;
  }
```

```
  @Override
  public BiologyLabwareItem saveItems(BiologyLabwareItem biologyLabwareItem) throws
Exception {
    int quantity = biologyLabwareItem.getQuantity();
    biologyLabwareItem.setLabwareStatus(ELabwareStatus.OK.asInt());
    int row = quantity;
    while (row > 0) {
      BiologyLabwareItem item = new BiologyLabwareItem();
      PropertyUtils.copyProperties(item, biologyLabwareItem);
      super.save(item);
      row--;
    }

    // update parent LabwareDefinition total quantity
    BiologyLabware ccd = new BiologyLabware();
    ccd.setId(biologyLabwareItem.getLabwareId());
    ccd = biologyLabwareService.get(ccd);
    ccd.setTotalQuantity(ccd.getTotalQuantity() + quantity);
    ccd.setOkQuantity(ccd.getOkQuantity() + quantity);
    biologyLabwareService.update(ccd);

    return biologyLabwareItem;
  }

  @Override
  public List<BiologyLabwareItem> searchLabwareItem(BiologyLabwareItem
biologyLabwareItem) throws Exception {
    SearchBiologyLabwareItem searchLabwareItem = new
SearchBiologyLabwareItem(biologyLabwareItem);
    biologyLabwareItem.setSearchModel(searchLabwareItem);
    return super.search(biologyLabwareItem);
  }

  @Override
  public BiologyLabwareItem createRepairAlert(BiologyLabwareItem biologyLabwareItem)
throws Exception {
    BiologyLabwareItem labwareItem = super.get(biologyLabwareItem);
    String labwareName = labwareItem.getBiologyLabware().getName();
    maintenanceRequestService.createRepairAlert(labwareItem, labwareName,
EModules.BIOLOGY_LABWARE_ITEM.asLong());
    return labwareItem;
  }

  @Override
  public BiologyLabwareItem createCondemnationAlert(BiologyLabwareItem
biologyLabwareItem) throws Exception {
    BiologyLabwareItem labwareItem = super.get(biologyLabwareItem);
    String labwareName = labwareItem.getBiologyLabware().getName();
    maintenanceRequestService.createCondemnationAlert(labwareItem, labwareName,
EModules.BIOLOGY_LABWARE_ITEM.asLong());
    return labwareItem;
  }
```

```
}
package edu.upm.dpsm.cims.biology.labware.item.services;

import java.util.List;

import edu.upm.dpsm.cims.biology.labware.item.models.BiologyLabwareItem;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
public interface BiologyLabwareItemService extends
BusinessService<BiologyLabwareItem> {
  public BiologyLabwareItem saveItems(BiologyLabwareItem biologyLabwareItem) throws
Exception;

  public List<BiologyLabwareItem> searchLabwareItem(BiologyLabwareItem
biologyLabwareItem) throws Exception;

  public BiologyLabwareItem createRepairAlert(BiologyLabwareItem biologyLabwareItem)
throws Exception;

  public BiologyLabwareItem createCondemnationAlert(BiologyLabwareItem
biologyLabwareItem) throws Exception;
}
package edu.upm.dpsm.cims.chemistry.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddChemistryChemicalController {
  @Autowired
  public ChemistryChemicalService chemistryChemicalService;
```

```
  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "chemistry/chemical/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated ChemistryChemical chemistryChemical,
HttpServletRequest request)
      throws Exception {

    chemistryChemicalService.save(chemistryChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_DEFINITION_CREATE_SUCCESS).getMap()
;
  }



}
package edu.upm.dpsm.cims.chemistry.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.validation.ValidationException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.html.HtmlUtil;
import edu.upm.dpsm.cims.core.html.json.JSONErrorMap;
import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class DeleteChemistryChemicalController {

  @Autowired
  public ChemistryChemicalService chemistryChemicalService;

  @Autowired
```

```
  public MessageSource messageSource;

  @RequestMapping(value = "deleteChemicalDef", method = RequestMethod.GET)
  public @ResponseBody
  Object deleteUser(
      @ModelAttribute ChemistryChemical chemistryChemical,
      Model model, HttpServletRequest request) {
    String message;
    try {
      chemistryChemicalService
          .delete(chemistryChemical);

      message = messageSource.getMessage(
          "Success.chemical.delete", null, "",
          null);
      return new JSONSuccessMap(
          message).addProp("redirect", "chemicals.do?"
          + HtmlUtil.urlEncode(ProjectConstants.SUCCESS_MESSAGE, message))
          .getMap();
    } catch (ValidationException e) {
      e.printStackTrace();
      message = e.getMessage();
    } catch (Exception e) {
      e.printStackTrace();
      message = messageSource.getMessage(
          "Failed.userAccount.delete", null, "",
          null);
    }

    return new JSONErrorMap(message).addProp("redirect", "chemicals.do?"
        + HtmlUtil.urlEncode(ProjectConstants.ERROR_MESSAGE, message))
        .getMap();
  }

}
package edu.upm.dpsm.cims.chemistry.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
```

```
 */
@Controller
public class GetChemistryChemicalController {

  @Autowired
  private ChemistryChemicalService chemistryChemicalService;

  @RequestMapping(value = "chemistry/chemical/get", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody ChemistryChemical chemistryChemical,
HttpServletRequest request)
      throws Exception {
    ChemistryChemical chemicalItem = chemistryChemicalService.get(chemistryChemical);

    return DTOUtil.copy(ChemistryChemical.class, chemicalItem, new String[] {
ChemistryChemical.CREATED_BY,
        ChemistryChemical.UPDATED_BY, ChemistryChemical.CHEMISTRY_CHEMICAL_ITEMS });
  }

}
package edu.upm.dpsm.cims.chemistry.chemical.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchChemistryChemicalController {

  @Autowired
  private ChemistryChemicalService chemistryChemicalService;

  @RequestMapping(value = "chemistry/chemical/search", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody ChemistryChemical chemistryChemical,
HttpServletRequest request)
      throws Exception {
```

```
    List<ChemistryChemical> chemicalList =
chemistryChemicalService.search(chemistryChemical);

    return DTOUtil.copy(ChemistryChemical.class, chemicalList, new String[] {
ChemistryChemical.CREATED_BY,
        ChemistryChemical.UPDATED_BY, ChemistryChemical.CHEMISTRY_CHEMICAL_ITEMS });
  }

}
package edu.upm.dpsm.cims.chemistry.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.validation.group.UpdateChemistryChemi
cal;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class UpdateChemistryChemicalController {
  @Autowired
  public ChemistryChemicalService chemistryChemicalService;

  @RequestMapping(value = "chemistry/chemical/update-definition", method =
RequestMethod.POST)
  public @ResponseBody
  Object updateChemical(@RequestBody @Validated(UpdateChemistryChemical.class)
ChemistryChemical chemistryChemical,
      HttpServletRequest request) throws Exception {

    ChemistryChemical ccd = new ChemistryChemical();
    ccd.setId(chemistryChemical.getId());
    ccd = chemistryChemicalService.get(ccd);
    BeanUtils.copyProperties(chemistryChemical, ccd, new String[] {
ChemistryChemical.ID,
        ChemistryChemical.TOTAL_QUANTITY, ChemistryChemical.CREATED_BY_ID,
ChemistryChemical.CREATED_DATE });
```

```
    chemistryChemicalService.update(ccd);
    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_DEFINITION_UPDATE_SUCCESS).getMap()
;
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.definition.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.FetchMode;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;

import
edu.upm.dpsm.cims.chemistry.chemical.definition.dao.ChemistryChemicalDefinitionDAO;
import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * @author Miguelino San Miguel
 */
@Repository
public class ChemistryChemicalDefinitionDAOImpl extends JPADao<ChemistryChemical>
implements ChemistryChemicalDefinitionDAO {

  @Override
  public List<ChemistryChemical> search(
      ChemistryChemical chemical) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session.createCriteria(
        ChemistryChemical.class).setFetchMode(
        "createdBy",
        FetchMode.JOIN);
    return super.search(chemical, criteria);
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.definition.dao;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * @author Miguelino San Miguel
 */
public interface ChemistryChemicalDefinitionDAO extends DAO<ChemistryChemical> {
}
package edu.upm.dpsm.cims.chemistry.chemical.definition.models;

import java.io.Serializable;
import java.util.List;
```

```
import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.validation.groups.Default;

import org.hibernate.validator.constraints.NotEmpty;

import
edu.upm.dpsm.cims.chemistry.chemical.definition.validation.group.UpdateChemistryChemi
cal;
import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;

@Entity
@Table(name = "chemistry_chemical_definition")
@Access(AccessType.FIELD)
public class ChemistryChemical extends JPAEntityModel implements Serializable,
MonitorItem {

  public final static String          NAME                     = "name";
  public final static String          MOL_FORMULA              = "molFormula";
  public final static String          COLOR                    = "color";
  public final static String          TEXTURE                  = "texture";
  public final static String          GRADE                    = "grade";
  public final static String          HYGROSCOPIC              = "hygroscopic";
  public final static String          EXPLOSIVE                = "explosive";
  public final static String          TOTAL_QUANTITY           = "totalQuantity";
  public final static String          THRESH                   = "thresh";
  public final static String          MINIMUM                  = "minimum";
  public final static String          CHEMISTRY_CHEMICAL_ITEMS =
"chemistryChemicalItems";

  private static final long           serialVersionUID         =
3529801466283219405L;

  @NotEmpty
  @Size(min = 2, max = 50)
  @Column(updatable = true, name = "name")
  private String                      name;

  @Size(min = 2, max = 100, groups = { Default.class, UpdateChemistryChemical.class
})
  @Column(updatable = true, name = "mol_formula")
```

```
  private String                      molFormula;

  @Column(updatable = true, name = "color")
  private String                      color;

  @Column(updatable = true, name = "texture")
  private String                      texture;

  @Column(updatable = true, name = "grade")
  private String                      grade;

  @Column(updatable = true, name = "hygroscopic")
  private Boolean                     hygroscopic;

  @Column(updatable = true, name = "explosive")
  private Boolean                     explosive;

  @Column(updatable = true, name = "total_quantity")
  private Long                        totalQuantity            = Long.valueOf(0L);

  @Column(updatable = true, name = "thresh")
  private Integer                     thresh;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
  private Integer                     minimum;

  @OneToMany(mappedBy = "chemistryChemical", cascade = { CascadeType.REMOVE },fetch =
FetchType.LAZY)
  private List<ChemistryChemicalItem> chemistryChemicalItems;

  @Transient
  private String                      username;

  public Integer getMinimum() {
    return minimum;
  }

  public void setMinimum(Integer minimum) {
    this.minimum = minimum;
  }

  public Boolean getHygroscopic() {
    return hygroscopic;
  }

  public void setHygroscopic(Boolean hygroscopic) {
    this.hygroscopic = hygroscopic;
  }

  public Boolean getExplosive() {
    return explosive;
```

```
}

public void setExplosive(Boolean explosive) {
  this.explosive = explosive;
}

public String getColor() {
  return color;
}

public void setColor(String color) {
  this.color = color;
}

public String getGrade() {
  return grade;
}

public void setGrade(String grade) {
  this.grade = grade;
}

public String getMolFormula() {
  return molFormula;
}

public void setMolFormula(String molFormula) {
  this.molFormula = molFormula;
}

public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public String getTexture() {
  return texture;
}

public void setTexture(String texture) {
  this.texture = texture;
}

public Integer getThresh() {
  return thresh;
}

public void setThresh(Integer thresh) {
  this.thresh = thresh;
}
```

```
  public Long getTotalQuantity() {
    return totalQuantity;
  }

  public void setTotalQuantity(Long totalQuantity) {
    this.totalQuantity = totalQuantity;
  }

  public List<ChemistryChemicalItem> getChemistryChemicalItems() {
    return chemistryChemicalItems;
  }

  public void setChemistryChemicalItems(List<ChemistryChemicalItem>
chemistryChemicalItems) {
    this.chemistryChemicalItems = chemistryChemicalItems;
  }

  public String getUsername() {
    return username;
  }

  public void setUsername(String username) {
    this.username = username;
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.definition.search;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import static
edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical.*;
/**
 * @author Miguelino San Miguel
 */
public class SearchChemistryChemicals extends JPASearchModel<ChemistryChemical> {

  /**
   * @param request
   * @param model
   * @throws Exception
   */
  public SearchChemistryChemicals(
      ChemistryChemical model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(
```

```
      ChemistryChemical model)
      throws Exception {
    setSearch(new SearchCondition().add(SearchCondition
        .conjunction()
        .add(
            SearchCondition
                .disjunction()
                .add(NAME, ERestrictionType.LIKE, model.getName(),
MatchMode.ANYWHERE)
                .add(MOL_FORMULA, ERestrictionType.LIKE, model.getMolFormula(),
MatchMode.ANYWHERE))
        .and(CREATED_BY_ID, ERestrictionType.EQ, model.getCreatedById())
        .and(ID, ERestrictionType.EQ, model.getId())
         )
     );
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.definition.services.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import
edu.upm.dpsm.cims.chemistry.chemical.definition.dao.ChemistryChemicalDefinitionDAO;
import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.search.SearchChemistryChemicals;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorSupply;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional(rollbackFor = Exception.class)
public class ChemistryChemicalServiceImpl extends JPAService<ChemistryChemical>
implements
    ChemistryChemicalService {

  @Override
  public ChemistryChemical update(ChemistryChemical t) throws Exception {
    return super.update(t);
  }

  @Override
  public ChemistryChemical save(ChemistryChemical t) throws Exception {
```

```
    return super.save(t);
  }

  @Autowired
  private ChemistryChemicalDefinitionDAO dao;

  @Transactional(readOnly = true)
  @Override
  public List<ChemistryChemical> search(ChemistryChemical chemical) throws Exception
{
    Pagination pagination = chemical.getPagination();
    pagination = (pagination == null) ? (new Pagination()) : pagination;
    pagination.setTotalSize(dao.getTotalSize(chemical));
    pagination.setEnable(true);
    chemical.setPagination(pagination);
    SearchChemistryChemicals scc = new SearchChemistryChemicals(chemical);
    chemical.setSearchModel(scc);
    return dao.search(chemical);
  }

  @Override
  public DAO<ChemistryChemical> getDAO() {
    return this.dao;
  }

}
package edu.upm.dpsm.cims.chemistry.chemical.definition.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface ChemistryChemicalService extends BusinessService<ChemistryChemical>
{

}
package edu.upm.dpsm.cims.chemistry.chemical.definition.validation.group;

/**
 * @author Miguelino San Miguel
 */
public interface UpdateChemistryChemical {

}
package edu.upm.dpsm.cims.chemistry.chemical.item.controllers;

import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import
edu.upm.dpsm.cims.chemistry.chemical.item.services.ChemistryChemicalItemService;
import
edu.upm.dpsm.cims.chemistry.chemical.item.validation.validator.ExistingChemicalDefini
tionValidator;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddChemistryChemicalItemController {
  @Autowired
  private ChemistryChemicalItemService chemistryChemicalDefinitionService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
    binder.addValidators(new ExistingChemicalDefinitionValidator());
  }

  @RequestMapping(value = "chemistry/chemical/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemicalItem(@RequestBody @Validated ChemistryChemicalItem
chemistryChemicalItem,
      HttpServletRequest request) throws Exception {

    chemistryChemicalDefinitionService.saveItems(chemistryChemicalItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_ITEM_ADD_SUCCESS).getMap();
  }
}package edu.upm.dpsm.cims.chemistry.chemical.item.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.FetchMode;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;
```

```
import edu.upm.dpsm.cims.chemistry.chemical.item.dao.ChemistryChemicalItemDAO;
import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * @author Miguelino San Miguel
 */
@Repository
public class ChemistryChemicalItemDAOImpl extends
JPADao<ChemistryChemicalItem>implements ChemistryChemicalItemDAO {

  @Override
  public List<ChemistryChemicalItem> search(ChemistryChemicalItem chemicalItem)
      throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session
        .createCriteria(ChemistryChemicalItem.class)
        .setFetchMode(
            "createdBy",
            FetchMode.JOIN)
        .createAlias(ChemistryChemicalItem.CHEMISTRY_CHEMICAL_DEFINITION,
            ChemistryChemicalItem.CHEMISTRY_CHEMICAL_DEFINITION)
        .setFetchMode(ChemistryChemicalItem.CHEMISTRY_CHEMICAL_DEFINITION,
            FetchMode.JOIN);

    // apply search parameters e.g. pagination
    return super.search(chemicalItem, criteria);
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.item.dao;

import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * @author Miguelino San Miguel
 */
public interface ChemistryChemicalItemDAO extends DAO<ChemistryChemicalItem> {

}
package edu.upm.dpsm.cims.chemistry.chemical.item.models;

import java.io.Serializable;
import java.sql.Timestamp;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
```

```
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.notetype.models.NoteType;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "chemistry_chemical_item")
@Access(AccessType.FIELD)
public class ChemistryChemicalItem extends JPAEntityModel implements Serializable{

  private static final long  serialVersionUID              = -392215716780171378L;

  public final static String CHEMICAL_ID                   = "chemicalId";
  public final static String CHEMICAL_NAME                 = "chemicalName";
  public final static String STATUS                        = "consumed";
  public final static String EXPIRATION_DATE               = "expDate";
  public final static String NOTE_TYPE_ID                  = "noteTypeId";
  public final static String NOTE_TYPE                     = "noteType";
  public static final String CHEMISTRY_CHEMICAL_DEFINITION = "chemistryChemical";

  @NotNull
  @Column(updatable = false, name = "chemical_id")
  private Long               chemicalId;

  @NotEmpty
  @Transient
  private String             chemicalName;

  @Column(updatable = true, name = "consumed")
  private Boolean           consumed;

  @Column(name = "note_type")
  private Long               noteTypeId;

  @ManyToOne
  @JoinColumn(name = "note_type", updatable = false, insertable = false)
  private NoteType           noteType;

  @Column(updatable = true, name = "exp_date")
  private Timestamp          expDate;

  @Min(value = 1)
  @Transient
  private Integer            quantity                      = 1;
```

```
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "chemical_id", updatable = false, insertable = false)
private ChemistryChemical  chemistryChemical;

public Long getChemicalId() {
  return chemicalId;
}

public void setChemicalId(Long chemicalId) {
  this.chemicalId = chemicalId;
}

public Boolean getConsumed() {
  return consumed;
}

public void setConsumed(Boolean consumed) {
  this.consumed = consumed;
}

public Long getNoteTypeId() {
  return noteTypeId;
}

public void setNoteTypeId(Long noteType) {
  this.noteTypeId = noteType;
}

public NoteType getNoteType() {
  return noteType;
}

public void setNoteType(NoteType noteType) {
  this.noteType = noteType;
}

public Timestamp getExpDate() {
  return expDate;
}

/**
 * @param expDate
 *          the expDate to set
 */
public void setExpDate(Timestamp expDate) {
  this.expDate = expDate;
}

public ChemistryChemical getChemistryChemicalDefinition() {
  return chemistryChemical;
}

public void setChemistryChemicalDefinition(ChemistryChemical chemistryChemical) {
```

```
    this.chemistryChemical = chemistryChemical;
  }

  public String getChemicalName() {
    return chemicalName;
  }

  public void setChemicalName(String chemicalName) {
    this.chemicalName = chemicalName;
  }

  public Integer getQuantity() {
    return quantity;
  }

  public void setQuantity(Integer quantity) {
    this.quantity = quantity;
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.item.search;

import javax.servlet.http.HttpServletRequest;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;

/**
 * @author Miguelino San Miguel
 */
public class SearchChemicalItems extends JPASearchModel<ChemistryChemicalItem> {

  /**
   * @param request
   * @param model
   * @throws Exception
   */
  public SearchChemicalItems(ChemistryChemicalItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(ChemistryChemicalItem model) throws Exception {

    setSearch(new SearchCondition()
        .add(ChemistryChemicalItem.CHEMISTRY_CHEMICAL_DEFINITION + "." +
ChemistryChemical.NAME,
            ERestrictionType.LIKE, model.getChemicalName(), MatchMode.START)
```

```
        .add(ChemistryChemicalItem.EXPIRATION_DATE, ERestrictionType.LE,
model.getExpDate())
        .add(ChemistryChemicalItem.CREATED_BY_ID, ERestrictionType.EQ,
model.getCreatedById())
        .add(ChemistryChemicalItem.NOTE_TYPE_ID, ERestrictionType.EQ,
model.getNoteTypeId()));
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.item.services.impl;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.chemistry.chemical.item.dao.ChemistryChemicalItemDAO;
import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.chemistry.chemical.item.search.SearchChemicalItems;
import
edu.upm.dpsm.cims.chemistry.chemical.item.services.ChemistryChemicalItemService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional
public class ChemistryChemicalItemServiceImpl extends
JPAService<ChemistryChemicalItem> implements
    ChemistryChemicalItemService {

  @Autowired
  private ChemistryChemicalItemDAO           dao;

  @Autowired
  private ChemistryChemicalService chemistryChemicalService;

  @Override
  public List<ChemistryChemicalItem> search(ChemistryChemicalItem chemicalItem)
throws Exception {
    Pagination pagination = chemicalItem.getPagination();
    if (pagination == null) {
      pagination = new Pagination();
      chemicalItem.setPagination(pagination);
    }
```

```
    SearchChemicalItems scc = new SearchChemicalItems(

    chemicalItem);
    chemicalItem.setSearchModel(scc);
    pagination.setTotalSize(dao.getTotalSize(chemicalItem));
    pagination.setEnable(true);
    return this.dao.search(chemicalItem);
  }

  @Override
  public DAO<ChemistryChemicalItem> getDAO() {
    return this.dao;
  }

  @Override
  public ChemistryChemicalItem saveItems(ChemistryChemicalItem chemistryChemicalItem)
throws Exception {
    int quantity = chemistryChemicalItem.getQuantity();
    chemistryChemicalItem.setConsumed(true);
    int row = quantity;
    while (row > 0) {
      ChemistryChemicalItem item = new ChemistryChemicalItem();
      PropertyUtils.copyProperties(item, chemistryChemicalItem);
      super.save(item);
      row--;
    }

    // update parent ChemicalDefinition total quantity
    ChemistryChemical ccd = new ChemistryChemical();
    ccd.setId(chemistryChemicalItem.getChemicalId());
    ccd = chemistryChemicalService.get(ccd);
    ccd.setTotalQuantity(ccd.getTotalQuantity() + quantity);
    chemistryChemicalService.update(ccd);

    return chemistryChemicalItem;
  }

  @Override
  public ChemistryChemicalItem save(ChemistryChemicalItem t) throws Exception {
    return super.save(t);
  }
}
package edu.upm.dpsm.cims.chemistry.chemical.item.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
@Transactional
```

```
public interface ChemistryChemicalItemService extends
BusinessService<ChemistryChemicalItem> {
  public  ChemistryChemicalItem saveItems(
      ChemistryChemicalItem chemistryChemicalItem) throws Exception;
}
package edu.upm.dpsm.cims.chemistry.chemical.item.validation.group;

/**
 * @author Miguelino San Miguel
 *
 */
public interface UpdateChemicalItem {

}
package edu.upm.dpsm.cims.chemistry.chemical.item.validation.validator;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import
edu.upm.dpsm.cims.chemistry.chemical.definition.services.ChemistryChemicalService;
import edu.upm.dpsm.cims.chemistry.chemical.item.models.ChemistryChemicalItem;
import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
@Component
public class ExistingChemicalDefinitionValidator implements Validator {
      @Autowired
      private MessageSource messageSource;

      @Autowired
      private ChemistryChemicalService chemistryChemicalService;
  @Override
  public boolean supports(Class<?> clazz) {
    return ChemistryChemicalItem.class.equals(clazz);
  }

  @Override
  public void validate(Object target, Errors errors) {
    try {
      ChemistryChemicalItem chemistryChemicalItem = (ChemistryChemicalItem) target;
      Long chemicalId = chemistryChemicalItem.getChemicalId();
      if (!ObjectUtils.isEmpty(chemicalId)) {
        BindingResult result = (BindingResult) errors;
```

```
        ChemistryChemical existingDef = new ChemistryChemical();
        existingDef.setId(chemicalId);
        existingDef = chemistryChemicalService.get(existingDef);
        if (existingDef == null) {
          ValidationUtils.addBindingError(result,

ChemistryChemicalItem.CHEMICAL_NAME,EValidationMessages.NON_EXISTING_CHEMICAL_NAME);
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

}
package edu.upm.dpsm.cims.chemistry.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddChemistryLabwareController {
  @Autowired
  private ChemistryLabwareService chemistryLabwareService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "chemistry/labware/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated ChemistryLabware chemistryChemical,
HttpServletRequest request)
      throws Exception {
```

```
    chemistryLabwareService.save(chemistryChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_CREATE_SUCCESS).getMap();
  }
}
package edu.upm.dpsm.cims.chemistry.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class GetChemistryLabwareController {

  @Autowired
  private ChemistryLabwareService chemistryLabwareService;

  @RequestMapping(value = "chemistry/labware/get", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody ChemistryLabware chemistryLabware,
HttpServletRequest request) throws Exception {
    ChemistryLabware labwareItem = chemistryLabwareService.get(chemistryLabware);

    return DTOUtil.copy(ChemistryLabware.class, labwareItem, new String[] {
ChemistryLabware.CREATED_BY,
        ChemistryLabware.UPDATED_BY, ChemistryLabware.CHEMISTRY_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.chemistry.labware.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchChemistryLabwareController {

  @Autowired
  private ChemistryLabwareService chemistryLabwareService;

  @RequestMapping(value = "chemistry/labware/search", method = RequestMethod.POST)
  public @ResponseBody
  Object labwareListPage(@RequestBody ChemistryLabware chemistryLabware,
HttpServletRequest request)
      throws Exception {
    List<ChemistryLabware> chemicalList =
chemistryLabwareService.search(chemistryLabware);

    return DTOUtil.copy(ChemistryLabware.class, chemicalList, new String[] {
ChemistryLabware.CREATED_BY,
        ChemistryLabware.UPDATED_BY, ChemistryLabware.CHEMISTRY_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.chemistry.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
```

```
 */
@Controller
public class UpdateChemistryLabwareController {
  @Autowired
  public ChemistryLabwareService chemistryLabwareService;

  @RequestMapping(value = "chemistry/labware/update-definition", method =
RequestMethod.POST)
  public @ResponseBody
  Object updateChemical(@RequestBody @Validated ChemistryLabware chemistryLabware,
HttpServletRequest request)
      throws Exception {

    ChemistryLabware ccd = new ChemistryLabware();
    ccd.setId(chemistryLabware.getId());
    ccd = chemistryLabwareService.get(ccd);
    BeanUtils.copyProperties(chemistryLabware, ccd, new String[] {
ChemistryLabware.ID,
        ChemistryLabware.TOTAL_QUANTITY, ChemistryLabware.OK_QUANTITY,
ChemistryLabware.CREATED_BY_ID,
        ChemistryLabware.CREATED_DATE });
    chemistryLabwareService.update(ccd);
    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_UPDATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.definition.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.labware.definition.dao.ChemistryLabwareDAO;
import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * ChemistryLabwareDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class ChemistryLabwareDAOImpl extends JPADao<ChemistryLabware> implements
ChemistryLabwareDAO {
```

```
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.definition.dao;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * ChemistryLabwareDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface ChemistryLabwareDAO extends DAO<ChemistryLabware> {

}
package edu.upm.dpsm.cims.chemistry.labware.definition.models;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "chemistry_labware_definition")
@Access(AccessType.FIELD)
public class ChemistryLabware extends JPAEntityModel implements MonitorItem,
Serializable {
```

```
  public void setName(String name) {
    this.name = name;
  }

  public void setMinimum(Integer minimum) {
    this.minimum = minimum;
  }

  private static final long          serialVersionUID        = -7617390104822472391L;
  public final static String         NAME                    = "name";
  public final static String         TYPE                    = "type";
  public final static String         SIZE                    = "size";
  public final static String         OK_QUANTITY             = "okQuantity";
  public final static String         BROKEN_QUANTITY         = "brokenQuantity";
  public final static String         STAINED_QUANTITY        = "stainedQuantity";
  public final static String         TOTAL_QUANTITY          = "totalQuantity";
  public final static String         MINIMUM                 = "minimum";
  public final static String         CHEMISTRY_LABWARE_ITEMS =
"chemistryLabwareItems";

  @Column(name = "type")
  private String                     type;

  @Column(name = "size")
  private String                     size;

  @Column(name = "name")
  private String                     name;

  @Column(name = "total_quantity")
  private Long                       totalQuantity           = Long.valueOf(0L);
;

  @Column(name = "broken_quantity")
  private Long                       brokenQuantity          = Long.valueOf(0L);
;

  @Column(name = "stained_quantity")
  private Long                       stainedQuantity         = Long.valueOf(0L);
;

  @Column(name = "ok_quantity")
  private Long                       okQuantity              = Long.valueOf(0L);
;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
  private Integer                    minimum;

  @OneToMany(mappedBy = "chemistryLabware", cascade = { CascadeType.REMOVE }, fetch =
FetchType.LAZY)
  private List<ChemistryLabwareItem> chemistryLabwareItems;
```

```
  public List<ChemistryLabwareItem> getChemistryLabwareItems() {
    return chemistryLabwareItems;
  }

  public void setChemistryLabwareItems(List<ChemistryLabwareItem>
chemistryLabwareItems) {
    this.chemistryLabwareItems = chemistryLabwareItems;
  }

  public String getType() {
    return type;
  }

  public void setType(String type) {
    this.type = type;
  }

  public String getSize() {
    return size;
  }

  public void setSize(String size) {
    this.size = size;
  }

  public Long getTotalQuantity() {
    return totalQuantity;
  }

  public void setTotalQuantity(Long totalQuantity) {
    this.totalQuantity = totalQuantity;
  }

  public Long getBrokenQuantity() {
    return brokenQuantity;
  }

  public void setBrokenQuantity(Long brokenQuantity) {
    this.brokenQuantity = brokenQuantity;
  }

  public Long getStainedQuantity() {
    return stainedQuantity;
  }

  public void setStainedQuantity(Long stainedQuantity) {
    this.stainedQuantity = stainedQuantity;
  }

  public Long getOkQuantity() {
    return okQuantity;
  }
```

```
  public void setOkQuantity(Long okQuantity) {
    this.okQuantity = okQuantity;
  }

  @Override
  public String getName() {
    return this.name;
  }

  @Override
  public Integer getMinimum() {
    return this.minimum;
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.definition.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.labware.definition.dao.ChemistryLabwareDAO;
import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;

/**
 * ChemistryLabwareServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class ChemistryLabwareServiceImpl extends JPAService<ChemistryLabware>
implements ChemistryLabwareService {

  @Autowired
  private ChemistryLabwareDAO dao;

  @Override
  public DAO<ChemistryLabware> getDAO() {
```

```
    return dao;
  }

}
package edu.upm.dpsm.cims.chemistry.labware.definition.services;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */

public interface ChemistryLabwareService extends BusinessService<ChemistryLabware> {

}
package edu.upm.dpsm.cims.chemistry.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.chemistry.labware.item.services.ChemistryLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddChemistryLabwareItemController {
  @Autowired
  private ChemistryLabwareItemService chemistryLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "chemistry/labware/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody @Validated ChemistryLabwareItem chemistryLabwareItem,
      HttpServletRequest request) throws Exception {

    chemistryLabwareItemService.saveItems(chemistryLabwareItem);
```

```
    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_ITEM_ADD_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.chemistry.labware.item.services.ChemistryLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateChemLabwareItemCondemnationAlertController {
  @Autowired
  private ChemistryLabwareItemService chemistryLabwareItemService;

  @MonitorRequest(status=EItemStatus.CONDEMNATION)
  @RequestMapping(value = "chemistry/labware/item/condemnation-alert/create", method
= RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody ChemistryLabwareItem chemistryLabwareItem,
HttpServletRequest request)
      throws Exception {

    chemistryLabwareItemService.createCondemnationAlert(chemistryLabwareItem);
```

```
    return
JSONMapUtils.createJSONSuccess(EMessages.CONDEMNATION_REQUEST_CREATE_SUCCESS).getMap(
);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.chemistry.labware.item.services.ChemistryLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateChemLabwareItemRepairAlertController {
  @Autowired
  private ChemistryLabwareItemService chemistryLabwareItemService;

  @MonitorRequest(status=EItemStatus.REPAIR)
  @RequestMapping(value = "chemistry/labware/item/repair-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody ChemistryLabwareItem chemistryLabwareItem,
HttpServletRequest request)
      throws Exception {

    chemistryLabwareItemService.createRepairAlert(chemistryLabwareItem);
```

```
    return
JSONMapUtils.createJSONSuccess(EMessages.REPAIR_REQUEST_CREATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.chemistry.labware.item.services.ChemistryLabwareItemService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * SearchChemicalLabwareItemController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class SearchChemicalLabwareItemController {
  @Autowired
  private ChemistryLabwareItemService chemistryLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "chemistry/labware/item/search", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody ChemistryLabwareItem chemistryLabwareItem,
HttpServletRequest request)
      throws Exception {
```

```
    List<ChemistryLabwareItem> list =
chemistryLabwareItemService.searchLabwareItem(chemistryLabwareItem);

    return DTOUtil.copy(ChemistryLabwareItem.class, list, new String[] {
ChemistryLabwareItem.CREATED_BY,
        ChemistryLabwareItem.UPDATED_BY, ChemistryLabwareItem.CHEMISTRY_LABWARE });
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.labware.item.dao.ChemistryLabwareItemDAO;
import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * ChemistryLabwareItemDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class ChemistryLabwareItemDAOImpl extends JPADao<ChemistryLabwareItem>
implements ChemistryLabwareItemDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.dao;

import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * ChemistryLabwareItemDAO
 *
 * Description:
```

```
 *
 * author : Miguelino San Miguel
 */
public interface ChemistryLabwareItemDAO extends DAO<ChemistryLabwareItem> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.models;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;

/**
 * ChemistryLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "chemistry_labware_item")
public class ChemistryLabwareItem extends JPAEntityModel implements Serializable,
Maintainable {
  public final static String LABWARE_ID        = "labwareId";
  public final static String CHEMISTRY_LABWARE = "chemistryLabware";
  private static final long  serialVersionUID  = 30111675844002629 57L;

  @Column(name = "labware_id")
  private Long               labwareId;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "labware_id", updatable = false, insertable = false)
  private ChemistryLabware   chemistryLabware;
```

```
@Column(name = "labware_status")
private Integer            labwareStatus;

@NotEmpty
@Transient
private String             labwareName;

@Transient
private Integer            quantity;

@Column(name="serial_number")
private String serialNumber;

public String getSerialNumber() {
  return serialNumber;
}

public void setSerialNumber(String serialNumber) {
  this.serialNumber = serialNumber;
}

public Integer getQuantity() {
  return quantity;
}

public void setQuantity(Integer quantity) {
  this.quantity = quantity;
}

public String getLabwareName() {
  return labwareName;
}

public void setLabwareName(String labwareName) {
  this.labwareName = labwareName;
}

public Long getLabwareId() {
  return labwareId;
}

public void setLabwareId(Long labwareId) {
  this.labwareId = labwareId;
}

public ChemistryLabware getChemistryLabware() {
  return chemistryLabware;
}

public void setChemistryLabware(ChemistryLabware chemistryLabware) {
  this.chemistryLabware = chemistryLabware;
}
```

```
  public Integer getLabwareStatus() {
    return labwareStatus;
  }

  public void setLabwareStatus(Integer labwareStatus) {
    this.labwareStatus = labwareStatus;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.search;

import static
edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem.LABWARE_ID;
import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
/**
 * SearchChemistryLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchChemistryLabwareItem extends JPASearchModel<ChemistryLabwareItem>
{

  public SearchChemistryLabwareItem(ChemistryLabwareItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(ChemistryLabwareItem model) throws Exception {
   setSearch(new SearchCondition()
     .add(LABWARE_ID,ERestrictionType.EQ, model.getLabwareId()));
  }


}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.chemistry.labware.item.services.impl;
```

```
import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.chemistry.labware.definition.models.ChemistryLabware;
import
edu.upm.dpsm.cims.chemistry.labware.definition.services.ChemistryLabwareService;
import edu.upm.dpsm.cims.chemistry.labware.item.dao.ChemistryLabwareItemDAO;
import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.chemistry.labware.item.search.SearchChemistryLabwareItem;
import edu.upm.dpsm.cims.chemistry.labware.item.services.ChemistryLabwareItemService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.labstatus.model.ELabwareStatus;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;
import edu.upm.dpsm.cims.modules.model.EModules;

/**
 * ChemistryLabwareItemServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class ChemistryLabwareItemServiceImpl extends JPAService<ChemistryLabwareItem>
implements
    ChemistryLabwareItemService {

  @Autowired
  private ChemistryLabwareItemDAO   dao;

  @Autowired
  private ChemistryLabwareService   chemistryLabwareService;

  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @Override
  public DAO<ChemistryLabwareItem> getDAO() {
    return dao;
  }

  @Override
  public ChemistryLabwareItem saveItems(ChemistryLabwareItem chemistryLabwareItem)
throws Exception {
    int quantity = chemistryLabwareItem.getQuantity();
    chemistryLabwareItem.setLabwareStatus(ELabwareStatus.OK.asInt());
```

```
    int row = quantity;
    while (row > 0) {
      ChemistryLabwareItem item = new ChemistryLabwareItem();
      PropertyUtils.copyProperties(item, chemistryLabwareItem);
      super.save(item);
      row--;
    }

    // update parent LabwareDefinition total quantity
    ChemistryLabware ccd = new ChemistryLabware();
    ccd.setId(chemistryLabwareItem.getLabwareId());
    ccd = chemistryLabwareService.get(ccd);
    ccd.setTotalQuantity(ccd.getTotalQuantity() + quantity);
    ccd.setOkQuantity(ccd.getOkQuantity() + quantity);
    chemistryLabwareService.update(ccd);

    return chemistryLabwareItem;
  }

  @Override
  public List<ChemistryLabwareItem> searchLabwareItem(ChemistryLabwareItem
chemistryLabwareItem) throws Exception {
    SearchChemistryLabwareItem searchLabwareItem = new
SearchChemistryLabwareItem(chemistryLabwareItem);
    chemistryLabwareItem.setSearchModel(searchLabwareItem);
    return super.search(chemistryLabwareItem);
  }

  @Override
  public ChemistryLabwareItem createRepairAlert(ChemistryLabwareItem
chemistryLabwareItem) throws Exception {
    ChemistryLabwareItem labwareItem = super.get(chemistryLabwareItem);
    String labwareName = labwareItem.getChemistryLabware().getName();
    maintenanceRequestService.createRepairAlert(labwareItem, labwareName,
EModules.CHEMISTRY_LABWARE_ITEM.asLong());
    return labwareItem;
  }

  @Override
  public ChemistryLabwareItem createCondemnationAlert(ChemistryLabwareItem
chemistryLabwareItem) throws Exception {
    ChemistryLabwareItem labwareItem = super.get(chemistryLabwareItem);
    String labwareName = labwareItem.getChemistryLabware().getName();
    maintenanceRequestService.createCondemnationAlert(labwareItem, labwareName,
        EModules.CHEMISTRY_LABWARE_ITEM.asLong());
    return labwareItem;
  }

}
package edu.upm.dpsm.cims.chemistry.labware.item.services;

import java.util.List;
```

```
import edu.upm.dpsm.cims.chemistry.labware.item.models.ChemistryLabwareItem;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
public interface ChemistryLabwareItemService extends
BusinessService<ChemistryLabwareItem> {
  public ChemistryLabwareItem saveItems(ChemistryLabwareItem chemistryLabwareItem)
throws Exception;

  public List<ChemistryLabwareItem> searchLabwareItem(ChemistryLabwareItem
chemistryLabwareItem) throws Exception;

  public ChemistryLabwareItem createRepairAlert(ChemistryLabwareItem
chemistryLabwareItem) throws Exception;

  public ChemistryLabwareItem createCondemnationAlert(ChemistryLabwareItem
chemistryLabwareItem) throws Exception;
}
package edu.upm.dpsm.cims.core.app.dao;

import java.util.List;

import edu.upm.dpsm.cims.core.app.model.Model;

/**
 * Common behaviors/components to DAO
 *
 * @author Miguelino San Miguel
 */

public interface DAO<T extends Model> {
	public T get(T t) throws Exception;

	public List<T> search(T t) throws Exception;

	public List<T> saveAll(List<T> tList) throws Exception;

	public T save(T t) throws Exception;

	public T delete(T t) throws Exception;

	public List<T> deleteAll(List<T> t) throws Exception;

	public T update(T t) throws Exception;

	public Long getTotalSize(T t) throws Exception;
}
package edu.upm.dpsm.cims.core.app.field;

import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
```

```
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class Field {

	private String	name;
	private String	alias;
	private String	table;

	public String getName() {
		return name;
	}

	public Field setName(String name) {
		this.name = name;
		return this;
	}

	public String getAlias() {
		return alias;
	}

	public Field setAlias(String alias) {
		this.alias = alias;
		return this;
	}

	public String getTable() {
		return table;
	}

	public Field setTable(String table) {
		this.table = table;
		return this;
	}

	@Override
	public int hashCode() {
		return HashCodeBuilder.reflectionHashCode(this, false);
	}

	@Override
	public boolean equals(Object obj) {
		return EqualsBuilder.reflectionEquals(this, obj, false);
	}

	@Override
	public String toString() {
		return (!ObjectUtils.isEmpty(table) ? table + "." : "")
		    + (ObjectUtils.isEmpty(alias) ? name : alias);
```

```
	}
}
package edu.upm.dpsm.cims.core.app.model;

import java.sql.Timestamp;

import edu.upm.dpsm.cims.useraccount.models.UserAccount;


/**
 * @author Miguelino San Miguel
 */
public interface EntityModel extends Model {
	public final static String CREATED_BY     = "createdBy";
	public final static String CREATED_BY_ID= "createdById";
	public final static String CREATED_DATE = "createdDate";

	public final static String UPDATED_BY     = "updatedBy";
	public final static String UPDATED_BY_ID= "updatedById";
	public final static String UPDATED_DATE = "updatedDate";

	public UserAccount getUpdatedBy();
	public void setUpdatedBy(UserAccount updatedBy);
	public Timestamp getUpdatedDate();
	public void setUpdatedDate(Timestamp updatedDate);
	public Timestamp getCreatedDate();
	public void setCreatedDate(Timestamp createdDate);
	public UserAccount getCreatedBy();
	public void setCreatedBy(UserAccount createdBy);
	public Long getCreatedById();
	public void setCreatedById(Long createdById);
	public Long getUpdatedById();
	public void setUpdatedById(Long updatedById);
}
package edu.upm.dpsm.cims.core.app.model;


/**
 * @author Miguelino San Miguel
 */
public interface LookupModel extends Model {
}
package edu.upm.dpsm.cims.core.app.model;


/**
 * @author Miguelino San Miguel
 */
public interface Model {
	public Long getId();


}
```

```
package edu.upm.dpsm.cims.core.app.service;

import java.util.List;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.app.model.Model;

/**
 * @author Miguelino San Miguel
 */

@Transactional
public interface BusinessService<T extends Model> {
	public DAO<T> getDAO();

	public T get(T t) throws Exception;

	public List<T> saveAll(List<T> tList) throws Exception;

	public T save(T t) throws Exception;

	public T delete(T t) throws Exception;

	public List<T> deleteAll(List<T> t) throws Exception ;

	public T update(T t) throws Exception;

	public List<T> search(T t) throws Exception;

	public Long getTotalSize(T t) throws Exception;
}
package edu.upm.dpsm.cims.core.app.table;

import org.hibernate.FetchMode;

/**
 * @author Miguelino San Miguel
 */
public class Table {
  private String name;
  private String alias;
  private FetchMode fetchMode;

  /**
   * @return the tableName
   */
  public String getName() {
    return name;
  }

  /**
```

```
   * @param tableName
   *          the tableName to set
   */
  public Table setName(String tableName) {
    this.name = tableName;
    return this;
  }

  /**
   * @return the alias
   */
  public String getAlias() {
    return alias;
  }

  /**
   * @param alias
   *          the alias to set
   */
  public Table setAlias(String alias) {
    this.alias = alias;
    return this;
  }

  /**
   * @return the fetchMode
   */
  public FetchMode getFetchMode() {
    return fetchMode;
  }

  /**
   * @param fetchMode
   *          the fetchMode to set
   */
  public Table setFetchMode(FetchMode fetchMode) {
    this.fetchMode = fetchMode;
    return this;
  }
}
package edu.upm.dpsm.cims.core.app.validation.constant;

import edu.upm.dpsm.cims.core.message.IResourceMessage;

public enum EValidationMessages implements IResourceMessage{
  /*
   *User Account
   */
  LOGIN_AUTHORIZATION("login.authorization.required"),

  NON_EXISTING_ACCOUNT("NotFound.loginForm.account"),
  INVALID_CONFIRMED_PASSWORD("Invalid.userAccount.confirmedPassword"),
  INVALID_OLD_PASSWORD("Invalid.userAccount.oldPassword"),
```

```
  UNAVAILABLE_USERNAME("NotAvailable.userAccount.username"),
  INVALID_SELF_UPDATE_ROLE("Invalid.userAccount.admin.update.self")

  /*
   * Chemical
   */
  ,
  NON_EXISTING_CHEMICAL_NAME("NotExisting.chemicalName"),
  CHEMICAL_OUT_OF_QUANTITY("Unavailable.chemical.quantity"),
  CHEMICAL_LACKING("Lacking.chemical.quantity"),

  /*
   *Labware
   **/

  /*
   * Math/CS Machines
   */
  ;
  private String messageKey;

  private EValidationMessages(String messageKey) {
    this.messageKey = messageKey;
  }

  @Override
  public String getMessageKey() {
    return this.messageKey;
  }
}
package edu.upm.dpsm.cims.core.app.validation.constraint;

import static java.lang.annotation.ElementType.ANNOTATION_TYPE;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.validation.Constraint;
import javax.validation.Payload;

import edu.upm.dpsm.cims.core.app.validation.validator.FieldMatchValidator;

/**
 * Validation annotation to validate that 2 fields have the same value. An array
 * of fields and their matching confirmation fields can be supplied. Example,
 * compare 1 pair of fields:
 *
 * @FieldMatch(first = "password", second = "confirmPassword", message =
 *                   "The password fields must match") Example, compare more
 *                   than 1 pair of fields:
```

```
 * @FieldMatch.List({
 * @FieldMatch(first = "password", second = "confirmPassword", message =
 *                   "The password fields must match"),
 * @FieldMatch(first = "email", second = "confirmEmail", message =
 *                   "The email fields must match")})
 */
@Target({ TYPE, ANNOTATION_TYPE })
@Retention(RUNTIME)
@Constraint(validatedBy = FieldMatchValidator.class)
@Documented
public @interface FieldMatch
{
  String message() default "{constraints.fieldmatch}";

  Class<?>[] groups() default {};

  Class<? extends Payload>[] payload() default {};

  /**
   * @return The first field
   */
  String first();

  /**
   * @return The second field
   */
  String second();

  /**
   * Defines several <code>@FieldMatch</code> annotations on the same element
   *
   * @see FieldMatch
   */
  @Target({ TYPE, ANNOTATION_TYPE })
  @Retention(RUNTIME)
  @Documented
  @interface List
  {
    FieldMatch[] value();
  }


}
```

```
package edu.upm.dpsm.cims.core.app.validation.exceptions;

import javax.validation.ValidationException;

/**
 * @author Miguelino San Miguel
 */
public class FieldValidationException extends ValidationException {
  private static final long serialVersionUID = -1446720573665273809L;
```

```
  protected String fieldName;

  public FieldValidationException(String fieldName, String message) {
    super(message);
    this.fieldName = fieldName;
  }

  /**
   * @return the fieldName
   */
  public String getFieldName() {
    return fieldName;
  }

}
package edu.upm.dpsm.cims.core.app.validation.util;

import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;

import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.exceptions.FieldValidationException;
import edu.upm.dpsm.cims.core.html.json.ValidationMap;
import edu.upm.dpsm.cims.core.message.MessagesUtils;
import edu.upm.dpsm.cims.core.util.MessageUtil;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class ValidationUtils {
	public static final String GENERAL_FORM_ERRORS = "formErrors";
	private final static Logger logger = Logger.getLogger(ValidationUtils.class);

	private static FieldError createFieldError(String targetName,
			String fieldName, String message) {
		return new FieldError(targetName, fieldName, message);
	}

	public static void addBindingError(BindingResult result, String fieldName,
			String message) {
		result.addError(createFieldError(result.getObjectName(), fieldName,
				message));
	}

	public static void addBindingError(BindingResult result, String fieldName,
			EValidationMessages validationCode) {
```

```
			result.addError(createFieldError(result.getObjectName(), fieldName,
					MessageUtil.getMessage(validationCode.getMessageKey())));
		}

		public static void bindValidationException(BindingResult result,
				FieldValidationException fve) {
			addBindingError(result, fve.getFieldName(), fve.getMessage());
		}

		public static Map<String, Object> toMap(BindingResult result) {
			ValidationMap map = new ValidationMap();
			List<ObjectError> objErrors = result.getAllErrors();
			for (ObjectError error : objErrors) {
				String key = GENERAL_FORM_ERRORS;
				if (error instanceof FieldError) {
					key = ((FieldError) error).getField();
				}

				map.addError(key, getLocalizedMessage(error));
			}
			return map.getMap();
		}

		private static String getLocalizedMessage(ObjectError error) {

			String localizedMessage = error.getDefaultMessage();

			String[] codes = error.getCodes();

			if (!ObjectUtils.isEmpty(codes)) {
				for (String code : codes) {
					try {
						String customMessage =
MessagesUtils.getMessage(code, error.getArguments());
						if (!ObjectUtils.isEmpty(customMessage)) {
							localizedMessage = customMessage;
							break;
						}
					} catch (Exception e) {
						logger.error(e.getMessage());
					}

				}
			}
			return localizedMessage;
		}

}
package edu.upm.dpsm.cims.core.app.validation.validator;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;
```

```
import org.apache.commons.beanutils.PropertyUtils;

import edu.upm.dpsm.cims.core.app.validation.constraint.FieldMatch;

public class FieldMatchValidator implements ConstraintValidator<FieldMatch, Object>
{
  private String firstFieldName;
  private String secondFieldName;

  @Override
  public void initialize(final FieldMatch constraintAnnotation)
  {
    firstFieldName = constraintAnnotation.first();
    secondFieldName = constraintAnnotation.second();
  }

  @Override
  public boolean isValid(final Object target,
      final ConstraintValidatorContext context)
  {
    try
    {
      final Object firstObj = PropertyUtils.getProperty(target, firstFieldName);
      final Object secondObj = PropertyUtils
          .getProperty(target, secondFieldName);

      return (firstObj == null && secondObj == null) || (firstObj != null
          && firstObj.equals(secondObj));
    } catch (final Exception ignore)
    {
      // ignore
    }
    return true;
  }


}
```

```
package edu.upm.dpsm.cims.core.converter;

import java.beans.PropertyEditorSupport;
import java.sql.Timestamp;
import java.util.TimeZone;

import org.apache.commons.beanutils.ConvertUtils;
import org.apache.commons.beanutils.converters.DateTimeConverter;
import org.joda.time.DateTime;
import org.joda.time.DateTimeUtils;
import org.joda.time.DateTimeZone;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;

import edu.upm.dpsm.cims.core.util.ObjectUtils;
```

```
/**
 * @author Miguelino San Miguel
 */
public class TimestampEditor extends PropertyEditorSupport {

  private DateTimeFormatter formatter;

  public TimestampEditor() {
    super();
    this.formatter = DateTimeFormat.fullDate();
  }

  /**
   * @param source
   */
  public TimestampEditor(String pattern, TimeZone timeZone) {
    super();
    this.formatter = DateTimeFormat.forPattern(pattern).withZone(
        DateTimeZone.forTimeZone(timeZone));
  }

  @Override
  public String getAsText() {

    Object value = super.getValue();
    if (value != null) {
      return formatter.print(((Timestamp) value).getTime());
    }
    return "";
  }

  @Override
  public void setAsText(String text) throws IllegalArgumentException {
    try {
      if (!ObjectUtils.isEmpty(text)) {
        DateTime dt = formatter.parseDateTime(text);
        Timestamp t = new Timestamp(dt.getMillis());
        super.setValue(t);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Feb 18, 2014
 */
package edu.upm.dpsm.cims.core.dto;
```

```
import java.sql.Timestamp;

import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * EntityDTO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class EntityDTO extends ModelDTO{

  private Long        createdById;

  private Timestamp   createdDate;

  private Long        updatedById;

  private Timestamp   updatedDate;

  public Long getCreatedById() {
    return createdById;
  }

  public void setCreatedById(Long createdById) {
    this.createdById = createdById;
  }

  public Timestamp getCreatedDate() {
    return createdDate;
  }

  public void setCreatedDate(Timestamp createdDate) {
    this.createdDate = createdDate;
  }

  public Long getUpdatedById() {
    return updatedById;
  }

  public void setUpdatedById(Long updatedById) {
    this.updatedById = updatedById;
  }

  public Timestamp getUpdatedDate() {
    return updatedDate;
  }

  public void setUpdatedDate(Timestamp updatedDate) {
    this.updatedDate = updatedDate;
  }
```

```
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Feb 18, 2014
 */
package edu.upm.dpsm.cims.core.dto;

import edu.upm.dpsm.cims.core.app.model.Model;

/**
 * ModelDTO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class ModelDTO  implements Model{

  private Long id;

  @Override
  public Long getId() {
    return id;
  }

  public void setId(Long id) {
    this.id = id;
  }

}
/**
 * CIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.exception;

import edu.upm.dpsm.cims.core.message.IResourceMessage;

/**
 * AuthorizationException
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class AuthorizationException extends LIMSException {
```

```
  private static final long serialVersionUID = 3318668288089201166L;

  private EAuthType         authType;

  public AuthorizationException(IResourceMessage message) {
    super(message);
  }

  public AuthorizationException(IResourceMessage message, EAuthType authType) {
    super(message);
    this.authType = authType;
  }

  public EAuthType getAuthType() {
    return this.authType;
  }
}
/**
 * CIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.exception;

/**
 * EAuthType
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum EAuthType {
  LOGIN;
}
package edu.upm.dpsm.cims.core.exception;

import edu.upm.dpsm.cims.core.message.IResourceMessage;

public enum EProjectExceptionMessage implements IResourceMessage {
  INTERNAL_SERVER_ERROR("");
  String messageKey;

  private EProjectExceptionMessage(String key) {
    this.messageKey = key;
  }

  @Override
  public String getMessageKey() {
    return this.messageKey;
  }
```

```
}
package edu.upm.dpsm.cims.core.exception;

import edu.upm.dpsm.cims.core.message.IResourceMessage;
import edu.upm.dpsm.cims.core.message.MessagesUtils;

public class LIMSException extends RuntimeException {

	private static final long serialVersionUID = -4608727221655242559L;

	public LIMSException(){
	}

	public LIMSException(IResourceMessage message){
	  super(MessagesUtils.getMessage(message));
	}

}
package edu.upm.dpsm.cims.core.html.json;

public enum ErrorStatus {
	VALIDATION, AUTH, SERVER;


}
package edu.upm.dpsm.cims.core.html.json;

/**
 * @author Miguelino San Miguel
 */
public class JSONErrorMap extends JSONMap {
	protected static final String ERROR = "error";

	public JSONErrorMap(String message) {
		super(message, false);
		this.addProp(ERROR, ErrorStatus.SERVER);
	}

	public JSONErrorMap(ErrorStatus errorFlag, String message) {
		super(message, false);
		this.addProp(ERROR, errorFlag);
	}

	public JSONErrorMap() {
		this(null);
	}

	@Override
	public JSONErrorMap setMessage(String message) {
		super.setMessage(message);
		return this;
```

```
	}

	@Override
	public JSONErrorMap setSuccess(boolean success) {
		super.setSuccess(success);
		return this;
	}

	@Override
	public <T> JSONErrorMap addProp(String key, T value) {
		super.addProp(key, value);
		return this;
	}
}
package edu.upm.dpsm.cims.core.html.json;

import java.util.HashMap;
import java.util.Map;

/**
 * @author Miguelino San Miguel
 */
public class JSONMap {
  protected final Map<String, Object> map;
  private static final String MESSAGE = "message";
  private static final String SUCCESS = "success";

  public JSONMap() {
    map = new HashMap<String, Object>();
  }

  public JSONMap(String message, boolean isSuccess) {
    this();
    setMessage(message);
    setSuccess(isSuccess);
  }

  public JSONMap setMessage(String message) {
    this.map.put(MESSAGE, message);
    return this;
  }

  public String getMessage() {
    return (String) this.map.get(MESSAGE);
  }

  public JSONMap setSuccess(boolean success) {
    this.map.put(SUCCESS, success);
    return this;
  }

  public boolean isSuccess() {
    return (Boolean) this.map.get(SUCCESS);
```

```
  }

  public Map<String, Object> getMap() {
    return this.map;
  }

  public <T> JSONMap addProp(String key, T value) {
    this.map.put(key, value);
    return this;
  }

  public Object getProp(String key) {
    return this.map.get(key);

  }
}
/**
 * CIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.html.json;

import edu.upm.dpsm.cims.core.message.IResourceMessage;
import edu.upm.dpsm.cims.core.message.MessagesUtils;

/**
 * JSONMapUtils
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class JSONMapUtils {

  public static JSONSuccessMap createJSONSuccess(IResourceMessage message) {
    return new JSONSuccessMap(MessagesUtils.getMessage(message));
  }

  public static JSONErrorMap createJSONError(ErrorStatus jsonErrorStatus,
IResourceMessage message) {
    return new JSONErrorMap(jsonErrorStatus, MessagesUtils.getMessage(message));
  }
}
package edu.upm.dpsm.cims.core.html.json;

/**
 * @author Miguelino San Miguel
 */
public class JSONSuccessMap extends JSONMap {
```

```
  public JSONSuccessMap(String message) {
    super(message, true);
  }

  public JSONSuccessMap() {
    this(null);
  }

  @Override
  public JSONSuccessMap setMessage(String message) {
    super.setMessage(message);
    return this;
  }

  @Override
  public JSONSuccessMap setSuccess(boolean success) {
    super.setSuccess(success);
    return this;
  }

  @Override
  public <T> JSONSuccessMap addProp(String key, T value) {
    super.addProp(key, value);
    return this;
  }
}
package edu.upm.dpsm.cims.core.html.json;

import java.io.PrintWriter;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.lang3.CharEncoding;
import org.codehaus.jackson.map.DeserializationConfig;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.map.SerializationConfig;
import org.springframework.http.MediaType;

/**
 * @author Miguelino San Miguel
 */
public class JSONUtil {
  private static ObjectMapper mapper;

  public static void initMapper(ObjectMapper mapperInstance) {
    if (mapper == null) {
      JSONUtil.mapper = mapperInstance;
      JSONUtil.mapper.configure(SerializationConfig.Feature.WRITE_NULL_MAP_VALUES,
false);
      JSONUtil.mapper.configure(SerializationConfig.Feature.FAIL_ON_EMPTY_BEANS,
false);
```

```
JSONUtil.mapper.configure(DeserializationConfig.Feature.FAIL_ON_UNKNOWN_PROPERTIES,
false);
      JSONUtil.mapper.configure(SerializationConfig.Feature.USE_ANNOTATIONS, true);
      JSONUtil.mapper.configure(SerializationConfig.Feature.WRITE_NULL_PROPERTIES,
false);
    }
  }

  public static ObjectMapper getMapper() {
    return mapper;
  }

  public static String toJSONString(Object object) {
    try {
      return mapper.writeValueAsString(object);
    } catch (Exception e) {
      e.printStackTrace();
    }
    return "";
  }

  public static <T> T readValue(String jsonString, Class<T> t) throws Exception {
    try {
      return mapper.readValue(jsonString, t);
    } catch (Exception e) {
      e.printStackTrace();
    }
    return null;
  }

  public static void sendJSONResponse(HttpServletResponse response, String jsonObj)
throws Exception {
    response.setCharacterEncoding(CharEncoding.UTF_8);
    response.setContentType(MediaType.APPLICATION_JSON_VALUE);
    PrintWriter out = response.getWriter();
    out.print(jsonObj);
    out.flush();
    out.close();
  }

  public static void sendJSONResponse(HttpServletResponse response, Object jsonObj)
throws Exception {
    sendJSONResponse(response, toJSONString(jsonObj));
  }
}
package edu.upm.dpsm.cims.core.html.json;

import java.util.HashMap;
import java.util.Map;

/**
 * @author Miguelino San Miguel
```

```
 */
public class ValidationMap extends JSONErrorMap {
	private static final String VALIDATION = "validations";
	private static final String HTML_NEW_LINE = "<br>";
	private final Map<String, String> vMap;

	public ValidationMap() {
		super();
		vMap = new HashMap<String, String>();
		this.addProp(ERROR, ErrorStatus.VALIDATION).addProp(VALIDATION, vMap);
	}

	public ValidationMap addError(String field, String validationMessage) {
		String value = vMap.get(field);
		if (value != null && !value.isEmpty()) {
			validationMessage = value + HTML_NEW_LINE + validationMessage;
		}
		vMap.put(field, validationMessage);
		return this;
	}

}
package edu.upm.dpsm.cims.core.html;

/**
 * @author Miguelino San Miguel
 */
public class HtmlConstants {

  public static final String TABLE_HEADER_METADATA = "TableHeaderMetadata";
  public static final String TABLE_BODY_METADATA = "TableBodyMetadata";
  public static final String AUTH_TYPE = "authType";

}
package edu.upm.dpsm.cims.core.html;

import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import javax.servlet.http.HttpServletRequest;

import org.apache.commons.beanutils.PropertyUtils;
import org.apache.commons.lang3.CharEncoding;

import edu.upm.dpsm.cims.core.PageConstant;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
```

```
 */
public class HtmlUtil {

  public static String urlEncode(Map<String, String> params) throws Exception {
    StringBuilder encoded = new StringBuilder();
    String[] keyArray = params.keySet().toArray(new String[] {});
    String firstKey = keyArray[0];
    String firstValue = params.remove(keyArray[0]);
    encoded.append(urlEncode(firstKey, firstValue));
    for (Entry<String, String> _param : params.entrySet()) {
      encoded.append("&" + urlEncode(_param.getKey(), _param.getValue()));
    }
    return encoded.toString();
  }

  public static String urlEncode(String key, String value) {
    try {
      return key + "=" + URLEncoder.encode(value, CharEncoding.UTF_8);
    } catch (Exception e) {
      e.printStackTrace();
    }
    return "";
  }

}
package edu.upm.dpsm.cims.core.interceptors;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.annotation.PostConstruct;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.lang3.CharEncoding;
import org.springframework.http.HttpStatus;
import org.springframework.web.servlet.HandlerMapping;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.exception.EAuthType;
import edu.upm.dpsm.cims.core.html.HtmlConstants;
import edu.upm.dpsm.cims.core.html.json.ErrorStatus;
import edu.upm.dpsm.cims.core.html.json.JSONErrorMap;
import edu.upm.dpsm.cims.core.html.json.JSONUtil;
import edu.upm.dpsm.cims.core.message.MessagesUtils;
import edu.upm.dpsm.cims.core.util.UserAccountUtil;

/**
 * @author Miguelino San Miguel
```

```
 */

public class LoginInterceptor extends HandlerInterceptorAdapter {

  List<String> excludedUrls = new ArrayList<String>();

  @PostConstruct
  protected void setExclusion() {
    excludedUrls.clear();
    excludedUrls.add("/");
    excludedUrls.add("/login");

    excludedUrls.add("/src**");
    excludedUrls.add("/app**");
    excludedUrls.add("/components**");
    excludedUrls.add("/assets**");

    excludedUrls.add("/index");
    excludedUrls.add("/getUserTypeList");
  }

  private void storeUserAccountOnThread(HttpServletRequest request) {
    HttpSession session = request.getSession(false);
    if (session != null) {
      Long userAccountId = (Long)
session.getAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION);
      UserAccountUtil.setUserAccountId(userAccountId);
    }
  }

  @Override
  public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler) throws Exception {
    storeUserAccountOnThread(request);
    String pattern = (String)
request.getAttribute(HandlerMapping.BEST_MATCHING_PATTERN_ATTRIBUTE);
    if (requiredLoginAccess(pattern)) {
      HttpSession session = request.getSession(false);

      if (session == null ||
session.getAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION) == null) {
        processLoginErrorResponse(request, response);
        return false;
      }

    }
    return super.preHandle(request, response, handler);
  }

  private boolean requiredLoginAccess(String pattern) {
    for (String excluded : excludedUrls) {
      if (excluded.endsWith("**")) {
        String prefix = excluded.substring(0, excluded.length() - 2);
```

```
        if (pattern.startsWith(prefix)) {
          return false;
        }
      } else {
        if (pattern.endsWith(excluded)) {
          return false;
        }
      }
    }
    return true;
  }

  private void processLoginErrorResponse(HttpServletRequest request,
HttpServletResponse response) throws Exception {
    response.setCharacterEncoding(CharEncoding.UTF_8);
    response.setStatus(HttpStatus.UNAUTHORIZED.value());
    PrintWriter writer = response.getWriter();
    writer.print(JSONUtil.toJSONString(new JSONErrorMap(ErrorStatus.AUTH,
MessagesUtils

.getMessage(EValidationMessages.LOGIN_AUTHORIZATION)).addProp(HtmlConstants.AUTH_TYPE
, EAuthType.LOGIN).getMap()));
    writer.flush();
  }
}
package edu.upm.dpsm.cims.core.jpa.dao;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;
import java.util.Set;

import org.apache.log4j.Logger;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.CriteriaSpecification;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Example;
import org.hibernate.criterion.MatchMode;
import org.hibernate.criterion.Order;
import org.hibernate.criterion.Projections;
import org.hibernate.criterion.Property;
import org.hibernate.criterion.Subqueries;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.PageConstant;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.app.model.EntityModel;
import edu.upm.dpsm.cims.core.jpa.model.JPAModel;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ConditionClause;
```

```
import edu.upm.dpsm.cims.core.jpa.search.order.EOrderType;
import edu.upm.dpsm.cims.core.jpa.search.order.JPAOrderClause;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.util.HibernateUtil;
import edu.upm.dpsm.cims.core.util.DateUtil;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.core.util.UserAccountUtil;

/**
 * @author Miguelino San Miguel
 *
 */
@Transactional
public abstract class JPADao<T extends JPAModel> implements DAO<T> {

  @Autowired
  protected HibernateUtil     hibernateUtil;
  private final static Logger logger = Logger.getLogger(JPADao.class);

  protected void prepareSearchProjection(T t, Criteria criteria) {
    // TODO
    JPASearchModel<T> bsm = (JPASearchModel<T>) t.getSearchModel();
    if (bsm != null) {

    }
  }

  @Override
  public List<T> deleteAll(List<T> tList) throws Exception {
    for(T t : tList){
      this.delete(t);
    }
    return tList;
  }

  protected void prepareSearchPagination(T t, Criteria criteria) throws Exception {
    Pagination p = t.getPagination();
    if (p != null) {
      Long totalSize = ObjectUtils.isEmpty(p.getTotalSize()) ? this.getTotalSize(t) :
p.getTotalSize();
      Long currentPage = ObjectUtils.isEmpty(p.getPage()) ? 1 : p.getPage();
      Long pageSize = ObjectUtils.isEmpty(p.getPageSize()) ? PageConstant.PAGE_SIZE :
p.getPageSize();
      Long totalPage = (long) (Math.ceil(totalSize.doubleValue() /
pageSize.doubleValue()));
      currentPage = ((currentPage > 1) && (currentPage > totalPage)) ? totalPage :
(currentPage);
      Long firstRow = ((currentPage - 1) * pageSize);
      Long lastRow = firstRow + pageSize;
      lastRow = lastRow > totalSize ? totalSize : lastRow;

      p.setPage(currentPage);
      p.setPageSize(pageSize);
```

```
      p.setTotalPage(totalPage);
      p.setFirstRow(firstRow + 1);
      p.setLastRow(lastRow + 1);

      criteria.setFirstResult(firstRow.intValue());
      criteria.setMaxResults(lastRow.intValue());
    }
  }

  protected void prepareSearchCriteria(T t, Criteria criteria) throws Exception {
    prepareSearchProjection(t, criteria);

    JPASearchModel<JPAModel> bsm = (JPASearchModel<JPAModel>) t.getSearchModel();
    if (bsm != null) {
      ConditionClause sc = bsm.getWhere();
      logger.info(sc.toString());
      if (sc != null) {
        Criterion crit = sc.getCriteria();
        if (crit != null) {
          criteria.add(crit);
        }
      }
    } else if (t != null) {

criteria.add(Example.create(t).enableLike(MatchMode.ANYWHERE).excludeZeroes().ignoreC
ase());
    }
  }

  protected List<Long> preparePaginatedCriteriaForId(T t) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Class<?> clazz = t.getClass();
    Criteria criteria = session.createCriteria(clazz);
    criteria.setProjection(Projections.distinct(Projections.id()));

    criteria.add(Subqueries.propertyIn(JPAModel.ID,
prepareSearchDetachedCriteria(t)));

    prepareSearchPagination(t, criteria);

    prepareSearchOrder(t, criteria);

    List<Long> ids = criteria.list();
    return !ObjectUtils.isEmpty(ids) ? ids : Arrays.asList(new Long[] { 0L });
  }

  protected DetachedCriteria prepareSearchDetachedCriteria(T t) throws Exception {
    Class<?> clazz = t.getClass();
    // String subqueryAlias = "subquery";
    DetachedCriteria criteria = DetachedCriteria.forClass(clazz);
    criteria.setProjection(Projections.id());

    JPASearchModel<JPAModel> bsm = (JPASearchModel<JPAModel>) t.getSearchModel();
```

```
    if (bsm != null) {
      ConditionClause sc = bsm.getWhere();
      logger.info(sc.toString());
      if (sc != null) {
        // bsm.setTableName(subqueryAlias);

        Criterion crit = sc.getCriteria();
        if (crit != null) {
          criteria.add(crit);
        }
      }
    } else if (t != null) {

criteria.add(Example.create(t).enableLike(MatchMode.ANYWHERE).excludeZeroes().ignoreC
ase());
    }
    return criteria;
  }

  protected void prepareSearchOrder(T t, Criteria criteria) throws Exception {
    // TODO Order dynamic
    JPASearchModel<T> bsm = (JPASearchModel<T>) t.getSearchModel();
    if (bsm != null) {
      JPAOrderClause oc = bsm.getOrderClause();
      if (oc != null && !ObjectUtils.isEmpty(oc.getOrderSet())) {
        Set<edu.upm.dpsm.cims.core.jpa.search.order.Order> orderFields =
oc.getOrderSet();
        for (edu.upm.dpsm.cims.core.jpa.search.order.Order order : orderFields) {
          if (EOrderType.ASC.equals(order.getOrderType())) {
            criteria.addOrder(Order.asc(order.getOrderBy().getName()));
          } else {
            criteria.addOrder(Order.desc(order.getOrderBy().getName()));
          }
        }
        return;
      }
    }
    criteria.addOrder(Order.desc(JPAModel.ID));
  }

  public List<T> search(T t) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Class<?> clazz = t.getClass();
    Criteria criteria = session.createCriteria(clazz);
    criteria.setResultTransformer(CriteriaSpecification.DISTINCT_ROOT_ENTITY);
    return this.search(t, criteria);
  }

  public List<T> search(T t, Criteria criteria) throws Exception {

    // apply search parameters e.g. pagination
    Pagination p = t.getPagination();
```

```
    if (p != null && p.isEnable()) {

criteria.add(Property.forName(JPAModel.ID).in(this.preparePaginatedCriteriaForId(t)))
;
    } else {
      criteria.add(Subqueries.propertyIn(JPAModel.ID,
prepareSearchDetachedCriteria(t)));
      prepareSearchOrder(t, criteria);
    }

    return criteria.list();
  }

  protected void setTimestamp(T t) throws Exception {
    if (t instanceof EntityModel) {
      Long userId = UserAccountUtil.getUserAccountId();

      if (userId == null) {
        // TODO throw exception
      }

      Timestamp currentTime = DateUtil.getCurrentTime();

      EntityModel cm = (EntityModel) t;

      if (ObjectUtils.isEmpty(t.getId())) {
        cm.setCreatedDate(currentTime);
        cm.setCreatedById(userId);
      }
      cm.setUpdatedDate(currentTime);
      cm.setUpdatedById(userId);
    }
  }

  public List<T> saveAll(List<T> tList) throws Exception {
    for (T t : tList) {
      this.save(t);
    }
    return tList;
  }

  public T save(T t) throws Exception {
    if (!ObjectUtils.isEmpty(t.getId())) {
      return update(t);
    }
    setTimestamp(t);
    Session session = this.hibernateUtil.getSession();
    session.save(t);
    return t;
  }

  public T delete(T t) throws Exception {
    Session session = this.hibernateUtil.getSession();
```

```
    session.delete(t);
    return t;
  }

  public T update(T t) throws Exception {
    setTimestamp(t);
    Session session = this.hibernateUtil.getSession();
    session.update(t);
    return t;
  }

  @SuppressWarnings("unchecked")
  public T get(T t) throws Exception {
    Session session = this.hibernateUtil.getSession();
    return (T) session.get(t.getClass(), t.getId());
  }

  public Long getTotalSize(T t) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Class<JPAModel> clazz = (Class<JPAModel>) t.getClass();
    Criteria criteria = session.createCriteria(clazz);
    criteria.setProjection(Projections.countDistinct(JPAModel.ID));
    criteria.add(Subqueries.propertyIn(JPAModel.ID,
prepareSearchDetachedCriteria(t)));

    Object result = criteria.uniqueResult();
    return (Long) result;
  }

  public void closeSession() {
    this.hibernateUtil.closeSession();
  }
}
package edu.upm.dpsm.cims.core.jpa.model;

import java.sql.Timestamp;

import javax.persistence.Column;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.MappedSuperclass;

import edu.upm.dpsm.cims.core.app.model.EntityModel;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 *
 */
@MappedSuperclass
public class JPAEntityModel extends JPAModel implements EntityModel {
```

```
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "created_by", updatable = false, insertable = false)
private UserAccount createdBy;

@Column(name = "created_by", nullable = false)
private Long        createdById;

@Column(updatable = false, name = "created_date", nullable = false)
private Timestamp   createdDate;

@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "updated_by", updatable = false, insertable = false)
private UserAccount updatedBy;

@Column(name = "updated_by", nullable = false)
private Long        updatedById;

@Column(updatable = true, name = "updated_date", nullable = false)
private Timestamp   updatedDate;

public UserAccount getUpdatedBy() {
  return updatedBy;
}

public void setUpdatedBy(UserAccount updatedBy) {
  this.updatedBy = updatedBy;
}

public Timestamp getUpdatedDate() {
  return updatedDate;
}

public void setUpdatedDate(Timestamp updatedDate) {
  this.updatedDate = updatedDate;
}

public Timestamp getCreatedDate() {
  return createdDate;
}

public void setCreatedDate(Timestamp createdDate) {
  this.createdDate = createdDate;
}

public UserAccount getCreatedBy() {
  return createdBy;
}

public void setCreatedBy(UserAccount createdBy) {
  this.createdBy = createdBy;
}

public Long getCreatedById() {
```

```
    return createdById;
  }

  public void setCreatedById(Long createdById) {
    this.createdById = createdById;
  }

  public Long getUpdatedById() {
    return updatedById;
  }

  public void setUpdatedById(Long updatedById) {
    this.updatedById = updatedById;
  }

}
package edu.upm.dpsm.cims.core.jpa.model;

import javax.persistence.MappedSuperclass;

import edu.upm.dpsm.cims.core.app.model.LookupModel;

/**
 * @author Miguelino San Miguel
 *
 */
@MappedSuperclass
public class JPALookupModel extends JPAModel implements LookupModel{

}
package edu.upm.dpsm.cims.core.jpa.model;

import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.MappedSuperclass;
import javax.persistence.Transient;

import org.codehaus.jackson.annotate.JsonIgnore;
import org.hibernate.annotations.GenericGenerator;

import edu.upm.dpsm.cims.core.app.model.Model;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;

/**
 * @author Miguelino San Miguel
 *
 */
@MappedSuperclass
public class JPAModel implements Model{

  public static final String ID = "id";
```

```
@JsonIgnore
@Transient
protected transient Pagination pagination;

@JsonIgnore
@Transient
protected transient JPASearchModel<? extends JPAModel> searchModel;

@Id
@GenericGenerator(name = "generic", strategy = "increment")
@GeneratedValue(generator = "generic")
@Column(updatable = false, name = "id", nullable = false, length = 10)
private Long id;

/**
 * @return the searchParameter
 */
public Pagination getPagination() {
  return pagination;
}

/**
 * @param pagination
 *          the searchParameter to set
 */
public void setPagination(Pagination pagination) {
  this.pagination = pagination;
}

/**
 * @return the id
 */
public Long getId() {
  return id;
}

/**
 * @param id
 *          the id to set
 */
public void setId(Long id) {
  this.id = id;
}

/**
 * @return the searchModel
 */
public JPASearchModel<? extends JPAModel> getSearchModel() {
  return searchModel;
}

/**
```

```
   * @param searchModel
   *          the searchModel to set
   */
  public void setSearchModel(JPASearchModel<? extends JPAModel> searchModel) {
    this.searchModel = searchModel;
  }
}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import org.hibernate.criterion.Criterion;

import edu.upm.dpsm.cims.core.jpa.search.projection.ProjectionContainer;

/**
 * @author Miguelino San Miguel
 */
public abstract class AbstractSearchCriterion implements ProjectionContainer {

  protected AbstractSearchCriterion() {}

  public abstract Criterion getCriteria() throws Exception;

  public AbstractSearchCriterion and(AbstractSearchCriterion so) {
    return (new SearchConjunction()).add(so).add(this);
  }

  public AbstractSearchCriterion or(AbstractSearchCriterion so) {
    return (new SearchDisjunction()).add(so).add(this);
  }

}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import org.hibernate.criterion.Criterion;

import edu.upm.dpsm.cims.core.jpa.search.projection.ProjectionContainer;

/**
 * @author Miguelino San Miguel
 */
public abstract class ConditionClause implements ProjectionContainer {

  public abstract Criterion getCriteria() throws Exception;

}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

/**
 * @author Miguelino San Miguel
 */
public enum ERestrictionType {
  EQ("eq", 1, "is equal to"),
  EQ_PROPERTY("eqProperty", 1, "is equal properties"),
```

```
  NE("ne", 1, "is not equal to"),
  NE_PROPERTY("neProperty", 1, "is not equal properties"),
  GT("gt", 1, "is greater than"),
  GT_PROPERTY("gt", 1, "is greater than"),
  GE("ge", 1, "is greater than or equal to"),
  GE_PROPERTY("ge", 1, "is greater than or equal to"),
  LT("lt", 1, "is less than"),
  LT_PROPERTY("lt", 1, "is less than"),
  LE("le", 1, "is less than or equal to"),
  LE_PROPERTY("le", 1, "is less than or equal to"),
  LIKE("like", 1, "is like"),
  IN("in", 1, "is in"),
  NULL("isNull", 1, "is null"),
  NOT_NULL("notNull", 1, "is not null"),
  LIKE_START("like", 2, "is like"),
  LIKE_END("like", 2, "is like"),
  ILIKE("ilike", 2, "ilike"),
  BETWEEN("between", 2, "is between");

  private String operator;
  private int operands;
  private String label;

  private ERestrictionType(String operator, int operands, String label) {
    this.operator = operator;
    this.operands = operands;
    this.label = label;
  }

  /**
   * @return the label
   */
  public String getLabel() {
    return label;
  }

  /**
   * @return the method
   */
  public String getOperator() {
    return operator;
  }

  public static ERestrictionType typeOf(String operator) throws Exception {
    for (ERestrictionType esr : ERestrictionType.values()) {
      if (esr.getOperator().equalsIgnoreCase(operator)) {
        return esr;
      }
    }
    throw new IllegalArgumentException(
        "No matching ESearchRestriction found for value " + operator);
  }
}
```

```
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import java.lang.reflect.Method;
import java.util.Collection;
import java.util.List;

import org.hibernate.criterion.Conjunction;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Disjunction;
import org.hibernate.criterion.MatchMode;
import org.hibernate.criterion.Restrictions;

import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class RestrictionsUtil {

  private static Method getMethod(String methodName, int operands) throws Exception {
    switch (operands) {
      case 2:
        if ("like".equalsIgnoreCase(methodName)) {
          return Restrictions.class.getMethod(methodName, String.class, String.class,
MatchMode.class);
        }
        return Restrictions.class.getMethod(methodName, String.class, Object.class,
Object.class);
      default:
        return Restrictions.class.getMethod(methodName, String.class, Object.class);
    }
  }

  public static Criterion getCriterion(SearchConjunction sc) throws Exception {
    Conjunction conjunction = Restrictions.conjunction();
    boolean empty = true;
    List<AbstractSearchCriterion> searchObjectList = sc.getConjunctionList();
    if (!ObjectUtils.isEmpty(searchObjectList)) {
      for (AbstractSearchCriterion _so : searchObjectList) {
        Criterion criterion = _so.getCriteria();
        if (criterion != null) {
          empty = false;
          conjunction.add(criterion);
        }
      }
    }
    return empty ? null : conjunction;
  }

  public static Criterion getCriterion(SearchDisjunction sc) throws Exception {
    Disjunction disjunction = Restrictions.disjunction();
    boolean empty = true;
```

```
    List<AbstractSearchCriterion> disjunctionList = sc.getDisjunctionList();
    if (!ObjectUtils.isEmpty(disjunctionList)) {
      for (AbstractSearchCriterion _so : disjunctionList) {
        Criterion criterion = _so.getCriteria();
        if (criterion != null) {
          empty = false;
          disjunction.add(criterion);
        }
      }
    }
    return empty ? null : disjunction;
  }

  public static Criterion getCriterion(AbstractSearchCriterion asc) throws Exception
{
    if (asc instanceof SearchConjunction) {
      return getCriterion((SearchConjunction) asc);
    } else if (asc instanceof SearchDisjunction) {
      return getCriterion((SearchDisjunction) asc);
    } else {
      return getCriterion((SearchCriterion) asc);
    }
  }

  public static Criterion getCriterion(String methodName, String propName, Object[]
values) throws Exception {

    if (!ObjectUtils.isEmpty(values) && !ObjectUtils.isEmpty(values[0])) {
      int operands = values.length;
      Method method = getMethod(methodName, operands);

      if (method != null) {

        switch (operands) {
          case 2:

            if (ObjectUtils.isEmpty(values[1])) {
              break;
            }

            return (Criterion) method.invoke(null, propName, values[0], values[1]);
          default:
            return (Criterion) method.invoke(null, propName, values[0]);
        }

      }

    }

    return null;
  }
```

```
  private static Criterion getCriterion(ERestrictionType type, SearchCriterion
criterion) throws Exception {
    Object values[] = criterion.getValues();
    Field field = criterion.getField();
    Field otherField = criterion.getOtherField();
    String propName = !ObjectUtils.isEmpty(field.getTable()) ? field.getTable() + "."
+ field.getName() : field
        .getName();
    if ((!ObjectUtils.isEmpty(values) && !ObjectUtils.isEmpty(values[0])) ||
otherField != null) {
      int operands = !ObjectUtils.isEmpty(values) ? values.length : 0;
      if (type != null) {
        switch (operands) {
          case 0:
            String otherPropName = !ObjectUtils.isEmpty(otherField.getTable()) ?
otherField.getTable() + "."
                + otherField.getName() : otherField.getName();
            getCriterion(type, propName, otherPropName);
            break;
          case 2:
            if (ObjectUtils.isEmpty(values[1])) {
              break;
            }
          default:
            return getCriterion(type, propName, values);
        }
      }
    }

    return null;
  }

  public static Criterion getCriterion(SearchCriterion criterion) throws Exception {
    if (criterion != null) {
      ERestrictionType esr = criterion.getOperator();
      if (esr != null && !ObjectUtils.isEmpty(criterion.getValues()))
        return getCriterion(esr, criterion);
    }
    return null;
  }

  public static Criterion getCriterion(ERestrictionType type, String propertyName,
String otherPropertyName)
      throws Exception {
    Criterion criterion = null;
    switch (type) {
      case EQ_PROPERTY:
        criterion = Restrictions.eqProperty(propertyName, otherPropertyName);
        break;
      case GT_PROPERTY:
        criterion = Restrictions.gtProperty(propertyName, otherPropertyName);
        break;
      case GE_PROPERTY:
```

```
        criterion = Restrictions.geProperty(propertyName, otherPropertyName);
        break;
      case LT_PROPERTY:
        criterion = Restrictions.ltProperty(propertyName, otherPropertyName);
        break;
      case LE_PROPERTY:
        criterion = Restrictions.leProperty(propertyName, otherPropertyName);
        break;
      case NE_PROPERTY:
        criterion = Restrictions.neProperty(propertyName, otherPropertyName);
        break;
      default:
    }
    return criterion;
  }

  public static Criterion getCriterion(ERestrictionType type, String propertyName,
Object[] values) throws Exception {
    Criterion criterion = null;
    switch (type) {
      case EQ:
        criterion = Restrictions.eq(propertyName, values[0]);
        break;
      case EQ_PROPERTY:
        criterion = Restrictions.eqProperty(propertyName, values[0].toString());
        break;
      case GT:
        criterion = Restrictions.gt(propertyName, values[0]);
        break;
      case GE:
        criterion = Restrictions.ge(propertyName, values[0]);
        break;
      case LT:
        criterion = Restrictions.lt(propertyName, values[0]);
        break;
      case LE:
        criterion = Restrictions.le(propertyName, values[0]);
        break;
      case NE:
        criterion = Restrictions.ne(propertyName, values[0]);
        break;
      case NE_PROPERTY:
        criterion = Restrictions.neProperty(propertyName, values[0].toString());
        break;
      case ILIKE:
        if (values.length == 1 || values[1] == null) {
          criterion = Restrictions.ilike(propertyName, values[0]);
        } else {
          criterion = Restrictions.ilike(propertyName, values[0].toString(),
(MatchMode) values[1]);
        }
        break;
      case LIKE:
```

```
        if (values.length == 1 || values[1] == null) {
          criterion = Restrictions.like(propertyName, values[0]);
        } else {
          criterion = Restrictions.like(propertyName, values[0].toString(),
(MatchMode) values[1]);
        }
        break;
      case IN:
        if (values[0] instanceof Collection<?>) {
          criterion = Restrictions.in(propertyName, (Collection<?>) values[0]);
        } else {
          criterion = Restrictions.in(propertyName, (Object[]) values[0]);
        }
        break;
      case NULL:
        criterion = Restrictions.isNull(propertyName);
        break;
      case NOT_NULL:
        criterion = Restrictions.isNotNull(propertyName);
        break;
      case BETWEEN:
        criterion = Restrictions.between(propertyName, values[0], values[1]);
        break;
      default:
    }
    return criterion;
  }
}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import java.util.List;

import org.hibernate.criterion.Criterion;

import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class SearchCondition extends ConditionClause {

  private SearchConjunction conjunctionList = new SearchConjunction();

  public SearchCondition() {
  }

  public SearchCondition propertyComparison(String fieldName, ERestrictionType
operator, String otherFieldName) throws Exception {
    this.conjunctionList.add(new SearchCriterion(new
Field().setName(fieldName).setAlias(fieldName), operator, new
Field().setName(otherFieldName)
        .setAlias(otherFieldName)));
```

```
    return this;
  }

  public <T extends Object> SearchCondition add(String fieldName, ERestrictionType
operator, T... values) throws Exception {
    return this.add(new Field().setName(fieldName).setAlias(fieldName), operator,
values);
  }

  public SearchCondition add(Field field, ERestrictionType operator, Object...
values) throws Exception {
    if (!ObjectUtils.isEmpty(values)) {
      this.conjunctionList.add(new SearchCriterion(field, operator, values));
    }
    return this;
  }

  public SearchCondition add(AbstractSearchCriterion criterion) throws Exception {
    this.conjunctionList.add(criterion);
    return this;
  }

  public static SearchDisjunction disjunction() {
    return new SearchDisjunction();
  }

  public static SearchConjunction conjunction() {
    return new SearchConjunction();
  }

  @Override
  public Criterion getCriteria() throws Exception {
    return this.conjunctionList.getCriteria();
  }

  @Override
  public String toString() {
    return this.conjunctionList.toString();
  }

  @Override
  public List<Field> getFields(List<Field> list) throws Exception {
    return this.conjunctionList.getFields(list);
  }
}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import java.util.ArrayList;
import java.util.List;

import org.hibernate.criterion.Conjunction;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Restrictions;
```

```
import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class SearchConjunction extends AbstractSearchCriterion {

	private List<AbstractSearchCriterion>  conjunctionList    = new
ArrayList<AbstractSearchCriterion>();

	public SearchConjunction add(String fieldName, ERestrictionType operator,
Object... values)
		throws Exception {
		   return this.add(new Field().setName(fieldName).setAlias(fieldName),
operator, values);
	}

	public SearchConjunction add(Field field, ERestrictionType operator, Object...
values)
		throws Exception {
		   if (!ObjectUtils.isEmpty(values)) {
			  this.conjunctionList.add(new SearchCriterion(field, operator,
values));
		   }
		   return this;
	}

	public SearchConjunction and(String fieldName, ERestrictionType operator,
Object... values)
		throws Exception {
		   return this.and(new Field().setName(fieldName).setAlias(fieldName),
operator, values);
	}

	public SearchConjunction and(Field field, ERestrictionType operator, Object...
values)
		throws Exception {
		   if (!ObjectUtils.isEmpty(values)) {
			  this.conjunctionList.add(new SearchCriterion(field, operator,
values));
		   }
		   return this;
	}

	public SearchConjunction add(AbstractSearchCriterion so) {
		   conjunctionList.add(so);
		   return this;
	}

	@Override
	public SearchConjunction and(AbstractSearchCriterion so) {
```

```
	conjunctionList.add(so);
	return this;
}

@Override
public Criterion getCriteria() throws Exception {
	Conjunction conjunction = Restrictions.conjunction();
	boolean empty = true;
	if (!ObjectUtils.isEmpty(conjunctionList)) {
		for (AbstractSearchCriterion _so : this.conjunctionList) {
			Criterion criterion = _so.getCriteria();
			if (criterion != null) {
				empty = false;
				conjunction.add(criterion);
			}
		}
	}
	return empty ? null : conjunction;
}

@Override
public String toString() {
	StringBuilder condition = new StringBuilder();
	if (!ObjectUtils.isEmpty(conjunctionList)) {
		condition.append("(");
		for (AbstractSearchCriterion _so : this.conjunctionList) {
			String critString = _so.toString();
			if (!ObjectUtils.isEmpty(critString)) {
				condition.append(critString + " and ");
			}
		}
		int size = condition.length();
		if (size > 4) {
			condition.delete(size - 4, size);
			condition.append(")");
			return condition.toString();
		}
	}
	return null;
}

/**
 * @return the conjunctionList
 */
protected List<AbstractSearchCriterion> getConjunctionList() {
	return conjunctionList;
}

@Override
public List<Field> getFields(List<Field> list) throws Exception {
	if (!this.conjunctionList.isEmpty()) {
		for (AbstractSearchCriterion _asc : this.conjunctionList) {
			_asc.getFields(list);
```

```
                }
            }
            return list;
        }
}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import java.util.List;

import org.hibernate.criterion.Criterion;

import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.jpa.search.projection.ProjectionContainer;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class SearchCriterion extends AbstractSearchCriterion implements
ProjectionContainer {
  private Field            field;
  private ERestrictionType operator;
  private Object[]         values;
  private Field            otherField;

  public SearchCriterion() {
  }

  public SearchCriterion(Field field, ERestrictionType operator, Field otherField) {
    this.field = field;
    this.operator = operator;
    this.otherField = otherField;
  }

  public SearchCriterion(Field field, ERestrictionType operator, Object... values) {
    this.field = field;
    this.operator = operator;
    this.values = values;
  }

  public SearchCriterion(String fieldName, ERestrictionType operator, Object...
values) {
    this(new Field().setAlias(fieldName).setName(fieldName), operator, values);
  }

  /**
   * @return the fieldName
   */
  public Field getField() {
    return field;
  }

  /**
```

```
 * @param fieldName
 *          the fieldName to set
 */
public void setField(Field field) {
  this.field = field;
}

/**
 * @return the operator
 */
public ERestrictionType getOperator() {
  return operator;
}

/**
 * @param operator
 *          the operator to set
 */
public void setOperator(ERestrictionType operator) {
  this.operator = operator;
}

/**
 * @return the values
 */
public Object[] getValues() {
  return values;
}

/**
 * @param values
 *          the values to set
 */
public void setValues(Object[] values) {
  this.values = values;
}

@Override
public Criterion getCriteria() throws Exception {
  return RestrictionsUtil.getCriterion(this);
}

@Override
public String toString() {
  if (!ObjectUtils.isEmpty(this.values) && !ObjectUtils.isEmpty(this.values[0])) {
    return this.field + " " + this.operator.getOperator() + " " + this.values[0];
  }
  return null;
}

@Override
public List<Field> getFields(List<Field> list) throws Exception {
  if (!list.contains(this.field)) {
```

```
      if (getCriteria() != null) {
        list.add(this.field);
      }
    }
    return list;
  }

  /**
   * @return the otherField
   */
  public Field getOtherField() {
    return otherField;
  }

  /**
   * @param otherField
   *          the otherField to set
   */
  public void setOtherField(Field otherField) {
    this.otherField = otherField;
  }

}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import java.util.ArrayList;
import java.util.List;

import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Disjunction;
import org.hibernate.criterion.Restrictions;

import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
public class SearchDisjunction extends AbstractSearchCriterion {

      private List<AbstractSearchCriterion>  disjunctionList    = new
ArrayList<AbstractSearchCriterion>();

      public SearchDisjunction add(AbstractSearchCriterion so) {
            disjunctionList.add(so);
            return this;
      }

      @Override
      public SearchDisjunction or(AbstractSearchCriterion so) {
            disjunctionList.add(so);
            return this;
      }
```

```
	public SearchDisjunction add(String fieldName, ERestrictionType operator,
Object... values)
	    throws Exception {
	       return this.add(new Field().setName(fieldName).setAlias(fieldName),
operator, values);
	}

	public SearchDisjunction add(Field field, ERestrictionType operator, Object...
values)
	    throws Exception {
	       if (!ObjectUtils.isEmpty(values)) {
	              this.disjunctionList.add(new SearchCriterion(field, operator,
values));
	       }
	       return this;
	}

	@Override
	public Criterion getCriteria() throws Exception {
	       Disjunction disjunction = Restrictions.disjunction();
	       boolean empty = true;
	       if (!ObjectUtils.isEmpty(disjunctionList)) {
	              for (AbstractSearchCriterion _so : this.disjunctionList) {
	                     Criterion criterion = _so.getCriteria();
	                     if (criterion != null) {
	                            empty = false;
	                            disjunction.add(criterion);
	                     }
	              }
	       }
	       return empty ? null : disjunction;
	}

	@Override
	public String toString() {
	       StringBuilder condition = new StringBuilder();
	       if (!ObjectUtils.isEmpty(disjunctionList)) {
	              condition.append("(");
	              for (AbstractSearchCriterion _so : this.disjunctionList) {
	                     String critString = _so.toString();
	                     if (!ObjectUtils.isEmpty(critString)) {
	                            condition.append(critString + " or ");
	                     }
	              }
	              int size = condition.length();
	              if (size > 3) {
	                     condition.delete(size - 3, size);
	                     condition.append(")");
	                     return condition.toString();
	              }
	       }
	       return null;
```

```
	}

	/**
	 * @return the disjunctionList
	 */
	protected List<AbstractSearchCriterion> getDisjunctionList() {
		return disjunctionList;
	}

	@Override
	public List<Field> getFields(List<Field> list) throws Exception {
		if (!this.disjunctionList.isEmpty()) {
			for (AbstractSearchCriterion _asc : this.disjunctionList) {
				_asc.getFields(list);
			}
		}
		return list;
	}
}
package edu.upm.dpsm.cims.core.jpa.search.criterion;

import java.util.List;

import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Example;

import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.app.model.Model;

/**
 * @author Miguelino San Miguel
 */
public class SimpleSearchCondition extends ConditionClause {

  private Model model;

  public SimpleSearchCondition() {}

  public SimpleSearchCondition(Model t) {
    this.model = t;
  }

  public Model getModel() {
    return model;
  }

  public SimpleSearchCondition setModel(Model t) {
    this.model = t;
    return this;
  }

  @Override
  public Criterion getCriteria() throws Exception {
```

```
    return Example.create(model).enableLike().ignoreCase().excludeZeroes();
  }

  @Override
  public List<Field> getFields(List<Field> list) throws Exception {
    return list;
  }
}
package edu.upm.dpsm.cims.core.jpa.search.order;

import org.hibernate.criterion.Order;



/**
 * @author Miguelino San Miguel
 */
public enum EOrderType {
  ASC, DESC;
  public Order getOrder(String propName) throws Exception {
    return OrderUtil.getOrder(this, propName);
  }
}
package edu.upm.dpsm.cims.core.jpa.search.order;

import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import edu.upm.dpsm.cims.core.app.field.Field;
import edu.upm.dpsm.cims.core.jpa.search.projection.ProjectionContainer;

/**
 * @author Miguelino San Miguel
 */
public class JPAOrderClause implements ProjectionContainer {
  private final Set<Order> orderSet = new HashSet<Order>();
  private final Set<Field> fieldSet = new HashSet<Field>();

  public JPAOrderClause addOrder(Field field, EOrderType type) throws Exception {
    orderSet.add(new Order(field, type));
    return this;
  }

  public Set<Order> getOrderSet() throws Exception {
    return this.orderSet;
  }

  @Override
  public List<Field> getFields(List<Field> list) throws Exception {
    fieldSet.clear();
    Order[] oArr = (Order[]) orderSet.toArray();
```

```
    for (Order o : oArr) {
      Field oField = o.getOrderBy();
      fieldSet.add(oField);
    }
    return Arrays.asList(fieldSet.toArray(new Field[] {}));
  }
}
package edu.upm.dpsm.cims.core.jpa.search.order;

import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;

import edu.upm.dpsm.cims.core.app.field.Field;

/**
 * @author Miguelino San Miguel
 */
public class Order {
	protected EOrderType	orderType;
	protected Field		orderBy;

	public Order() {
	}

	public Order(Field field, EOrderType type) {
		this.orderBy = field;
		this.orderType = type;
	}

	public EOrderType getOrderType() {
		return orderType;
	}

	public void setOrderType(EOrderType order) {
		this.orderType = order;
	}

	public Field getOrderBy() {
		return orderBy;
	}

	public void setOrderBy(Field orderBy) {
		this.orderBy = orderBy;
	}

	@Override
	public int hashCode() {
		return new HashCodeBuilder().append(this.orderBy).toHashCode();
	}

	@Override
	public boolean equals(Object obj) {
		return EqualsBuilder.reflectionEquals(this, obj, false);
```

```
	}
}
package edu.upm.dpsm.cims.core.jpa.search.order;

import org.hibernate.criterion.Order;



/**
 * @author Miguelino San Miguel
 */
public class OrderUtil {

  public static Order getOrder(EOrderType orderBy, String propName)
      throws Exception {
    if (EOrderType.DESC.equals(orderBy)) {
      return Order.desc(propName);
    }
    return Order.asc(propName);

  }
}
package edu.upm.dpsm.cims.core.jpa.search.pagination;

/**
 * @author Miguelino San Miguel
 */
public class Pagination {

	private Long   pageSize;
	private Long   page;
	private Long   totalSize;
	private Long   totalPage;

	private Long   firstRow;
	private Long   lastRow;

	private boolean    enable = true;

	/**
	 * @return the pageSize
	 */
	public Long getPageSize() {
		return pageSize;
	}

	/**
	 * @param pageSize
	 *          the pageSize to set
	 */
	public void setPageSize(Long pageSize) {
		this.pageSize = pageSize;
	}
```

```
/**
 * @return the page
 */
public Long getPage() {
	return page;
}

/**
 * @param page
 *          the page to set
 */
public void setPage(Long page) {
	this.page = page;
}

/**
 * @return the totalSize
 */
public Long getTotalSize() {
	return totalSize;
}

/**
 * @param totalSize
 *          the totalSize to set
 */
public void setTotalSize(Long totalSize) {
	this.totalSize = totalSize;
}

public Long getTotalPage() {
	return totalPage;
}

public void setTotalPage(Long totalPage) {
	this.totalPage = totalPage;
}

public Long getFirstRow() {
	return firstRow;
}

public void setFirstRow(Long firstRow) {
	this.firstRow = firstRow;
}

public Long getLastRow() {
	return lastRow;
}

public void setLastRow(Long lastRow) {
	this.lastRow = lastRow;
```

```
	}

	public boolean isEnable() {
		return enable;
	}

	public void setEnable(boolean enable) {
		this.enable = enable;
	}
}
package edu.upm.dpsm.cims.core.jpa.search.projection;

import java.util.List;

import edu.upm.dpsm.cims.core.app.field.Field;

/**
 * @author Miguelino San Miguel
 */
public interface ProjectionContainer {
	public List<Field> getFields(List<Field> list) throws Exception;
}
package edu.upm.dpsm.cims.core.jpa.search.projection;

import java.util.List;

import edu.upm.dpsm.cims.core.app.field.Field;

/**
 * @author Miguelino San Miguel
 */
public class ProjectionMap implements ProjectionContainer {
  @Override
  public List<Field> getFields(List<Field> list) throws Exception {
    // TODO
    return null;
  }

}
package edu.upm.dpsm.cims.core.jpa.search.projection;

/**
 * @author Miguelino San Miguel
 */
public class ProjectionUtil {

}
package edu.upm.dpsm.cims.core.jpa.search;

import java.util.ArrayList;
import java.util.List;

import edu.upm.dpsm.cims.core.app.field.Field;
```

```
import edu.upm.dpsm.cims.core.jpa.model.JPAModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ConditionClause;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SimpleSearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.order.EOrderType;
import edu.upm.dpsm.cims.core.jpa.search.order.JPAOrderClause;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;

/**
 * @author Miguelino San Miguel
 *
 */
public class JPASearchModel<T extends JPAModel> {
  protected ConditionClause conditionClause;
  protected JPAOrderClause  order;
  protected Pagination      pagination;
  private String            tableName;

  protected JPASearchModel(T model) throws Exception {
    if (model != null) {
      this.createSearchCriteria(model);
      this.createSearchOrder(model);
    }
  }

  protected void createSearchCriteria(T model) throws Exception {
    setSearch(new SimpleSearchCondition().setModel(model));
  }

  protected void createSearchOrder(T model) throws Exception {
    setOrderClause(new JPAOrderClause().addOrder(new
Field().setName(T.ID).setAlias(T.ID), EOrderType.ASC));
  }

  protected JPASearchModel<T> setSearch(ConditionClause conditionClause) {
    this.conditionClause = conditionClause;
    return this;
  }

  public ConditionClause getWhere() {
    return this.conditionClause;
  }

  public JPAOrderClause getOrderClause() {
    return order;
  }

  public void setOrderClause(JPAOrderClause order) {
    this.order = order;
  }

  public Pagination getPagination() {
    return pagination;
```

```
  }

  public String getTableName() {
    return tableName;
  }

  public void setTableName(String tableName) {
    this.tableName = tableName;
    if (tableName != null && !tableName.isEmpty()) {
      if (this.conditionClause instanceof SearchCondition) {
        List<Field> fields = new ArrayList<Field>();
        try {
          ((SearchCondition) this.conditionClause).getFields(fields);
          for (Field f : fields) {
            f.setTable(tableName);
          }
        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }

      }
    }
  }

}
package edu.upm.dpsm.cims.core.jpa.service;

import java.util.List;

import javax.annotation.PostConstruct;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.core.jpa.model.JPAModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorSupply;

/**
 * @author Miguelino San Miguel
 *
 */
public abstract class JPAService<T extends JPAModel> implements BusinessService<T> {

  protected DAO<T> dao;

  @PostConstruct
  public void init() {
    this.dao = this.getDAO();
  }

  public abstract DAO<T> getDAO();

  public T get(T t) throws Exception {
```

```
    return this.dao.get(t);
  }

  public List<T> saveAll(List<T> tList) throws Exception {
    return this.dao.saveAll(tList);
  }

  @MonitorSupply
  public T save(T t) throws Exception {
    t.setId(null);
    return this.dao.save(t);
  }

  public T delete(T t) throws Exception {
    return this.dao.delete(t);
  }

  public List<T> deleteAll(List<T> t) throws Exception {
    return this.dao.deleteAll(t);
  }

  @MonitorSupply
  public T update(T t) throws Exception {
    return this.dao.update(t);
  }

  public List<T> search(T t) throws Exception {
    return this.dao.search(t);
  }

  public Long getTotalSize(T t) throws Exception {
    return dao.getTotalSize(t);
  }

}
package edu.upm.dpsm.cims.core.jpa.util;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;

/**
 * @author Miguelino San Miguel
 */
public class HibernateUtil {

  @Autowired
  private SessionFactory sessionFactory;

  private final ThreadLocal<Session> localThread = new ThreadLocal<Session>();

  /**
```

```
   * @return the sessionFactory
   */
  public SessionFactory getSessionFactory() {
    return sessionFactory;
  }

  public Session getSession() throws HibernateException {
    Session s = localThread.get();
    // Open a new Session, if this thread has none yet
    if (!isOpenSession(s)) {
      // NOTE: Do not use openSession since
      // framework is dependent on Transactional management
      // and on transaction begin, it always call openSession which
      // creates new session, so to use that same session
      // must use getCurrentSession in providing the DAO's
      s = sessionFactory.getCurrentSession();
      // Store it in the ThreadLocal variable
      localThread.set(s);
    }
    return s;
  }

  public void closeSession() throws HibernateException {
    Session s = localThread.get();
    if (isOpenSession(s)) {
      s.close();
    }
    localThread.set(null);
  }

  private boolean isOpenSession(Session s) {
    return s != null && s.isOpen();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.message;

/**
 * EMessages
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum EMessages implements IResourceMessage {
```

```
  USER_ACCOUNT_UPDATE_SUCCESS("Success.userAccount.update"),

  USER_ACCOUNT_DELETE_SUCCESS("Success.userAccount.delete"),

  CHEMICAL_DEFINITION_CREATE_SUCCESS("Success.chemicalDefinition.add"),
  CHEMICAL_DEFINITION_CREATE_FAIL("Fail.chemicalDefinition.add"),

  CHEMICAL_DEFINITION_UPDATE_SUCCESS("Success.chemicalDefinition.update"),
  CHEMICAL_DEFINITION_UPDATE_FAIL("Fail.chemicalDefinition.update"),

  CHEMICAL_ITEM_ADD_SUCCESS("Success.chemicalItem.add"),
  CHEMICAL_ITEM_ADD_FAIL("Fail.chemicalItem.add"),

  LABWARE_DEFINITION_CREATE_SUCCESS("Success.labwareDefinition.add"),
  LABWARE_DEFINITION_CREATE_FAIL("Fail.labwareDefinition.add"),

  LABWARE_DEFINITION_UPDATE_SUCCESS("Success.labwareDefinition.update"),
  LABWARE_DEFINITION_UPDATE_FAIL("Fail.labwareDefinition.update"),

  LABWARE_ITEM_ADD_SUCCESS("Success.labwareItem.add"),
  LABWARE_ITEM_ADD_FAIL("Fail.labwareItem.add"),

  REPLENISHMENT_REQUEST_CREATE_SUCCESS("Success.replenishmentRequest.create"),
  REPAIR_REQUEST_CREATE_SUCCESS("Success.repairRequest.create"),
  CONDEMNATION_REQUEST_CREATE_SUCCESS("Success.condemnationRequest.create"),

  APPROVED_REQUEST_SUCCESS("Success.maintenanceRequest.approved"),

  REJECTED_REQUEST_SUCCESS("Success.maintenanceRequest.rejected"),

  ACCOUNTABILITY_CREATE_SUCCESS("Success.accountability.add"),

  ACCOUNTABILITY_CLEAR_SUCCESS("Success.accountability.clear"),

  ACCOUNTABILITY_UNCLEAR_SUCCESS("Success.accountability.unclear"),
  ;

  private String messageKey;

  private EMessages(String key) {
    this.messageKey = key;
  }

  @Override
  public String getMessageKey() {
    return this.messageKey;
  }

}
/**
 * CIMS
 *
 * License :
```

```
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.message;

/**
 * IResourceMessage
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface IResourceMessage {
  public String getMessageKey();
}
/**
 * CIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.message;

import org.springframework.context.MessageSource;

/**
 * MessagesUtils
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class MessagesUtils {

  private static MessageSource messageSource;

  public static void setMessageSource(MessageSource messageSource) {
    if (MessagesUtils.messageSource == null) {
      MessagesUtils.messageSource = messageSource;
    }
  }

  public static String getMessage(IResourceMessage message) {
    return messageSource.getMessage(message.getMessageKey(), null, null, null);
  }

  public static String getMessage(String code, Object[] arguments) {
    return messageSource.getMessage(code, arguments, null, null);
  }
}
package edu.upm.dpsm.cims.core.util;
```

```
import java.sql.Timestamp;
import java.util.Calendar;
import java.util.TimeZone;

import org.joda.time.DateTime;
import org.joda.time.DateTimeZone;

import edu.upm.dpsm.cims.core.ProjectConstants;

/**
 * @author Miguelino San Miguel
 */
public class DateUtil {

  public static Timestamp getCurrentTime() {
    return new Timestamp(DateTime.now().getMillis());
  }

  public static Timestamp getTodayStartTime() {
    Calendar cal = Calendar.getInstance(TimeZone.getDefault());
    cal.set(Calendar.HOUR_OF_DAY, 0);
    cal.set(Calendar.MINUTE, 0);
    cal.set(Calendar.SECOND, 0);
    cal.set(Calendar.MILLISECOND, 0);
    return new Timestamp(cal.getTimeInMillis());
  }

  public static Timestamp getTodayEndTime() {
    Calendar cal = Calendar.getInstance(TimeZone.getDefault());
    cal.set(Calendar.HOUR_OF_DAY, 23);
    cal.set(Calendar.MINUTE, 59);
    cal.set(Calendar.SECOND, 59);
    cal.set(Calendar.MILLISECOND, 999);
    return new Timestamp(cal.getTimeInMillis());
  }

  public static String toString(Timestamp timestamp) {
    return toString(timestamp, ProjectConstants.DATE_FORMAT,
DateTimeZone.getDefault());
  }

  public static String toString(Timestamp timestamp, String format, DateTimeZone
timezone) {
    return new DateTime(timestamp.getTime(), timezone).toString(format);
  }
}
package edu.upm.dpsm.cims.core.util;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import org.springframework.beans.BeanUtils;
```

```
public class DTOUtil {

	public static <T, E> E copy(Class<E> clazz, T source,
			String... ignoreProperties) {

		try {
			E e = clazz.newInstance();

			BeanUtils.copyProperties(source, e, ignoreProperties);

			return e;
		} catch (InstantiationException e1) {
			e1.printStackTrace();
		} catch (IllegalAccessException e1) {
			e1.printStackTrace();
		} catch (Exception e) {
			e.printStackTrace();
		}
		return null;

	}

	public static <T, E> List<E> copy(Class<E> clazz, List<T> sourceList,
			String... ignoreProperties) {

		try {
			List<E> newList = new ArrayList<E>();

			for (T sourceItem : sourceList) {
				E e = clazz.newInstance();

				BeanUtils.copyProperties(sourceItem, e, ignoreProperties);
				newList.add(e);
			}
			return newList;
		} catch (InstantiationException e1) {
			e1.printStackTrace();
		} catch (IllegalAccessException e1) {
			e1.printStackTrace();
		} catch (Exception e) {
			e.printStackTrace();
		}

		return Collections.emptyList();

	}

}
/**
 * CIMS
 *
 * License :
```

```
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
package edu.upm.dpsm.cims.core.util;

import javax.servlet.http.HttpSession;

/**
 * HttpSessionUtil
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class HttpSessionUtils {

  public static boolean exists(HttpSession session, String key) {
    return session != null && session.getAttribute(key) != null;
  }
}
package edu.upm.dpsm.cims.core.util;

import java.util.Locale;

import org.apache.log4j.Logger;
import org.springframework.context.MessageSource;

import edu.upm.dpsm.cims.core.AppContext;

public class MessageUtil {

	private static MessageSource messageSource;
	private static final Logger logger = Logger.getLogger(MessageUtil.class);

	static {
		messageSource = AppContext.getBean(MessageSource.class);
	}

	public static String getMessage(String code) {
		try {
			Locale locale = Locale.getDefault();
			String customMessage = messageSource.getMessage(code, null, null,
					locale);
			if (!ObjectUtils.isEmpty(customMessage)) {
				return customMessage;

			}
		} catch (Exception e) {
			logger.error(e.getMessage());
		}
		return null;
	}
```

```
}
package edu.upm.dpsm.cims.core.util;

import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.List;
import java.util.Map;
import java.util.Set;

/**
 * @author Miguelino San Miguel
 */
public class ObjectUtils {

  public static boolean isEmpty(Object a) {
    if (a == null) {
      return true;
    }
    if (a instanceof String) {
      return isEmpty((String) a);
    } else if (a instanceof Number) {
      return isEmpty((Number) a);
    }
    else if (a instanceof List) {
      return isEmpty((List<?>) a);
    }
    else if (a instanceof Set) {
      return isEmpty((Set<?>) a);
    } else if (a instanceof Map) {
      return isEmpty((Map<?, ?>) a);
    }
    return false;
  }

  public static boolean isEmpty(Number a) {
    return a == null || a.doubleValue() == 0.0;
  }

  public static boolean isEmpty(Long a) {
    return a == null || a.longValue() == 0L;
  }

  public static boolean isEmpty(Integer a) {
    return a == null || a.intValue() == 0;
  }

  public static boolean isEmpty(BigInteger a) {
    return a == null || a.intValue() == 0;
  }

  public static boolean isEmpty(BigDecimal a) {
    return a == null || a.doubleValue() == 0.0;
  }
```

```
  public static boolean isEmpty(Double a) {
    return a == null || a.doubleValue() == 0.0;
  }

  public static boolean isEmpty(String a) {
    return a == null || a.isEmpty();
  }

  public static boolean isEmpty(List<?> list) {
    return list == null || list.isEmpty();
  }

  public static boolean isEmpty(Set<?> set) {
    return set == null || set.isEmpty();
  }

  public static boolean isEmpty(Map<?, ?> map) {
    return map == null || map.isEmpty();
  }

  public static boolean isEmpty(Object[] objs) {
    return objs == null || objs.length == 0;
  }
}
package edu.upm.dpsm.cims.core.util;

import javax.servlet.http.HttpServletRequest;

/**
 * @author Miguelino San Miguel
 */
public class RequestUtil {
  private final static ThreadLocal<HttpServletRequest> threadLocal = new
ThreadLocal<HttpServletRequest>();

  public static HttpServletRequest getRequest() throws Exception {
    return RequestUtil.threadLocal.get();
  }

  public static void setRequest(HttpServletRequest request)
      throws Exception {
    RequestUtil.threadLocal.set(request);
  }
}
package edu.upm.dpsm.cims.core.util;


public class UserAccountUtil {

  private final static ThreadLocal<Long> threadLocal = new ThreadLocal<Long>();

  public static Long getUserAccountId() {
```

```
    return threadLocal.get();
  }

  public static void setUserAccountId(Long userAccountId) {
    threadLocal.set(userAccountId);
  }
}
package edu.upm.dpsm.cims.core;

import org.apache.log4j.Logger;
import org.springframework.beans.BeansException;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;

/**
 * @author Miguelino San Miguel
 */
public class AppContext implements ApplicationContextAware {
  private static final Logger       logger = Logger.getLogger(AppContext.class);
  private static ApplicationContext ctx;

  @Override
  public void setApplicationContext(final ApplicationContext ctx) throws
BeansException {
    if (AppContext.ctx == null) {
      AppContext.ctx = ctx;
      logger.info("Initializes AppContext that provides Spring ApplicationContext");
    }
  }

  public static Object getBean(String bean) {
    return AppContext.ctx.getBean(bean);
  }

  public static <T> T getBean(Class<T> bean) {
    return AppContext.ctx.getBean(bean);
  }
}
package edu.upm.dpsm.cims.core;

/**
 * @author Miguelino San Miguel
 */
public class PageConstant {
  public static final String TOTAL_ROWS = "TOTAL_ROWS";
  public static final String TOTAL_PAGE = "TOTAL_PAGE";
  public static final String CURRENT_PAGE = "CURRENT_PAGE";
  public static final String ROW_PER_PAGE = "ROW_PER_PAGE";

  public static final long PAGE_SIZE = 15;

}
package edu.upm.dpsm.cims.core;
```

```
/**
 * @author Miguelino San Miguel
 */
public class ProjectConstants {
  public static final String CONTEXT_URL = "LIMS";
  // DATA FORMAT
  public static final String INTEGER_FORMAT = "";
  public static final String DECIMAL_FORMAT = "";
  public static final String MONEY_FORMAT = "";
  public static final String DATE_FORMAT = "yyyy/MM/dd";
  public static final String DATE_TIME_FORMAT = "yyyy/MM/dd HH:mm:ss";

  // REGEX
  public static final String PATTERN_ALPHA_NUMERIC = "^[A-Za-z]+[A-Za-z0-9]*$";
  public static final String PATTERN_ALPHA_NUMERIC_SEPARATOR = "^[A-Za-z]+[A-Za-z_0-
9]*$";
  public static final String PATTERN_NAME = "^[A-Za-z]+[A-Za-z\\s]*$";

  // MESSAGE FLAGS
  public static final String SUCCESS_MESSAGE = "successmessage";
  public static final String ERROR_MESSAGE = "errormessage";

  //
  public static final String USER_ACCOUNT_ID_SESSION = "UserAccountSession";

  //
  public static final String BASE_URL = "BASE_URL";

  public static final String TEMP_FOLDER_PATH = "src/tempfiles/";

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.department.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.department.model.Department;
import edu.upm.dpsm.cims.department.service.DepartmentService;
```

```
/**
 * GetDeparmentListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetDeparmentListController {

  @Autowired
  private DepartmentService departmentService;

  @RequestMapping(value = "department/list", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(HttpServletRequest request) throws Exception {
    return departmentService.search(new Department());
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.department.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.department.dao.DepartmentDAO;
import edu.upm.dpsm.cims.department.model.Department;

/**
 * DepartmentDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class DepartmentDAOImpl extends JPADao<Department> implements DepartmentDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
```

```
 */
package edu.upm.dpsm.cims.department.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.department.model.Department;

/**
 * DepartmentDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface DepartmentDAO extends DAO<Department> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.department.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * Department
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "department")
public class Department extends JPALookupModel {
  @Column(name = "code")
  private String code;
  @Column(name = "name")
  private String name;

  public String getCode() {
    return code;
  }

  public void setCode(String code) {
    this.code = code;
  }
```

```
  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.department.service.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.department.dao.DepartmentDAO;
import edu.upm.dpsm.cims.department.model.Department;
import edu.upm.dpsm.cims.department.service.DepartmentService;

/**
 * DepartmentServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class DepartmentServiceImpl extends JPAService<Department> implements
DepartmentService {

  @Autowired
  private DepartmentDAO dao;
  @Override
  public DAO<Department> getDAO() {
    return dao;
  }

}
/**
 * LIMS
 *
 * License :
 *
```

```
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.department.service;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.department.model.Department;

/**
 * DepartmentService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface DepartmentService extends BusinessService<Department> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.itemstatus.model;

/**
 * EItemStatus
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum EItemStatus {
  USABLE(1,"usable"),
  REPLENISHMENT(2,"replenishment"),
  CONDEMNATION(3,"condemnation"),
  REPAIR(4,"repair")
  ;

  private int status;
  private String name;
  private EItemStatus(int status, String name){
    this.status = status;
    this.name = name;
  }

  public int asInt(){
    return this.status;
  }

  public String getName() {
    return name;
```

```
  }


}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.laboratoryunit.model.LaboratoryUnit;
import edu.upm.dpsm.cims.laboratoryunit.service.LaboratoryUnitService;

/**
 * GetLaboratoryUnitListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetLaboratoryUnitListController {

  @Autowired
  private LaboratoryUnitService laboratoryUnitService;

  @RequestMapping(value = "laboratory-unit/list", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(HttpServletRequest request) throws Exception {
    return laboratoryUnitService.search(new LaboratoryUnit());
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.dao.impl;

import org.springframework.stereotype.Repository;
```

```
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.laboratoryunit.dao.LaboratoryUnitDAO;
import edu.upm.dpsm.cims.laboratoryunit.model.LaboratoryUnit;

/**
 * LaboratoryUnitDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class LaboratoryUnitDAOImpl extends JPADao<LaboratoryUnit> implements
LaboratoryUnitDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.laboratoryunit.model.LaboratoryUnit;

/**
 * LaboratoryUnitDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface LaboratoryUnitDAO extends DAO<LaboratoryUnit> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.model;

/**
 * ELaboratoryUnit
 *
```

```
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum ELaboratoryUnit {
  PHYSICS(1),
  CHEMISTRY(2),
  BIOLOGY(3),
  MATH_CS(4),

  ;
  private long laboratoryUnitId;

  private ELaboratoryUnit(long laboratoryUnitId) {
    this.laboratoryUnitId = laboratoryUnitId;
  }

  public static ELaboratoryUnit valueOf(long id) {
    for (ELaboratoryUnit unit : ELaboratoryUnit.values()) {
      if (unit.getId().equals(id)) {
        return unit;
      }
    }
    throw new IllegalArgumentException("No matching ELaboratoryUnit for the given
long value");
  }

  public Long getId() {
    return this.laboratoryUnitId;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.model;

import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * LaboratoryUnit
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
```

```
@Table(name = "laboratory_unit")
public class LaboratoryUnit extends JPALookupModel {
  private String name;

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.service.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.laboratoryunit.dao.LaboratoryUnitDAO;
import edu.upm.dpsm.cims.laboratoryunit.model.LaboratoryUnit;
import edu.upm.dpsm.cims.laboratoryunit.service.LaboratoryUnitService;

/**
 * LaboratoryUnitServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class LaboratoryUnitServiceImpl extends JPAService<LaboratoryUnit> implements
LaboratoryUnitService {

  @Autowired
  private LaboratoryUnitDAO dao;

  @Override
  public DAO<LaboratoryUnit> getDAO() {
    return dao;
  }

}
```

```
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 31, 2014
 */
package edu.upm.dpsm.cims.laboratoryunit.service;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.laboratoryunit.model.LaboratoryUnit;

/**
 * LaboratoryUnitService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface LaboratoryUnitService extends BusinessService<LaboratoryUnit> {

}
package edu.upm.dpsm.cims.labstatus.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.labstatus.model.LabwareStatus;

/**
 * @author Miguelino San Miguel
 *
 */
public interface LabwareStatusDAO extends DAO<LabwareStatus> {

}
package edu.upm.dpsm.cims.labstatus.dao;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.labstatus.model.LabwareStatus;

/**
 * @author Miguelino San Miguel
 *
 *
 */
@Repository
@Transactional(rollbackFor = Exception.class)
public class LabwareStatusDAOImpl extends JPADao<LabwareStatus> implements
LabwareStatusDAO {

}
```

```
package edu.upm.dpsm.cims.labstatus.model;

/**
 * @author Miguelino San Miguel
 *
 */
public enum ELabwareStatus {
  OK(1, "ok"),
  STAINED(1, "stained"),
  BROKEN(1, "broken");

  private String name;
  private int    statusId;

  private ELabwareStatus(int id, String name) {
    this.statusId = id;
    this.name = name;
  }

  public boolean equals(String name) {
    return this.name.equals(name);
  }

  public boolean equals(Integer id) {
    return id != null && this.statusId == id.intValue();
  }

  public int asInt() {
    return this.statusId;
  }
}
package edu.upm.dpsm.cims.labstatus.model;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * @author Miguelino San Miguel
 *
 */
@Entity
@Table(name = "labware_status")
@Access(AccessType.FIELD)
public class LabwareStatus extends JPALookupModel {
       @Column(name = "name", updatable = false)
       private String     name;

       public String getName() {
```

```
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

}
package edu.upm.dpsm.cims.labstatus.service;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.labstatus.model.LabwareStatus;

/**
 * @author Miguelino San Miguel
 *
 */
public interface LabwareStatusService extends BusinessService<LabwareStatus> {

}
package edu.upm.dpsm.cims.labstatus.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.labstatus.dao.LabwareStatusDAO;
import edu.upm.dpsm.cims.labstatus.model.LabwareStatus;

/**
 * @author Miguelino San Miguel
 *
 */
@Repository
@Transactional
public class LabwareStatusServiceImpl extends JPAService<LabwareStatus> implements
LabwareStatusService {

	@Autowired
	private LabwareStatusDAO  dao;

	@Override
	public DAO<LabwareStatus> getDAO() {
		return dao;
	}

}
/**
 * LIMS
 *
 * License :
```

```
 *
 * author: Miguelino San Miguel  Mar 24, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.aop.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;

/**
 * MonitorRequest
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Retention(RetentionPolicy.RUNTIME)
public @interface MonitorRequest {
  public abstract EItemStatus status();
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.aop.annotation;

import java.lang.annotation.Documented;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * MonitorSupply
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Retention(RetentionPolicy.RUNTIME)
@Target({ java.lang.annotation.ElementType.METHOD,
java.lang.annotation.ElementType.TYPE })
@Inherited
@Documented
public @interface MonitorSupply {

}
/**
 * LIMS
```

```
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 24, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.aop.aspect;

import java.util.List;

import javax.validation.ValidationException;

import org.apache.log4j.Logger;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchExistingRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * MonitorRequestAspect
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Aspect
@Component
public class MonitorRequestAspect {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  private final static Logger       logger =
Logger.getLogger(MonitorRequestAspect.class);


@Pointcut("execution(@edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequ
est public * *(..))")
  public void controllerMethod() {
    logger.debug("Monitoring supply update is triggered in a transactional method");
  }

  @Before("controllerMethod() && @annotation( requestAnnotation )")
  public void afterTransactionalMethod(JoinPoint joinPoint, MonitorRequest
requestAnnotation) throws Exception {
```

```
    Object[] methodParameters = joinPoint.getArgs();
    if (methodParameters[0] instanceof Maintainable) {

      EItemStatus status = requestAnnotation.status();
      Maintainable labwareItem = (Maintainable) methodParameters[0];
      MaintenanceRequest request = new MaintenanceRequest();
      request.setDefinitionId(labwareItem.getLabwareId());
      request.setItemId(labwareItem.getId());
      request.setItemStatus(status.asInt());
      SearchExistingRequest ser = new SearchExistingRequest(request);
      request.setSearchModel(ser);
      List<MaintenanceRequest> list = maintenanceRequestService.search(request);
      if (!ObjectUtils.isEmpty(list)) {
        throw new ValidationException("The labware item is already in request for " +
status.getName());
      }
    }

  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.aop.aspect;

import javax.persistence.Table;

import org.apache.log4j.Logger;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * MonitorSupplyAspect
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Aspect
@Component
public class MonitorSupplyAspect {
  @Autowired
```

```
  private MaintenanceRequestService maintenanceRequestService;

  private final static Logger       logger =
Logger.getLogger(MonitorSupplyAspect.class);

  public MonitorSupplyAspect() {
  }


@Pointcut("execution(@edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorSupp
ly public * *(..))")
  public void transactionalMethod() {
    logger.debug("Monitoring supply update is triggered in a transactional method");
  }

  @AfterReturning("transactionalMethod()")
  public void afterTransactionalMethod(JoinPoint joinPoint) {
    Object[] methodParameters = joinPoint.getArgs();
    if (methodParameters[0] instanceof MonitorItem) {
      MonitorItem monitoredItem = (MonitorItem) methodParameters[0];
      if ((monitoredItem.getMinimum() != null)) {
        try {
          Table tableAnnotation =
monitoredItem.getClass().getAnnotation(Table.class);
          String moduleName = tableAnnotation.name();
          if ((monitoredItem.getTotalQuantity().longValue() <
monitoredItem.getMinimum().longValue())) {

            logger.debug("Monitoring supply checkpoint is triggered, minimum amount
is reached for module entity "
                + moduleName);
            maintenanceRequestService.createAlertForReplenishment(moduleName,
monitoredItem);
          } else {
            maintenanceRequestService.deleteAlertForReplenishment(moduleName,
monitoredItem);
          }
        } catch (Exception e) {
          logger.error(e);
        }

      }
    }
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.aop.model;
```

```
/**
 * MonitorItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface MonitorItem{
  public Long getId();
  public String getName();
  public Long getTotalQuantity();
  public Integer getMinimum();
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateLabwareItemCondemnationRequestController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/condemnation-request/create", method =
RequestMethod.POST)
  public @ResponseBody
```

```
  Object doAddLabwareItem(@RequestBody MaintenanceRequest maintenanceRequest,
HttpServletRequest request)
      throws Exception {

    maintenanceRequestService.createCondemnationRequest(maintenanceRequest);

    return
JSONMapUtils.createJSONSuccess(EMessages.CONDEMNATION_REQUEST_CREATE_SUCCESS).getMap(
);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateLabwareItemRepairRequestController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/repair-request/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody MaintenanceRequest maintenanceRequest,
HttpServletRequest request)
      throws Exception {
```

```
    maintenanceRequestService.createRepairRequest(maintenanceRequest);

    return
JSONMapUtils.createJSONSuccess(EMessages.REPAIR_REQUEST_CREATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * GetReplenishmentAlertsListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetAlertsListController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/alert/list", method =
RequestMethod.POST)
  public @ResponseBody
  Object getAlertList(@RequestBody MaintenanceRequest mRequest, HttpServletRequest
request) throws Exception {

    List<MaintenanceRequest> alertsList =
maintenanceRequestService.searchAlerts(mRequest);

    return DTOUtil.copy(MaintenanceRequest.class, alertsList, new String[] {
MaintenanceRequest.CREATED_BY,
```

```
        MaintenanceRequest.UPDATED_BY });
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 6, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * GetPendingRequestTotalCountController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetMaintenanceRequestListController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/list", method = RequestMethod.POST)
  public @ResponseBody
  Object getAlertList(@RequestBody MaintenanceRequest mRequest,HttpServletRequest
request) throws Exception {

    List<MaintenanceRequest> mRequestList =
maintenanceRequestService.searchUserMaintenanceRequest(mRequest);

    return DTOUtil.copy(MaintenanceRequest.class, mRequestList, new String[] {
MaintenanceRequest.CREATED_BY,
        MaintenanceRequest.UPDATED_BY });
  }
}
/**
```

```
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * GetReplenishmentAlertsListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetRequestTotalCountController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/total", method = RequestMethod.POST)
  public @ResponseBody
  Object getAlertList(@RequestBody MaintenanceRequest mRequest, HttpServletRequest
request) throws Exception {

    Long totalCount =
maintenanceRequestService.totalUserMaintenanceRequest(mRequest);

    return new JSONSuccessMap().addProp("totalCount", totalCount).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 6, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;

/**
 * RequestForReplenishmentController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class RequestForReplenishmentController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/replenishment-request/create", method
= RequestMethod.POST)
  public @ResponseBody
  Object getAlertList(@RequestBody MaintenanceRequest maintenanceRequest,
HttpServletRequest request) throws Exception {

    maintenanceRequestService.requestForReplenishment(maintenanceRequest);

    return
JSONMapUtils.createJSONSuccess(EMessages.REPLENISHMENT_REQUEST_CREATE_SUCCESS).getMap
();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 6, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;
import edu.upm.dpsm.cims.requeststatus.model.ERequestStatus;

/**
 * GetPendingRequestTotalCountController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class SendApprovalRequestController {
  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @RequestMapping(value = "maintenance-request/pending-request/approval", method =
RequestMethod.POST)
  public @ResponseBody
  Object getAlertList(@RequestBody MaintenanceRequest maintenanceRequest,
HttpServletRequest request) throws Exception {
    ERequestStatus requestStatus =
ERequestStatus.valueOf(maintenanceRequest.getRequestStatus());

    MaintenanceRequest existingPendingRequest =
maintenanceRequestService.get(maintenanceRequest);
    if (requestStatus.equals(ERequestStatus.APPROVED)) {
      existingPendingRequest.setRequestStatus(ERequestStatus.APPROVED.asInt());
    } else {
      existingPendingRequest.setRequestStatus(ERequestStatus.REJECTED.asInt());
    }

    maintenanceRequestService.update(existingPendingRequest);

    EMessages successMessage = requestStatus.equals(ERequestStatus.APPROVED) ?
EMessages.APPROVED_REQUEST_SUCCESS
        : EMessages.REJECTED_REQUEST_SUCCESS;

    return JSONMapUtils.createJSONSuccess(successMessage).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
```

```
 */
package edu.upm.dpsm.cims.maintenancerequest.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;

/**
 * MaintenanceRequestDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface MaintenanceRequestDAO extends DAO<MaintenanceRequest>{

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.dao;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;

/**
 * MaintenanceRequestDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class MaintenanceRequestDAOImpl extends JPADao<MaintenanceRequest> implements
MaintenanceRequestDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 24, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.model;
```

```
/**
 * Maintainable
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface Maintainable {
  public String getSerialNumber();
  public Long getLabwareId();
  public Long getId();
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;

/**
 * MaintenanceRequest
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "maintenance_request")
public class MaintenanceRequest extends JPAEntityModel {
  public final static String MODULE_ID      = "moduleId";
  public final static String ITEM_ID        = "itemId";
  public final static String ITEM_NAME      = "itemName";
  public final static String ITEM_STATUS    = "itemStatus";
  public final static String REQUEST_STATUS = "requestStatus";
  public final static String SERIAL_NUMBER  = "serialNumber";

  @Column(name = "module_id")
  private Long               moduleId;

  @Column(name = "definition_id")
  private Long               definitionId;

  @Column(name = "item_id")
  private Long               itemId;
```

```
@Column(name = "item_name")
private String             itemName;

@Column(name = "item_status")
private Integer            itemStatus;

@Column(name = "request_status")
private Integer            requestStatus;

@Column(name = "serial_number")
private String             serialNumber;

public Long getDefinitionId() {
  return definitionId;
}

public void setDefinitionId(Long definitionId) {
  this.definitionId = definitionId;
}

public String getItemName() {
  return itemName;
}

public void setItemName(String itemName) {
  this.itemName = itemName;
}

public Long getModuleId() {
  return moduleId;
}

public void setModuleId(Long moduleId) {
  this.moduleId = moduleId;
}

public Long getItemId() {
  return itemId;
}

public void setItemId(Long itemId) {
  this.itemId = itemId;
}

public Integer getItemStatus() {
  return itemStatus;
}

public void setItemStatus(Integer itemStatus) {
  this.itemStatus = itemStatus;
}

public Integer getRequestStatus() {
```

```
    return requestStatus;
  }

  public void setRequestStatus(Integer requestStatus) {
    this.requestStatus = requestStatus;
  }

  public String getSerialNumber() {
    return serialNumber;
  }

  public void setSerialNumber(String serialNumber) {
    this.serialNumber = serialNumber;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.search;

import static edu.upm.dpsm.cims.core.jpa.model.JPAModel.ID;
import static edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_ID;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_NAME;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_STATUS;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.MODULE_ID;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.REQUEST_STATUS;
import edu.upm.dpsm.cims.core.AppContext;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchDisjunction;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.core.util.UserAccountUtil;
import edu.upm.dpsm.cims.laboratoryunit.model.ELaboratoryUnit;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.modules.model.EModules;
import edu.upm.dpsm.cims.requeststatus.model.ERequestStatus;
import edu.upm.dpsm.cims.useraccount.models.ERoles;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * SearchActiveReplenishmentRecord
 *
 * Description:
```

```
 *
 * author : Miguelino San Miguel
 */
public class SearchActiveReplenishmentRecord extends
JPASearchModel<MaintenanceRequest> {

  public SearchActiveReplenishmentRecord(MaintenanceRequest model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(MaintenanceRequest model) throws Exception {
    SearchCondition searchCondition = new SearchCondition();
    if (!ObjectUtils.isEmpty(model.getModuleId())) {
      if (model.getModuleId() % 2 == 0) {
        searchCondition.add(MODULE_ID, ERestrictionType.EQ, model.getModuleId());
      } else {
        searchCondition.add(new SearchDisjunction().add(MODULE_ID,
ERestrictionType.EQ, model.getModuleId()).add(
            MODULE_ID, ERestrictionType.EQ, model.getModuleId() + 1L));

      }

    }
    UserAccount userAccount = this.getUserAccount();
    Long unitId = userAccount.getLaboratoryUnitId();
    Long roleId = userAccount.getRoleId();

    if (ERoles.LAB_ATTENDANT.getId().equals(roleId) ||
ERoles.LAB_FACULTY_COORDINATOR.getId().equals(roleId)) {

      ELaboratoryUnit unit = ELaboratoryUnit.valueOf(unitId);
      switch (unit) {
        case BIOLOGY:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.BIOLOGY_CHEMICAL.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.BIOLOGY_CHEMICAL_ITEM.asLong())
              .add(MODULE_ID, ERestrictionType.EQ, EModules.BIOLOGY_LABWARE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.BIOLOGY_LABWARE_ITEM.asLong()));
          break;
        case CHEMISTRY:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_CHEMICAL.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_CHEMICAL_ITEM.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_LABWARE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_LABWARE_ITEM.asLong()));
```

```
          break;
        case MATH_CS:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ, EModules.MATH_CS_MACHINE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.MATH_CS_MACHINE_ITEM.asLong()));
          break;
        case PHYSICS:
        default:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.PHYSICS_CHEMICAL.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.PHYSICS_CHEMICAL_ITEM.asLong())
              .add(MODULE_ID, ERestrictionType.EQ, EModules.PHYSICS_LABWARE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.PHYSICS_LABWARE_ITEM.asLong()));
      }
    }
    setSearch(searchCondition.add(SearchCondition
        .conjunction()
        .add(ID, ERestrictionType.EQ, model.getId())
        .add(ITEM_STATUS, ERestrictionType.EQ, model.getItemStatus())
        .add(ITEM_ID, ERestrictionType.EQ, model.getItemId())
        .add(ITEM_NAME, ERestrictionType.EQ, model.getItemName())
        .add(
            new SearchDisjunction().add(REQUEST_STATUS, ERestrictionType.EQ,
ERequestStatus.ALERT.asInt())
                .add(REQUEST_STATUS, ERestrictionType.EQ,
ERequestStatus.UNDER_REQUEST.asInt())
                .add(REQUEST_STATUS, ERestrictionType.EQ,
ERequestStatus.APPROVED.asInt()))

    ));
  }

  private UserAccount getUserAccount() throws Exception {
    Long id = UserAccountUtil.getUserAccountId();
    UserAccountService userAccountService =
AppContext.getBean(UserAccountService.class);
    return userAccountService.get(id);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 25, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.search;

import static edu.upm.dpsm.cims.core.jpa.model.JPAModel.ID;
```

```
import static edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_ID;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_NAME;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_STATUS;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.REQUEST_STATUS;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchDisjunction;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.requeststatus.model.ERequestStatus;

/**
 * SearchExistingRequest
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchExistingRequest extends JPASearchModel<MaintenanceRequest> {
  public SearchExistingRequest(MaintenanceRequest model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(MaintenanceRequest model) throws Exception {

    final SearchCondition searchCondition = new SearchCondition();
    setSearch(searchCondition
      .add(SearchCondition.conjunction()
          .add(ID,ERestrictionType.EQ,model.getId())
          .add(ITEM_STATUS,ERestrictionType.EQ,model.getItemStatus())
          .add(ITEM_ID,ERestrictionType.EQ,model.getItemId())
          .add(ITEM_NAME,ERestrictionType.EQ,model.getItemName())
          .add(new SearchDisjunction()
              .add(REQUEST_STATUS,ERestrictionType.EQ,ERequestStatus.ALERT.asInt())

.add(REQUEST_STATUS,ERestrictionType.EQ,ERequestStatus.UNDER_REQUEST.asInt())

.add(REQUEST_STATUS,ERestrictionType.EQ,ERequestStatus.APPROVED.asInt()))

     ));
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
```

```
package edu.upm.dpsm.cims.maintenancerequest.search;

import static edu.upm.dpsm.cims.core.jpa.model.JPAModel.ID;
import static edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_ID;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_NAME;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_STATUS;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.MODULE_ID;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.REQUEST_STATUS;
import edu.upm.dpsm.cims.core.AppContext;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchDisjunction;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.core.util.UserAccountUtil;
import edu.upm.dpsm.cims.laboratoryunit.model.ELaboratoryUnit;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.modules.model.EModules;
import edu.upm.dpsm.cims.useraccount.models.ERoles;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * SearchMaintenanceRequest
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchMaintenanceRequest extends JPASearchModel<MaintenanceRequest> {

  public SearchMaintenanceRequest(MaintenanceRequest model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(MaintenanceRequest model) throws Exception {

    SearchCondition searchCondition = new SearchCondition();

    if (!ObjectUtils.isEmpty(model.getModuleId())) {
      if (model.getModuleId() % 2 == 0) {
        searchCondition.add(MODULE_ID, ERestrictionType.EQ, model.getModuleId());
      } else {
        searchCondition.add(new SearchDisjunction().add(MODULE_ID,
ERestrictionType.EQ, model.getModuleId()).add(
            MODULE_ID, ERestrictionType.EQ, model.getModuleId() + 1L));

      }
```

```
    }

    UserAccount userAccount = this.getUserAccount();
    Long unitId = userAccount.getLaboratoryUnitId();
    Long roleId = userAccount.getRoleId();

    if (ERoles.LAB_ATTENDANT.getId().equals(roleId) ||
ERoles.LAB_FACULTY_COORDINATOR.getId().equals(roleId)) {

      ELaboratoryUnit unit = ELaboratoryUnit.valueOf(unitId);
      switch (unit) {
        case BIOLOGY:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.BIOLOGY_CHEMICAL.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.BIOLOGY_CHEMICAL_ITEM.asLong())
              .add(MODULE_ID, ERestrictionType.EQ, EModules.BIOLOGY_LABWARE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.BIOLOGY_LABWARE_ITEM.asLong()));
          break;
        case CHEMISTRY:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_CHEMICAL.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_CHEMICAL_ITEM.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_LABWARE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.CHEMISTRY_LABWARE_ITEM.asLong()));
          break;
        case MATH_CS:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ, EModules.MATH_CS_MACHINE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.MATH_CS_MACHINE_ITEM.asLong()));
          break;
        case PHYSICS:
        default:
          searchCondition.add(SearchCondition.disjunction()
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.PHYSICS_CHEMICAL.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.PHYSICS_CHEMICAL_ITEM.asLong())
              .add(MODULE_ID, ERestrictionType.EQ, EModules.PHYSICS_LABWARE.asLong())
              .add(MODULE_ID, ERestrictionType.EQ,
EModules.PHYSICS_LABWARE_ITEM.asLong()));
      }
    }

    setSearch(searchCondition.add(SearchCondition.conjunction().add(ID,
ERestrictionType.EQ, model.getId())
```

```
        .add(ITEM_STATUS, ERestrictionType.EQ, model.getItemStatus())
        .add(ITEM_ID, ERestrictionType.EQ, model.getItemId()).add(ITEM_NAME,
ERestrictionType.EQ, model.getItemName())
        .add(REQUEST_STATUS, ERestrictionType.EQ, model.getRequestStatus())

    ));

  }

  private UserAccount getUserAccount() throws Exception {
    Long id = UserAccountUtil.getUserAccountId();
    UserAccountService userAccountService =
AppContext.getBean(UserAccountService.class);
    return userAccountService.get(id);
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 25, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.search;

import static edu.upm.dpsm.cims.core.app.model.EntityModel.UPDATED_DATE;
import static edu.upm.dpsm.cims.core.jpa.model.JPAModel.ID;
import static edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_ID;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.ITEM_NAME;
import static
edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest.REQUEST_STATUS;

import java.sql.Timestamp;

import org.joda.time.LocalDate;

import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchDisjunction;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.requeststatus.model.ERequestStatus;

/**
 * SearchExistingRequest
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchStaleRequest extends JPASearchModel<MaintenanceRequest> {
```

```
  public SearchStaleRequest(MaintenanceRequest model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(MaintenanceRequest model) throws Exception {

    final SearchCondition searchCondition = new SearchCondition();
    LocalDate currentDate = new LocalDate();
    setSearch(searchCondition
      .add(SearchCondition.conjunction()
          .add(ID,ERestrictionType.EQ,model.getId())
          .add(ITEM_ID,ERestrictionType.EQ,model.getItemId())
          .add(ITEM_NAME,ERestrictionType.EQ,model.getItemName())
          .add(UPDATED_DATE,ERestrictionType.LT,new
Timestamp(currentDate.minusDays(4).toDate().getTime()))
          .add(new SearchDisjunction()

.add(REQUEST_STATUS,ERestrictionType.EQ,ERequestStatus.REJECTED.asInt())

.add(REQUEST_STATUS,ERestrictionType.EQ,ERequestStatus.APPROVED.asInt()))

     ));
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.service;

import java.util.List;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;

/**
 * MaintenanceRequestService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface MaintenanceRequestService extends
BusinessService<MaintenanceRequest> {
  public MaintenanceRequest createAlertForReplenishment(String table, MonitorItem
monitoredItem) throws Exception;
```

```
  public MaintenanceRequest deleteAlertForReplenishment(String table, MonitorItem
monitoredItem) throws Exception;

  public List<MaintenanceRequest> searchAlertForReplenishment(String table,
MonitorItem monitoredItem) throws Exception;

  public Long totalAlerts() throws Exception;

  public List<MaintenanceRequest> searchAlerts(MaintenanceRequest maintenanceRequest)
throws Exception;

  public MaintenanceRequest requestForReplenishment(MaintenanceRequest
maintenanceRequest) throws Exception;

  public List<MaintenanceRequest> searchUserMaintenanceRequest(MaintenanceRequest
maintenanceRequest)
      throws Exception;

  public Long totalUserMaintenanceRequest(MaintenanceRequest maintenanceRequest)
throws Exception;

  public MaintenanceRequest createRepairAlert(Maintainable labware, String itemName,
Long moduleId)
      throws Exception;

  public MaintenanceRequest createCondemnationAlert(Maintainable labware, String
itemName, Long moduleId)
      throws Exception;

  public MaintenanceRequest createRepairRequest(MaintenanceRequest t) throws
Exception;

  public MaintenanceRequest createCondemnationRequest(MaintenanceRequest t) throws
Exception;

  public void deleteStaleRequests() throws Exception;
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.service;

import java.util.List;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
```

```
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;
import edu.upm.dpsm.cims.maintenancerequest.dao.MaintenanceRequestDAO;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchActiveReplenishmentRecord;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchMaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchStaleRequest;
import edu.upm.dpsm.cims.modules.model.Modules;
import edu.upm.dpsm.cims.modules.service.ModulesService;
import edu.upm.dpsm.cims.requeststatus.model.ERequestStatus;

/**
 * MaintenanceRequestServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class MaintenanceRequestServiceImpl extends JPAService<MaintenanceRequest>
implements MaintenanceRequestService {
  @Autowired
  private MaintenanceRequestDAO dao;

  protected static Logger       logger =
Logger.getLogger(MaintenanceRequestServiceImpl.class);

  @Autowired
  private ModulesService        modulesService;

  @Override
  public DAO<MaintenanceRequest> getDAO() {
    return dao;
  }

  @Override
  public MaintenanceRequest createAlertForReplenishment(String table, MonitorItem
monitoredItem) throws Exception {
    List<MaintenanceRequest> requests = null;

    requests = this.searchAlertForReplenishment(table, monitoredItem);

    if (ObjectUtils.isEmpty(requests)) {
      Modules module = modulesService.getModuleForTable(table);
      MaintenanceRequest request = new MaintenanceRequest();
      request.setModuleId(module.getId());
      request.setDefinitionId(monitoredItem.getId());
```

```
      request.setItemName(monitoredItem.getName());
      request.setItemStatus(EItemStatus.REPLENISHMENT.asInt());
      request.setRequestStatus(ERequestStatus.ALERT.asInt());
      return this.save(request);
    }
    return null;
  }

  @Override
  public MaintenanceRequest deleteAlertForReplenishment(String table, MonitorItem
monitoredItem) throws Exception {
    List<MaintenanceRequest> requests = this.searchAlertForReplenishment(table,
monitoredItem);

    if (requests.size() > 0) {
      for (MaintenanceRequest request : requests) {
        this.delete(request);
      }
      // NOTE: expecting that a single requests exist for this replenishment
      // alert
      return requests.get(0);
    }

    return null;
  }

  @Override
  public List<MaintenanceRequest> searchAlertForReplenishment(String table,
MonitorItem monitoredItem) throws Exception {
    Modules module = modulesService.getModuleForTable(table);
    MaintenanceRequest request = new MaintenanceRequest();
    request.setModuleId(module.getId());
    request.setDefinitionId(monitoredItem.getId());
    request.setItemStatus(EItemStatus.REPLENISHMENT.asInt());
    SearchActiveReplenishmentRecord searchRequest = new
SearchActiveReplenishmentRecord(request);
    request.setSearchModel(searchRequest);
    return super.search(request);
  }

  @Override
  public List<MaintenanceRequest> searchAlerts(MaintenanceRequest alertRequest)
throws Exception {
    alertRequest.setRequestStatus(ERequestStatus.ALERT.asInt());
    SearchActiveReplenishmentRecord searchRequest = new
SearchActiveReplenishmentRecord(alertRequest);
    alertRequest.setSearchModel(searchRequest);
    return super.search(alertRequest);

  }

  @Override
  public Long totalAlerts() throws Exception {
```

```
    MaintenanceRequest alertRequest = new MaintenanceRequest();
    alertRequest.setRequestStatus(ERequestStatus.ALERT.asInt());
    SearchActiveReplenishmentRecord searchRequest = new
SearchActiveReplenishmentRecord(alertRequest);
    alertRequest.setSearchModel(searchRequest);
    return super.getTotalSize(alertRequest);
  }

  @Override
  public MaintenanceRequest requestForReplenishment(MaintenanceRequest
maintenanceRequest) throws Exception {
    MaintenanceRequest alertRequest = this.get(maintenanceRequest);
    alertRequest.setRequestStatus(ERequestStatus.UNDER_REQUEST.asInt());
    return super.update(alertRequest);
  }

  @Override
  public List<MaintenanceRequest> searchUserMaintenanceRequest(MaintenanceRequest
maintenanceRequest) throws Exception {
    SearchMaintenanceRequest searchRequest = new
SearchMaintenanceRequest(maintenanceRequest);
    maintenanceRequest.setSearchModel(searchRequest);
    return super.search(maintenanceRequest);

  }

  @Override
  public Long totalUserMaintenanceRequest(MaintenanceRequest maintenanceRequest)
throws Exception {
    SearchMaintenanceRequest searchRequest = new
SearchMaintenanceRequest(maintenanceRequest);
    maintenanceRequest.setSearchModel(searchRequest);
    return super.getTotalSize(maintenanceRequest);
  }

  @Override
  public MaintenanceRequest createRepairAlert(Maintainable labware, String itemName,
Long moduleId) throws Exception {
    MaintenanceRequest request = new MaintenanceRequest();
    request.setDefinitionId(labware.getLabwareId());
    request.setItemId(labware.getId());
    request.setItemName(itemName);
    request.setModuleId(moduleId);
    request.setItemStatus(EItemStatus.REPAIR.asInt());
    request.setRequestStatus(ERequestStatus.ALERT.asInt());
    request.setSerialNumber(labware.getSerialNumber());
    return super.save(request);
  }

  @Override
  public MaintenanceRequest createCondemnationAlert(Maintainable labware, String
itemName, Long moduleId)
      throws Exception {
```

```
    MaintenanceRequest request = new MaintenanceRequest();
    request.setDefinitionId(labware.getLabwareId());
    request.setItemId(labware.getId());
    request.setItemName(itemName);
    request.setModuleId(moduleId);
    request.setItemStatus(EItemStatus.CONDEMNATION.asInt());
    request.setRequestStatus(ERequestStatus.ALERT.asInt());
    request.setSerialNumber(labware.getSerialNumber());
    return super.save(request);
  }

  @Override
  public MaintenanceRequest createRepairRequest(MaintenanceRequest t) throws
Exception {
    MaintenanceRequest request = super.get(t);
    request.setItemStatus(EItemStatus.REPAIR.asInt());
    request.setRequestStatus(ERequestStatus.UNDER_REQUEST.asInt());
    return super.update(request);
  }

  @Override
  public MaintenanceRequest createCondemnationRequest(MaintenanceRequest t) throws
Exception {
    MaintenanceRequest request = super.get(t);
    request.setItemStatus(EItemStatus.CONDEMNATION.asInt());
    request.setRequestStatus(ERequestStatus.UNDER_REQUEST.asInt());
    return super.update(request);
  }

  @Scheduled(cron="0 0 12 * * ?")
  @Override
  public void deleteStaleRequests() throws Exception {
    logger.info("Scheduled deleting of stale maintenance requests is on going");
    MaintenanceRequest request = new MaintenanceRequest();
    SearchStaleRequest ssr = new SearchStaleRequest(request);
    request.setSearchModel(ssr);
    List<MaintenanceRequest> staleRequestList = super.search(request);
    int size = ObjectUtils.isEmpty(staleRequestList) ? 0 : staleRequestList.size();
    logger.info("Found " + size + " stale maintenance requests!");
    if (size > 0) {
      super.deleteAll(staleRequestList);
    }
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 1, 2014
 */
package edu.upm.dpsm.cims.maintenancerequest.service;
```

```
import java.util.List;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;
import edu.upm.dpsm.cims.maintenancerequest.dao.MaintenanceRequestDAO;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;
import edu.upm.dpsm.cims.maintenancerequest.model.MaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchActiveReplenishmentRecord;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchMaintenanceRequest;
import edu.upm.dpsm.cims.maintenancerequest.search.SearchStaleRequest;
import edu.upm.dpsm.cims.modules.model.Modules;
import edu.upm.dpsm.cims.modules.service.ModulesService;
import edu.upm.dpsm.cims.requeststatus.model.ERequestStatus;

/**
 * MaintenanceRequestServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class MaintenanceRequestServiceImpl extends JPAService<MaintenanceRequest>
implements MaintenanceRequestService {
  @Autowired
  private MaintenanceRequestDAO dao;

  protected static Logger       logger =
Logger.getLogger(MaintenanceRequestServiceImpl.class);

  @Autowired
  private ModulesService        modulesService;

  @Override
  public DAO<MaintenanceRequest> getDAO() {
    return dao;
  }

  @Override
  public MaintenanceRequest createAlertForReplenishment(String table, MonitorItem
monitoredItem) throws Exception {
    List<MaintenanceRequest> requests = null;
```

```
    requests = this.searchAlertForReplenishment(table, monitoredItem);

    if (ObjectUtils.isEmpty(requests)) {
      Modules module = modulesService.getModuleForTable(table);
      MaintenanceRequest request = new MaintenanceRequest();
      request.setModuleId(module.getId());
      request.setDefinitionId(monitoredItem.getId());
      request.setItemName(monitoredItem.getName());
      request.setItemStatus(EItemStatus.REPLENISHMENT.asInt());
      request.setRequestStatus(ERequestStatus.ALERT.asInt());
      return this.save(request);
    }
    return null;
  }

  @Override
  public MaintenanceRequest deleteAlertForReplenishment(String table, MonitorItem
monitoredItem) throws Exception {
    List<MaintenanceRequest> requests = this.searchAlertForReplenishment(table,
monitoredItem);

    if (requests.size() > 0) {
      for (MaintenanceRequest request : requests) {
        this.delete(request);
      }
      // NOTE: expecting that a single requests exist for this replenishment
      // alert
      return requests.get(0);
    }

    return null;
  }

  @Override
  public List<MaintenanceRequest> searchAlertForReplenishment(String table,
MonitorItem monitoredItem) throws Exception {
    Modules module = modulesService.getModuleForTable(table);
    MaintenanceRequest request = new MaintenanceRequest();
    request.setModuleId(module.getId());
    request.setDefinitionId(monitoredItem.getId());
    request.setItemStatus(EItemStatus.REPLENISHMENT.asInt());
    SearchActiveReplenishmentRecord searchRequest = new
SearchActiveReplenishmentRecord(request);
    request.setSearchModel(searchRequest);
    return super.search(request);
  }

  @Override
  public List<MaintenanceRequest> searchAlerts(MaintenanceRequest alertRequest)
throws Exception {
    alertRequest.setRequestStatus(ERequestStatus.ALERT.asInt());
```

```
    SearchActiveReplenishmentRecord searchRequest = new
SearchActiveReplenishmentRecord(alertRequest);
    alertRequest.setSearchModel(searchRequest);
    return super.search(alertRequest);

  }

  @Override
  public Long totalAlerts() throws Exception {
    MaintenanceRequest alertRequest = new MaintenanceRequest();
    alertRequest.setRequestStatus(ERequestStatus.ALERT.asInt());
    SearchActiveReplenishmentRecord searchRequest = new
SearchActiveReplenishmentRecord(alertRequest);
    alertRequest.setSearchModel(searchRequest);
    return super.getTotalSize(alertRequest);
  }

  @Override
  public MaintenanceRequest requestForReplenishment(MaintenanceRequest
maintenanceRequest) throws Exception {
    MaintenanceRequest alertRequest = this.get(maintenanceRequest);
    alertRequest.setRequestStatus(ERequestStatus.UNDER_REQUEST.asInt());
    return super.update(alertRequest);
  }

  @Override
  public List<MaintenanceRequest> searchUserMaintenanceRequest(MaintenanceRequest
maintenanceRequest) throws Exception {
    SearchMaintenanceRequest searchRequest = new
SearchMaintenanceRequest(maintenanceRequest);
    maintenanceRequest.setSearchModel(searchRequest);
    return super.search(maintenanceRequest);

  }

  @Override
  public Long totalUserMaintenanceRequest(MaintenanceRequest maintenanceRequest)
throws Exception {
    SearchMaintenanceRequest searchRequest = new
SearchMaintenanceRequest(maintenanceRequest);
    maintenanceRequest.setSearchModel(searchRequest);
    return super.getTotalSize(maintenanceRequest);
  }

  @Override
  public MaintenanceRequest createRepairAlert(Maintainable labware, String itemName,
Long moduleId) throws Exception {
    MaintenanceRequest request = new MaintenanceRequest();
    request.setDefinitionId(labware.getLabwareId());
    request.setItemId(labware.getId());
    request.setItemName(itemName);
    request.setModuleId(moduleId);
    request.setItemStatus(EItemStatus.REPAIR.asInt());
```

```
    request.setRequestStatus(ERequestStatus.ALERT.asInt());
    request.setSerialNumber(labware.getSerialNumber());
    return super.save(request);
  }

  @Override
  public MaintenanceRequest createCondemnationAlert(Maintainable labware, String
itemName, Long moduleId)
      throws Exception {
    MaintenanceRequest request = new MaintenanceRequest();
    request.setDefinitionId(labware.getLabwareId());
    request.setItemId(labware.getId());
    request.setItemName(itemName);
    request.setModuleId(moduleId);
    request.setItemStatus(EItemStatus.CONDEMNATION.asInt());
    request.setRequestStatus(ERequestStatus.ALERT.asInt());
    request.setSerialNumber(labware.getSerialNumber());
    return super.save(request);
  }

  @Override
  public MaintenanceRequest createRepairRequest(MaintenanceRequest t) throws
Exception {
    MaintenanceRequest request = super.get(t);
    request.setItemStatus(EItemStatus.REPAIR.asInt());
    request.setRequestStatus(ERequestStatus.UNDER_REQUEST.asInt());
    return super.update(request);
  }

  @Override
  public MaintenanceRequest createCondemnationRequest(MaintenanceRequest t) throws
Exception {
    MaintenanceRequest request = super.get(t);
    request.setItemStatus(EItemStatus.CONDEMNATION.asInt());
    request.setRequestStatus(ERequestStatus.UNDER_REQUEST.asInt());
    return super.update(request);
  }

  @Scheduled(cron="0 0 12 * * ?")
  @Override
  public void deleteStaleRequests() throws Exception {
    logger.info("Scheduled deleting of stale maintenance requests is on going");
    MaintenanceRequest request = new MaintenanceRequest();
    SearchStaleRequest ssr = new SearchStaleRequest(request);
    request.setSearchModel(ssr);
    List<MaintenanceRequest> staleRequestList = super.search(request);
    int size = ObjectUtils.isEmpty(staleRequestList) ? 0 : staleRequestList.size();
    logger.info("Found " + size + " stale maintenance requests!");
    if (size > 0) {
      super.deleteAll(staleRequestList);
    }
  }
```

```
}
package edu.upm.dpsm.cims.mathcs.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;
import edu.upm.dpsm.cims.mathcs.labware.definition.services.MathCSLabwareService;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class GetMathCSLabwareController {

  @Autowired
  private MathCSLabwareService mathCSLabwareService;

  @RequestMapping(value = "mathcs/labware/get", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody MathCSLabware mathCSLabware,
HttpServletRequest request) throws Exception {
    MathCSLabware labwareItem = mathCSLabwareService.get(mathCSLabware);

    return DTOUtil.copy(MathCSLabware.class, labwareItem, new String[] {
MathCSLabware.CREATED_BY,
        MathCSLabware.UPDATED_BY, MathCSLabware.MATH_CS_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.mathcs.labware.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;
import edu.upm.dpsm.cims.mathcs.labware.definition.services.MathCSLabwareService;
```

```
/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchMathCSLabwareController {

  @Autowired
  private MathCSLabwareService mathCSLabwareService;

  @RequestMapping(value = "mathcs/labware/search", method = RequestMethod.POST)
  public @ResponseBody
  Object labwareListPage(@RequestBody MathCSLabware mathCSLabware, HttpServletRequest
request)
      throws Exception {
    List<MathCSLabware> chemicalList = mathCSLabwareService.search(mathCSLabware);

    return DTOUtil.copy(MathCSLabware.class, chemicalList, new String[] {
MathCSLabware.CREATED_BY,
        MathCSLabware.UPDATED_BY, MathCSLabware.MATH_CS_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.mathcs.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;
import edu.upm.dpsm.cims.mathcs.labware.definition.services.MathCSLabwareService;

/**
 * @author Miguelino San Miguel
 * @param <MathCSChemical>
 */
@Controller
public class UpdateMathCSLabwareController {
  @Autowired
  public MathCSLabwareService mathCSLabwareService;

  @RequestMapping(value = "mathcs/labware/update-definition", method =
RequestMethod.POST)
  public @ResponseBody
```

```
  Object updateChemical(@RequestBody @Validated MathCSLabware mathCSLabware,
HttpServletRequest request)
      throws Exception {

    MathCSLabware ccd = new MathCSLabware();
    ccd.setId(mathCSLabware.getId());
    ccd = mathCSLabwareService.get(ccd);
    BeanUtils.copyProperties(mathCSLabware, ccd, new String[] { MathCSLabware.ID,
MathCSLabware.TOTAL_QUANTITY,
        MathCSLabware.OK_QUANTITY, MathCSLabware.CREATED_BY_ID,
MathCSLabware.CREATED_DATE });
    mathCSLabwareService.update(ccd);
    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_UPDATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.definition.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.mathcs.labware.definition.dao.MathCSLabwareDAO;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;

/**
 * MathCSLabwareDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class MathCSLabwareDAOImpl extends JPADao<MathCSLabware> implements
MathCSLabwareDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.definition.dao;
```

```
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;

/**
 * MathCSLabwareDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface MathCSLabwareDAO extends DAO<MathCSLabware> {

}
package edu.upm.dpsm.cims.mathcs.labware.definition.models;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "mathcs_machine_definition")
@Access(AccessType.FIELD)
public class MathCSLabware extends JPAEntityModel implements MonitorItem,
Serializable {

  private static final long        serialVersionUID      = -7617390104822472391L;
  public final static String       NAME                  = "name";
  public final static String       TYPE                  = "type";
  public final static String       SIZE                  = "size";
  public final static String       OK_QUANTITY           = "okQuantity";
  public final static String       BROKEN_QUANTITY       = "brokenQuantity";
  public final static String       STAINED_QUANTITY      = "stainedQuantity";
  public final static String       TOTAL_QUANTITY        = "totalQuantity";
  public final static String       MINIMUM               = "minimum";
  public final static String       MATH_CS_LABWARE_ITEMS = "mathCSLabwareItems";
```

```
  @Column(name = "type")
  private String                    type;

  @Column(name = "size")
  private String                    size;

  @Column(name = "name")
  private String                    name;

  @Column(name = "total_quantity")
  private Long                      totalQuantity         = Long.valueOf(0L);      ;

  @Column(name = "broken_quantity")
  private Long                      brokenQuantity        = Long.valueOf(0L);      ;

  @Column(name = "stained_quantity")
  private Long                      stainedQuantity       = Long.valueOf(0L);      ;

  @Column(name = "ok_quantity")
  private Long                      okQuantity            = Long.valueOf(0L);      ;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
  private Integer                   minimum;

  @OneToMany(mappedBy = "mathCSLabware", cascade = { CascadeType.REMOVE }, fetch =
FetchType.LAZY)
  private List<MathCSLabwareItem> mathCSLabwareItems;

  public List<MathCSLabwareItem> getMathCSLabwareItems() {
    return mathCSLabwareItems;
  }

  public void setMathCSLabwareItems(List<MathCSLabwareItem> mathCSLabwareItems) {
    this.mathCSLabwareItems = mathCSLabwareItems;
  }

  public String getType() {
    return type;
  }

  public void setType(String type) {
    this.type = type;
  }

  public String getSize() {
    return size;
  }

  public void setSize(String size) {
    this.size = size;
```

```
  }

  public Long getTotalQuantity() {
    return totalQuantity;
  }

  public void setTotalQuantity(Long totalQuantity) {
    this.totalQuantity = totalQuantity;
  }

  public Long getBrokenQuantity() {
    return brokenQuantity;
  }

  public void setBrokenQuantity(Long brokenQuantity) {
    this.brokenQuantity = brokenQuantity;
  }

  public Long getStainedQuantity() {
    return stainedQuantity;
  }

  public void setStainedQuantity(Long stainedQuantity) {
    this.stainedQuantity = stainedQuantity;
  }

  public Long getOkQuantity() {
    return okQuantity;
  }

  public void setOkQuantity(Long okQuantity) {
    this.okQuantity = okQuantity;
  }

  @Override
  public String getName() {
    return this.name;
  }

  @Override
  public Integer getMinimum() {
    return this.minimum;
  }

  public void setName(String name) {
    this.name = name;
  }

  public void setMinimum(Integer minimum) {
    this.minimum = minimum;
  }

}
```

```
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.definition.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.mathcs.labware.definition.dao.MathCSLabwareDAO;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;
import edu.upm.dpsm.cims.mathcs.labware.definition.services.MathCSLabwareService;

/**
 * MathCSLabwareServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class MathCSLabwareServiceImpl extends JPAService<MathCSLabware> implements
MathCSLabwareService {

  @Autowired
  private MathCSLabwareDAO dao;

  @Override
  public DAO<MathCSLabware> getDAO() {
    return dao;
  }

}
package edu.upm.dpsm.cims.mathcs.labware.definition.services;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;

/**
 * @author Miguelino San Miguel
 */

public interface MathCSLabwareService extends BusinessService<MathCSLabware> {

}
package edu.upm.dpsm.cims.mathcs.labware.item.controllers;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;
import edu.upm.dpsm.cims.mathcs.labware.item.services.MathCSLabwareItemService;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddMathCSLabwareItemController {
  @Autowired
  private MathCSLabwareItemService mathCSLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "mathcs/labware/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody @Validated MathCSLabwareItem mathCSLabwareItem,
      HttpServletRequest request) throws Exception {

    mathCSLabwareItemService.saveItems(mathCSLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_ITEM_ADD_SUCCESS).getMap();
  }


}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.controllers;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;
import edu.upm.dpsm.cims.mathcs.labware.item.services.MathCSLabwareItemService;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateMathCSLabwareItemCondemnationAlertController {
  @Autowired
  private MathCSLabwareItemService mathCSLabwareItemService;

  @MonitorRequest(status=EItemStatus.CONDEMNATION)
  @RequestMapping(value = "mathcs/labware/item/condemnation-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody MathCSLabwareItem mathCSLabwareItem,
HttpServletRequest request)
      throws Exception {

    mathCSLabwareItemService.createCondemnationAlert(mathCSLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.CONDEMNATION_REQUEST_CREATE_SUCCESS).getMap(
);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;
import edu.upm.dpsm.cims.mathcs.labware.item.services.MathCSLabwareItemService;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreateMathCSLabwareItemRepairAlertController {
  @Autowired
  private MathCSLabwareItemService mathCSLabwareItemService;

  @MonitorRequest(status=EItemStatus.REPAIR)
  @RequestMapping(value = "mathcs/labware/item/repair-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody MathCSLabwareItem mathCSLabwareItem,
HttpServletRequest request)
      throws Exception {

    mathCSLabwareItemService.createRepairAlert(mathCSLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.REPAIR_REQUEST_CREATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;
import edu.upm.dpsm.cims.mathcs.labware.item.services.MathCSLabwareItemService;

/**
 * SearchChemicalLabwareItemController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class SearchMathCSChemicalLabwareItemController {
  @Autowired
  private MathCSLabwareItemService mathCSLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "mathcs/labware/item/search", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody MathCSLabwareItem mathCSLabwareItem,
HttpServletRequest request)
      throws Exception {

    List<MathCSLabwareItem> list =
mathCSLabwareItemService.searchLabwareItem(mathCSLabwareItem);

    return DTOUtil.copy(MathCSLabwareItem.class, list, new String[] {
MathCSLabwareItem.CREATED_BY,
        MathCSLabwareItem.UPDATED_BY, MathCSLabwareItem.MATH_CS_LABWARE });
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.dao.impl;
```

```
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.mathcs.labware.item.dao.MathCSLabwareItemDAO;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;

/**
 * MathCSLabwareItemDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class MathCSLabwareItemDAOImpl extends JPADao<MathCSLabwareItem> implements
MathCSLabwareItemDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;

/**
 * MathCSLabwareItemDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface MathCSLabwareItemDAO extends DAO<MathCSLabwareItem> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.models;

import java.io.Serializable;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;
import edu.upm.dpsm.cims.mathcs.labware.definition.models.MathCSLabware;

/**
 * MathCSLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "mathcs_machine_item")
public class MathCSLabwareItem extends JPAEntityModel implements Serializable,
Maintainable {
  public final static String LABWARE_ID        = "labwareId";
  public final static String MATH_CS_LABWARE = "mathCSLabware";
  private static final long  serialVersionUID  = 30111675844002629 57L;

  @Column(name = "labware_id")
  private Long               labwareId;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "labware_id", updatable = false, insertable = false)
  private MathCSLabware   mathCSLabware;

  @Column(name = "labware_status")
  private Integer            labwareStatus;

  @NotEmpty
  @Transient
  private String             labwareName;

  @Transient
  private Integer            quantity;

  @Column(name="serial_number")
  private String serialNumber;

  public String getSerialNumber() {
    return serialNumber;
  }
```

```
  public void setSerialNumber(String serialNumber) {
    this.serialNumber = serialNumber;
  }

  public Integer getQuantity() {
    return quantity;
  }

  public void setQuantity(Integer quantity) {
    this.quantity = quantity;
  }

  public String getLabwareName() {
    return labwareName;
  }

  public void setLabwareName(String labwareName) {
    this.labwareName = labwareName;
  }

  public Long getLabwareId() {
    return labwareId;
  }

  public void setLabwareId(Long labwareId) {
    this.labwareId = labwareId;
  }

  public MathCSLabware getMathCSLabware() {
    return mathCSLabware;
  }

  public void setMathCSLabware(MathCSLabware mathCSLabware) {
    this.mathCSLabware = mathCSLabware;
  }

  public Integer getLabwareStatus() {
    return labwareStatus;
  }

  public void setLabwareStatus(Integer labwareStatus) {
    this.labwareStatus = labwareStatus;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.search;
```

```
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;
import static edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem.*;
/**
 * SearchMathCSLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchMathCSLabwareItem extends JPASearchModel<MathCSLabwareItem> {

  public SearchMathCSLabwareItem(MathCSLabwareItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(MathCSLabwareItem model) throws Exception {
   setSearch(new SearchCondition()
     .add(LABWARE_ID,ERestrictionType.EQ, model.getLabwareId()));
  }


}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.mathcs.labware.item.search;

import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;
import static edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem.*;
/**
 * SearchMathCSLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchMathCSLabwareItem extends JPASearchModel<MathCSLabwareItem> {

  public SearchMathCSLabwareItem(MathCSLabwareItem model) throws Exception {
    super(model);
  }
```

```
  @Override
  protected void createSearchCriteria(MathCSLabwareItem model) throws Exception {
   setSearch(new SearchCondition()
     .add(LABWARE_ID,ERestrictionType.EQ, model.getLabwareId()));
  }


}
package edu.upm.dpsm.cims.mathcs.labware.item.services;

import java.util.List;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.mathcs.labware.item.models.MathCSLabwareItem;

/**
 * @author Miguelino San Miguel
 */
public interface MathCSLabwareItemService extends BusinessService<MathCSLabwareItem>
{
  public MathCSLabwareItem saveItems(MathCSLabwareItem mathCSLabwareItem) throws
Exception;

  public List<MathCSLabwareItem> searchLabwareItem(MathCSLabwareItem
mathCSLabwareItem) throws Exception;

  public MathCSLabwareItem createRepairAlert(MathCSLabwareItem mathCSLabwareItem)
throws Exception;

  public MathCSLabwareItem createCondemnationAlert(MathCSLabwareItem
mathCSLabwareItem) throws Exception;
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.modules.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.modules.model.Modules;
import edu.upm.dpsm.cims.modules.search.SearchLaboratoryModule;
import edu.upm.dpsm.cims.modules.service.ModulesService;
```

```
/**
 * GetModuleListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetModuleListController {

  @Autowired
  private ModulesService modulesService;

  @RequestMapping(value = "modules/list", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(HttpServletRequest request) throws Exception {
    Modules module = new Modules();
    SearchLaboratoryModule searchModule = new SearchLaboratoryModule(module);
    module.setSearchModel(searchModule);
    return modulesService.search(module);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.modules.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.modules.model.Modules;

/**
 * ModulesDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface ModulesDAO extends DAO<Modules> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.modules.dao;
```

```
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.modules.model.Modules;

/**
 * ModulesDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class ModulesDAOImpl extends JPADao<Modules> implements ModulesDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.modules.model;

/**
 * EModules
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum EModules {
  USERS(1),
  ROLES(2),
  CHEMISTRY_CHEMICAL(3),
  CHEMISTRY_CHEMICAL_ITEM(4),
  CHEMISTRY_LABWARE(5),
  CHEMISTRY_LABWARE_ITEM(6),
  PHYSICS_CHEMICAL(7),
  PHYSICS_CHEMICAL_ITEM(8),
  PHYSICS_LABWARE(9),
  PHYSICS_LABWARE_ITEM(10),
  MATH_CS_MACHINE(11),
  MATH_CS_MACHINE_ITEM(12),
  BIOLOGY_CHEMICAL(13),
  BIOLOGY_CHEMICAL_ITEM(14),
  BIOLOGY_LABWARE(15),
  BIOLOGY_LABWARE_ITEM(16),

  ;
```

```
  private long moduleId;

  private EModules(long moduleId) {
    this.moduleId = moduleId;
  }

  public static EModules valueOf(Long moduleId) {
    for (EModules module : EModules.values()) {
      if(moduleId != null && module.asLong() == moduleId.longValue()){
        return module;
      }
    }
    throw new RuntimeException("No matching enum value for given module id");
  }

  public long asLong() {
    return this.moduleId;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.modules.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * Modules
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "modules")
public class Modules extends JPALookupModel {

  public final static String DEPARTMENT_ID = "departmentId";
  public final static String TABLE = "table";
  public final static String NAME = "name";

  @Column(name = "department_id")
  private Long   departmentId;

  @Column(name = "table")
```

```
  private String table;

  @Column(name = "name")
  private String name;

  public Long getDepartmentId() {
    return departmentId;
  }

  public void setDepartmentId(Long departmentId) {
    this.departmentId = departmentId;
  }

  public String getTable() {
    return table;
  }

  public void setTable(String table) {
    this.table = table;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 11, 2014
 */
package edu.upm.dpsm.cims.modules.search;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchDisjunction;
import edu.upm.dpsm.cims.modules.model.Modules;
import static edu.upm.dpsm.cims.modules.model.Modules.*;

/**
 * SearchLaboratoryModule
 *
 * Description:
 *
```

```
 * author : Miguelino San Miguel
 */
public class SearchLaboratoryModule extends JPASearchModel<Modules> {

  public SearchLaboratoryModule(Modules model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(Modules model) throws Exception {

    setSearch(new SearchCondition()
    .add(new SearchDisjunction()
         .add(TABLE, ERestrictionType.LIKE, "definition",MatchMode.ANYWHERE)
        )

     );
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.modules.service;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.modules.model.Modules;

/**
 * ModulesService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface ModulesService extends BusinessService<Modules> {

  Modules getModuleForTable(String table) throws Exception;
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.modules.service;

import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.modules.dao.ModulesDAO;
import edu.upm.dpsm.cims.modules.model.Modules;

/**
 * ModulesServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class ModulesServiceImpl extends JPAService<Modules> implements ModulesService
{

  @Autowired
  private ModulesDAO dao;

  @Override
  public DAO<Modules> getDAO() {
    return dao;
  }

  @Override
  public Modules getModuleForTable(String table) throws Exception{
    Modules module = new Modules();
    module.setTable(table);
    List<Modules> modulesList = this.search(module);
    return modulesList.size() == 1 ? modulesList.get(0) : null;
  }

}
package edu.upm.dpsm.cims.notetype.dao.impl;

import org.springframework.stereotype.Repository;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.notetype.dao.NoteTypeDAO;
import edu.upm.dpsm.cims.notetype.models.NoteType;

/**
 * @author Miguelino San Miguel
 */
@Repository
public class NoteTypeDAOImpl extends JPADao<NoteType> implements NoteTypeDAO {
```

```
}
package edu.upm.dpsm.cims.notetype.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.notetype.models.NoteType;

/**
 * @author Miguelino San Miguel
 */
public interface NoteTypeDAO extends DAO<NoteType> {

}
package edu.upm.dpsm.cims.notetype.models;

import java.io.Serializable;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "note_type")
@Access(AccessType.FIELD)
public class NoteType extends JPALookupModel implements Serializable {

  private static final long serialVersionUID = 1L;

  @Column(name = "description", updatable = true, length = 50)
  private String description;

  /**
   * @return the description
   */
  public String getDescription() {
    return description;
  }

  /**
   * @param description
   *          the description to set
   */
  public void setDescription(String description) {
    this.description = description;
  }
}
package edu.upm.dpsm.cims.notetype.services.impl;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.notetype.dao.NoteTypeDAO;
import edu.upm.dpsm.cims.notetype.models.NoteType;
import edu.upm.dpsm.cims.notetype.services.NoteTypeService;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional
public class NoteTypeServiceImpl extends JPAService<NoteType> implements
NoteTypeService {

  @Autowired
  private NoteTypeDAO dao;

  @Override
  public DAO<NoteType> getDAO() {
    return this.dao;
  }

}
package edu.upm.dpsm.cims.notetype.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.notetype.models.NoteType;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface NoteTypeService extends BusinessService<NoteType> {

}
package edu.upm.dpsm.cims.physics.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddPhysicsChemicalController {
  @Autowired
  public PhysicsChemicalService physicsChemicalService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "physics/chemical/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated PhysicsChemical physicsChemical,
HttpServletRequest request)
      throws Exception {

    physicsChemicalService.save(physicsChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_DEFINITION_CREATE_SUCCESS).getMap()
;
  }



}
```

```
package edu.upm.dpsm.cims.physics.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
```

```
 * @author Miguelino San Miguel
 */
@Controller
public class GetPhysicsChemicalController {

  @Autowired
  private PhysicsChemicalService physicsChemicalService;

  @RequestMapping(value = "physics/chemical/get", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody PhysicsChemical physicsChemical,
HttpServletRequest request)
      throws Exception {
    PhysicsChemical chemicalItem = physicsChemicalService.get(physicsChemical);

    return DTOUtil.copy(PhysicsChemical.class, chemicalItem, new String[] {
PhysicsChemical.CREATED_BY,
        PhysicsChemical.UPDATED_BY, PhysicsChemical.PHYSICS_CHEMICAL_ITEMS });
  }

}
package edu.upm.dpsm.cims.physics.chemical.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchPhysicsChemicalController {

  @Autowired
  private PhysicsChemicalService physicsChemicalService;

  @RequestMapping(value = "physics/chemical/search", method = RequestMethod.POST)
  public @ResponseBody
  Object chemicalListPage(@RequestBody PhysicsChemical physicsChemical,
HttpServletRequest request)
      throws Exception {
```

```
    List<PhysicsChemical> chemicalList =
physicsChemicalService.search(physicsChemical);

    return DTOUtil.copy(PhysicsChemical.class, chemicalList, new String[] {
PhysicsChemical.CREATED_BY,
        PhysicsChemical.UPDATED_BY, PhysicsChemical.PHYSICS_CHEMICAL_ITEMS });
  }

}
package edu.upm.dpsm.cims.physics.chemical.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import
edu.upm.dpsm.cims.physics.chemical.definition.validation.group.UpdatePhysicsChemical;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class UpdatePhysicsChemicalController {
  @Autowired
  public PhysicsChemicalService physicsChemicalService;

  @RequestMapping(value = "physics/chemical/update-definition", method =
RequestMethod.POST)
  public @ResponseBody
  Object updateChemical(@RequestBody @Validated(UpdatePhysicsChemical.class)
PhysicsChemical physicsChemical,
      HttpServletRequest request) throws Exception {

    PhysicsChemical ccd = new PhysicsChemical();
    ccd.setId(physicsChemical.getId());
    ccd = physicsChemicalService.get(ccd);
    BeanUtils.copyProperties(physicsChemical, ccd, new String[] { PhysicsChemical.ID,
PhysicsChemical.TOTAL_QUANTITY,
        PhysicsChemical.CREATED_BY_ID, PhysicsChemical.CREATED_DATE });
    physicsChemicalService.update(ccd);
```

```
    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_DEFINITION_UPDATE_SUCCESS).getMap()
;
  }
}
package edu.upm.dpsm.cims.physics.chemical.definition.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.FetchMode;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;

import
edu.upm.dpsm.cims.physics.chemical.definition.dao.PhysicsChemicalDefinitionDAO;
import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * @author Miguelino San Miguel
 */
@Repository
public class PhysicsChemicalDefinitionDAOImpl extends JPADao<PhysicsChemical>
implements PhysicsChemicalDefinitionDAO {

  @Override
  public List<PhysicsChemical> search(
      PhysicsChemical chemical) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session.createCriteria(
        PhysicsChemical.class).setFetchMode(
        "createdBy",
        FetchMode.JOIN);
    return super.search(chemical, criteria);
  }
}
package edu.upm.dpsm.cims.physics.chemical.definition.dao;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * @author Miguelino San Miguel
 */
public interface PhysicsChemicalDefinitionDAO extends DAO<PhysicsChemical> {
}
package edu.upm.dpsm.cims.physics.chemical.definition.models;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Access;
```

```
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.validation.groups.Default;

import org.hibernate.validator.constraints.NotEmpty;

import
edu.upm.dpsm.cims.physics.chemical.definition.validation.group.UpdatePhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;

@Entity
@Table(name = "physics_chemical_definition")
@Access(AccessType.FIELD)
public class PhysicsChemical extends JPAEntityModel implements Serializable,
MonitorItem {

  public final static String          NAME                      = "name";
  public final static String          MOL_FORMULA               = "molFormula";
  public final static String          COLOR                     = "color";
  public final static String          TEXTURE                   = "texture";
  public final static String          GRADE                     = "grade";
  public final static String          HYGROSCOPIC               = "hygroscopic";
  public final static String          EXPLOSIVE                 = "explosive";
  public final static String          TOTAL_QUANTITY            = "totalQuantity";
  public final static String          THRESH                    = "thresh";
  public final static String          MINIMUM                   = "minimum";
  public final static String          PHYSICS_CHEMICAL_ITEMS =
"physicsChemicalItems";

  private static final long           serialVersionUID          =
3529801466283219405L;

  @NotEmpty
  @Size(min = 2, max = 50)
  @Column(updatable = true, name = "name")
  private String                      name;

  @Size(min = 2, max = 100, groups = { Default.class, UpdatePhysicsChemical.class })
  @Column(updatable = true, name = "mol_formula")
  private String                      molFormula;

  @Column(updatable = true, name = "color")
```

```
  private String                      color;

  @Column(updatable = true, name = "texture")
  private String                      texture;

  @Column(updatable = true, name = "grade")
  private String                      grade;

  @Column(updatable = true, name = "hygroscopic")
  private Boolean                     hygroscopic;

  @Column(updatable = true, name = "explosive")
  private Boolean                     explosive;

  @Column(updatable = true, name = "total_quantity")
  private Long                        totalQuantity           = Long.valueOf(0L);

  @Column(updatable = true, name = "thresh")
  private Integer                     thresh;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
  private Integer                     minimum;

  @OneToMany(mappedBy = "physicsChemical", cascade = { CascadeType.REMOVE },fetch =
FetchType.LAZY)
  private List<PhysicsChemicalItem> physicsChemicalItems;

  @Transient
  private String                      username;

  public Integer getMinimum() {
    return minimum;
  }

  public void setMinimum(Integer minimum) {
    this.minimum = minimum;
  }

  public Boolean getHygroscopic() {
    return hygroscopic;
  }

  public void setHygroscopic(Boolean hygroscopic) {
    this.hygroscopic = hygroscopic;
  }

  public Boolean getExplosive() {
    return explosive;
  }

  public void setExplosive(Boolean explosive) {
```

```
  this.explosive = explosive;
}

public String getColor() {
  return color;
}

public void setColor(String color) {
  this.color = color;
}

public String getGrade() {
  return grade;
}

public void setGrade(String grade) {
  this.grade = grade;
}

public String getMolFormula() {
  return molFormula;
}

public void setMolFormula(String molFormula) {
  this.molFormula = molFormula;
}

public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public String getTexture() {
  return texture;
}

public void setTexture(String texture) {
  this.texture = texture;
}

public Integer getThresh() {
  return thresh;
}

public void setThresh(Integer thresh) {
  this.thresh = thresh;
}

public Long getTotalQuantity() {
  return totalQuantity;
```

```
  }

  public void setTotalQuantity(Long totalQuantity) {
    this.totalQuantity = totalQuantity;
  }

  public List<PhysicsChemicalItem> getPhysicsChemicalItems() {
    return physicsChemicalItems;
  }

  public void setPhysicsChemicalItems(List<PhysicsChemicalItem> physicsChemicalItems)
{
    this.physicsChemicalItems = physicsChemicalItems;
  }

  public String getUsername() {
    return username;
  }

  public void setUsername(String username) {
    this.username = username;
  }
}
package edu.upm.dpsm.cims.physics.chemical.definition.search;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import static edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical.*;
/**
 * @author Miguelino San Miguel
 */
public class SearchPhysicsChemicals extends JPASearchModel<PhysicsChemical> {

  /**
   * @param request
   * @param model
   * @throws Exception
   */
  public SearchPhysicsChemicals(
      PhysicsChemical model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(
      PhysicsChemical model)
      throws Exception {
    setSearch(new SearchCondition().add(SearchCondition
        .conjunction()
```

```
        .add(
            SearchCondition
                .disjunction()
                .add(NAME, ERestrictionType.LIKE, model.getName(),
MatchMode.ANYWHERE)
                .add(MOL_FORMULA, ERestrictionType.LIKE, model.getMolFormula(),
MatchMode.ANYWHERE))
        .and(CREATED_BY_ID, ERestrictionType.EQ, model.getCreatedById())
        .and(ID, ERestrictionType.EQ, model.getId())
         )
     );
  }
}
package edu.upm.dpsm.cims.physics.chemical.definition.services.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import
edu.upm.dpsm.cims.physics.chemical.definition.dao.PhysicsChemicalDefinitionDAO;
import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.search.SearchPhysicsChemicals;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorSupply;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional(rollbackFor = Exception.class)
public class PhysicsChemicalServiceImpl extends JPAService<PhysicsChemical>
implements
    PhysicsChemicalService {

  @Override
  public PhysicsChemical update(PhysicsChemical t) throws Exception {
    return super.update(t);
  }

  @Override
  public PhysicsChemical save(PhysicsChemical t) throws Exception {

    return super.save(t);
  }

  @Autowired
  private PhysicsChemicalDefinitionDAO dao;
```

```
  @Transactional(readOnly = true)
  @Override
  public List<PhysicsChemical> search(PhysicsChemical chemical) throws Exception {
    Pagination pagination = chemical.getPagination();
    pagination = (pagination == null) ? (new Pagination()) : pagination;
    pagination.setTotalSize(dao.getTotalSize(chemical));
    pagination.setEnable(true);
    chemical.setPagination(pagination);
    SearchPhysicsChemicals scc = new SearchPhysicsChemicals(chemical);
    chemical.setSearchModel(scc);
    return dao.search(chemical);
  }

  @Override
  public DAO<PhysicsChemical> getDAO() {
    return this.dao;
  }

}
package edu.upm.dpsm.cims.physics.chemical.definition.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface PhysicsChemicalService extends BusinessService<PhysicsChemical> {

}
package edu.upm.dpsm.cims.physics.chemical.definition.validation.group;

/**
 * @author Miguelino San Miguel
 */
public interface UpdatePhysicsChemical {

}
package edu.upm.dpsm.cims.physics.chemical.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.physics.chemical.item.services.PhysicsChemicalItemService;
import
edu.upm.dpsm.cims.physics.chemical.item.validation.validator.ExistingPhysicsChemicalD
efinitionValidator;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddPhysicsChemicalItemController {
  @Autowired
  private PhysicsChemicalItemService physicsChemicalDefinitionService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
    binder.addValidators(new ExistingPhysicsChemicalDefinitionValidator());
  }

  @RequestMapping(value = "physics/chemical/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemicalItem(@RequestBody @Validated PhysicsChemicalItem
physicsChemicalItem,
      HttpServletRequest request) throws Exception {

    physicsChemicalDefinitionService.saveItems(physicsChemicalItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.CHEMICAL_ITEM_ADD_SUCCESS).getMap();
  }


}
```

```
package edu.upm.dpsm.cims.physics.chemical.item.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.FetchMode;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;

import edu.upm.dpsm.cims.physics.chemical.item.dao.PhysicsChemicalItemDAO;
import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
```

```
 * @author Miguelino San Miguel
 */
@Repository
public class PhysicsChemicalItemDAOImpl extends JPADao<PhysicsChemicalItem>implements
PhysicsChemicalItemDAO {

  @Override
  public List<PhysicsChemicalItem> search(PhysicsChemicalItem chemicalItem)
      throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session
        .createCriteria(PhysicsChemicalItem.class)
        .setFetchMode(
            "createdBy",
            FetchMode.JOIN)
        .createAlias(PhysicsChemicalItem.PHYSICS_CHEMICAL_DEFINITION,
            PhysicsChemicalItem.PHYSICS_CHEMICAL_DEFINITION)
        .setFetchMode(PhysicsChemicalItem.PHYSICS_CHEMICAL_DEFINITION,
            FetchMode.JOIN);

    // apply search parameters e.g. pagination
    return super.search(chemicalItem, criteria);
  }
}
package edu.upm.dpsm.cims.physics.chemical.item.dao;

import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * @author Miguelino San Miguel
 */
public interface PhysicsChemicalItemDAO extends DAO<PhysicsChemicalItem> {

}
package edu.upm.dpsm.cims.physics.chemical.item.models;

import java.io.Serializable;
import java.sql.Timestamp;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.NotEmpty;
```

```
import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.notetype.models.NoteType;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "physics_chemical_item")
@Access(AccessType.FIELD)
public class PhysicsChemicalItem extends JPAEntityModel implements Serializable{

  private static final long  serialVersionUID             = -392215716780171378L;

  public final static String CHEMICAL_ID                  = "chemicalId";
  public final static String CHEMICAL_NAME                = "chemicalName";
  public final static String STATUS                       = "consumed";
  public final static String EXPIRATION_DATE              = "expDate";
  public final static String NOTE_TYPE_ID                 = "noteTypeId";
  public final static String NOTE_TYPE                    = "noteType";
  public static final String PHYSICS_CHEMICAL_DEFINITION = "physicsChemical";

  @NotNull
  @Column(updatable = false, name = "chemical_id")
  private Long               chemicalId;

  @NotEmpty
  @Transient
  private String             chemicalName;

  @Column(updatable = true, name = "consumed")
  private Boolean           consumed;

  @Column(name = "note_type")
  private Long               noteTypeId;

  @ManyToOne
  @JoinColumn(name = "note_type", updatable = false, insertable = false)
  private NoteType           noteType;

  @Column(updatable = true, name = "exp_date")
  private Timestamp          expDate;

  @Min(value = 1)
  @Transient
  private Integer            quantity                     = 1;

  @ManyToOne(fetch = FetchType.LAZY)
  @JoinColumn(name = "chemical_id", updatable = false, insertable = false)
  private PhysicsChemical  physicsChemical;

  public Long getChemicalId() {
```

```
  return chemicalId;
}

public void setChemicalId(Long chemicalId) {
  this.chemicalId = chemicalId;
}

public Boolean getConsumed() {
  return consumed;
}

public void setConsumed(Boolean consumed) {
  this.consumed = consumed;
}

public Long getNoteTypeId() {
  return noteTypeId;
}

public void setNoteTypeId(Long noteType) {
  this.noteTypeId = noteType;
}

public NoteType getNoteType() {
  return noteType;
}

public void setNoteType(NoteType noteType) {
  this.noteType = noteType;
}

public Timestamp getExpDate() {
  return expDate;
}

/**
 * @param expDate
 *          the expDate to set
 */
public void setExpDate(Timestamp expDate) {
  this.expDate = expDate;
}

public PhysicsChemical getPhysicsChemicalDefinition() {
  return physicsChemical;
}

public void setPhysicsChemicalDefinition(PhysicsChemical physicsChemical) {
  this.physicsChemical = physicsChemical;
}

public String getChemicalName() {
  return chemicalName;
```

```
  }

  public void setChemicalName(String chemicalName) {
    this.chemicalName = chemicalName;
  }

  public Integer getQuantity() {
    return quantity;
  }

  public void setQuantity(Integer quantity) {
    this.quantity = quantity;
  }
}
package edu.upm.dpsm.cims.physics.chemical.item.search;

import javax.servlet.http.HttpServletRequest;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;

/**
 * @author Miguelino San Miguel
 */
public class SearchChemicalItems extends JPASearchModel<PhysicsChemicalItem> {

  /**
   * @param request
   * @param model
   * @throws Exception
   */
  public SearchChemicalItems(PhysicsChemicalItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(PhysicsChemicalItem model) throws Exception {

    setSearch(new SearchCondition()
        .add(PhysicsChemicalItem.PHYSICS_CHEMICAL_DEFINITION + "." +
PhysicsChemical.NAME,
            ERestrictionType.LIKE, model.getChemicalName(), MatchMode.START)
        .add(PhysicsChemicalItem.EXPIRATION_DATE, ERestrictionType.LE,
model.getExpDate())
        .add(PhysicsChemicalItem.CREATED_BY_ID, ERestrictionType.EQ,
model.getCreatedById())
        .add(PhysicsChemicalItem.NOTE_TYPE_ID, ERestrictionType.EQ,
model.getNoteTypeId()));
```

```
  }
}
package edu.upm.dpsm.cims.physics.chemical.item.services.impl;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.physics.chemical.item.dao.PhysicsChemicalItemDAO;
import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.physics.chemical.item.search.SearchChemicalItems;
import edu.upm.dpsm.cims.physics.chemical.item.services.PhysicsChemicalItemService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional
public class PhysicsChemicalItemServiceImpl extends JPAService<PhysicsChemicalItem>
implements
    PhysicsChemicalItemService {

  @Autowired
  private PhysicsChemicalItemDAO           dao;

  @Autowired
  private PhysicsChemicalService physicsChemicalService;

  @Override
  public List<PhysicsChemicalItem> search(PhysicsChemicalItem chemicalItem) throws
Exception {
    Pagination pagination = chemicalItem.getPagination();
    if (pagination == null) {
      pagination = new Pagination();
      chemicalItem.setPagination(pagination);
    }

    SearchChemicalItems scc = new SearchChemicalItems(

    chemicalItem);
    chemicalItem.setSearchModel(scc);
    pagination.setTotalSize(dao.getTotalSize(chemicalItem));
    pagination.setEnable(true);
    return this.dao.search(chemicalItem);
  }
```

```
  @Override
  public DAO<PhysicsChemicalItem> getDAO() {
    return this.dao;
  }

  @Override
  public PhysicsChemicalItem saveItems(PhysicsChemicalItem physicsChemicalItem)
throws Exception {
    int quantity = physicsChemicalItem.getQuantity();
    physicsChemicalItem.setConsumed(true);
    int row = quantity;
    while (row > 0) {
      PhysicsChemicalItem item = new PhysicsChemicalItem();
      PropertyUtils.copyProperties(item, physicsChemicalItem);
      super.save(item);
      row--;
    }

    // update parent ChemicalDefinition total quantity
    PhysicsChemical ccd = new PhysicsChemical();
    ccd.setId(physicsChemicalItem.getChemicalId());
    ccd = physicsChemicalService.get(ccd);
    ccd.setTotalQuantity(ccd.getTotalQuantity() + quantity);
    physicsChemicalService.update(ccd);

    return physicsChemicalItem;
  }

  @Override
  public PhysicsChemicalItem save(PhysicsChemicalItem t) throws Exception {
    return super.save(t);
  }
}
package edu.upm.dpsm.cims.physics.chemical.item.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface PhysicsChemicalItemService extends
BusinessService<PhysicsChemicalItem> {
  public  PhysicsChemicalItem saveItems(
      PhysicsChemicalItem physicsChemicalItem) throws Exception;
}
package edu.upm.dpsm.cims.physics.chemical.item.validation.group;

/**
```

```
 * @author Miguelino San Miguel
 *
 */
public interface UpdateChemicalItem {

}
package edu.upm.dpsm.cims.physics.chemical.item.validation.validator;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.physics.chemical.definition.models.PhysicsChemical;
import edu.upm.dpsm.cims.physics.chemical.definition.services.PhysicsChemicalService;
import edu.upm.dpsm.cims.physics.chemical.item.models.PhysicsChemicalItem;
import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.ObjectUtils;

/**
 * @author Miguelino San Miguel
 */
@Component
public class ExistingPhysicsChemicalDefinitionValidator implements Validator {
      @Autowired
      private MessageSource messageSource;

      @Autowired
      private PhysicsChemicalService physicsChemicalService;
  @Override
  public boolean supports(Class<?> clazz) {
    return PhysicsChemicalItem.class.equals(clazz);
  }

  @Override
  public void validate(Object target, Errors errors) {
    try {
      PhysicsChemicalItem physicsChemicalItem = (PhysicsChemicalItem) target;
      Long chemicalId = physicsChemicalItem.getChemicalId();
      if (!ObjectUtils.isEmpty(chemicalId)) {
        BindingResult result = (BindingResult) errors;
        PhysicsChemical existingDef = new PhysicsChemical();
        existingDef.setId(chemicalId);
        existingDef = physicsChemicalService.get(existingDef);
        if (existingDef == null) {
          ValidationUtils.addBindingError(result,

PhysicsChemicalItem.CHEMICAL_NAME,EValidationMessages.NON_EXISTING_CHEMICAL_NAME);
        }
      }
```

```
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

}
package edu.upm.dpsm.cims.physics.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddPhysicsLabwareController {
  @Autowired
  private PhysicsLabwareService physicsLabwareService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "physics/labware/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated PhysicsLabware physicsChemical,
HttpServletRequest request)
      throws Exception {

    physicsLabwareService.save(physicsChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_CREATE_SUCCESS).getMap();
  }
}
package edu.upm.dpsm.cims.physics.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddPhysicsLabwareController {
  @Autowired
  private PhysicsLabwareService physicsLabwareService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "physics/labware/add", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddChemical(@RequestBody @Validated PhysicsLabware physicsChemical,
HttpServletRequest request)
      throws Exception {

    physicsLabwareService.save(physicsChemical);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_CREATE_SUCCESS).getMap();
  }
}
package edu.upm.dpsm.cims.physics.labware.definition.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
```

```
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;
import edu.upm.dpsm.cims.core.util.DTOUtil;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchPhysicsLabwareController {

  @Autowired
  private PhysicsLabwareService physicsLabwareService;

  @RequestMapping(value = "physics/labware/search", method = RequestMethod.POST)
  public @ResponseBody
  Object labwareListPage(@RequestBody PhysicsLabware physicsLabware,
HttpServletRequest request)
      throws Exception {
    List<PhysicsLabware> chemicalList = physicsLabwareService.search(physicsLabware);

    return DTOUtil.copy(PhysicsLabware.class, chemicalList, new String[] {
PhysicsLabware.CREATED_BY,
        PhysicsLabware.UPDATED_BY, PhysicsLabware.PHYSICS_LABWARE_ITEMS });
  }

}
package edu.upm.dpsm.cims.physics.labware.definition.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class UpdatePhysicsLabwareController {
  @Autowired
  public PhysicsLabwareService physicsLabwareService;

  @RequestMapping(value = "physics/labware/update-definition", method =
RequestMethod.POST)
```

```
  public @ResponseBody
  Object updateChemical(@RequestBody @Validated PhysicsLabware physicsLabware,
HttpServletRequest request)
      throws Exception {

    PhysicsLabware ccd = new PhysicsLabware();
    ccd.setId(physicsLabware.getId());
    ccd = physicsLabwareService.get(ccd);
    BeanUtils.copyProperties(physicsLabware, ccd, new String[] { PhysicsLabware.ID,
PhysicsLabware.TOTAL_QUANTITY,
        PhysicsLabware.OK_QUANTITY, PhysicsLabware.CREATED_BY_ID,
PhysicsLabware.CREATED_DATE });
    physicsLabwareService.update(ccd);
    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_DEFINITION_UPDATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.physics.labware.definition.dao.impl;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.labware.definition.dao.PhysicsLabwareDAO;
import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * PhysicsLabwareDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class PhysicsLabwareDAOImpl extends JPADao<PhysicsLabware> implements
PhysicsLabwareDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
```

```
package edu.upm.dpsm.cims.physics.labware.definition.dao;

import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * PhysicsLabwareDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface PhysicsLabwareDAO extends DAO<PhysicsLabware> {

}
package edu.upm.dpsm.cims.physics.labware.definition.models;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.maintenancerequest.aop.model.MonitorItem;

/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "physics_labware_definition")
@Access(AccessType.FIELD)
public class PhysicsLabware extends JPAEntityModel implements MonitorItem,
Serializable {

  private static final long        serialVersionUID      = -7617390104822472391L;
  public final static String       NAME                  = "name";
  public final static String       TYPE                  = "type";
  public final static String       SIZE                  = "size";
  public final static String       OK_QUANTITY           = "okQuantity";
  public final static String       BROKEN_QUANTITY       = "brokenQuantity";
  public final static String       STAINED_QUANTITY      = "stainedQuantity";
  public final static String       TOTAL_QUANTITY        = "totalQuantity";
  public final static String       MINIMUM               = "minimum";
```

```
  public final static String       PHYSICS_LABWARE_ITEMS = "physicsLabwareItems";

  @Column(name = "type")
  private String                   type;

  @Column(name = "size")
  private String                   size;

  @Column(name = "name")
  private String                   name;

  @Column(name = "total_quantity")
  private Long                     totalQuantity         = Long.valueOf(0L);      ;

  @Column(name = "broken_quantity")
  private Long                     brokenQuantity        = Long.valueOf(0L);      ;

  @Column(name = "stained_quantity")
  private Long                     stainedQuantity       = Long.valueOf(0L);      ;

  @Column(name = "ok_quantity")
  private Long                     okQuantity            = Long.valueOf(0L);      ;

  @NotNull
  @Min(0)
  @Column(updatable = true, name = "minimum")
  private Integer                  minimum;

  @OneToMany(mappedBy = "physicsLabware", cascade = { CascadeType.REMOVE }, fetch =
FetchType.LAZY)
  private List<PhysicsLabwareItem> physicsLabwareItems;

  public List<PhysicsLabwareItem> getPhysicsLabwareItems() {
    return physicsLabwareItems;
  }

  public void setPhysicsLabwareItems(List<PhysicsLabwareItem> physicsLabwareItems) {
    this.physicsLabwareItems = physicsLabwareItems;
  }

  public String getType() {
    return type;
  }

  public void setType(String type) {
    this.type = type;
  }

  public String getSize() {
    return size;
  }

  public void setSize(String size) {
```

```
  this.size = size;
}

public Long getTotalQuantity() {
  return totalQuantity;
}

public void setTotalQuantity(Long totalQuantity) {
  this.totalQuantity = totalQuantity;
}

public Long getBrokenQuantity() {
  return brokenQuantity;
}

public void setBrokenQuantity(Long brokenQuantity) {
  this.brokenQuantity = brokenQuantity;
}

public Long getStainedQuantity() {
  return stainedQuantity;
}

public void setStainedQuantity(Long stainedQuantity) {
  this.stainedQuantity = stainedQuantity;
}

public Long getOkQuantity() {
  return okQuantity;
}

public void setOkQuantity(Long okQuantity) {
  this.okQuantity = okQuantity;
}

@Override
public String getName() {
  return this.name;
}

@Override
public Integer getMinimum() {
  return this.minimum;
}

public void setName(String name) {
  this.name = name;
}

public void setMinimum(Integer minimum) {
  this.minimum = minimum;
}
```

```
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 4, 2014
 */
package edu.upm.dpsm.cims.physics.labware.definition.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.labware.definition.dao.PhysicsLabwareDAO;
import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;

/**
 * PhysicsLabwareServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class PhysicsLabwareServiceImpl extends JPAService<PhysicsLabware> implements
PhysicsLabwareService {

  @Autowired
  private PhysicsLabwareDAO dao;

  @Override
  public DAO<PhysicsLabware> getDAO() {
    return dao;
  }

}
package edu.upm.dpsm.cims.physics.labware.definition.services;

import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */

public interface PhysicsLabwareService extends BusinessService<PhysicsLabware> {

}
```

```
package edu.upm.dpsm.cims.physics.labware.item.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.physics.labware.item.services.PhysicsLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddPhysicsLabwareItemController {
  @Autowired
  private PhysicsLabwareItemService physicsLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "physics/labware/add-item", method = RequestMethod.POST)
  public @ResponseBody
  Object doAction(@RequestBody @Validated PhysicsLabwareItem physicsLabwareItem,
      HttpServletRequest request) throws Exception {

    physicsLabwareItemService.saveItems(physicsLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.LABWARE_ITEM_ADD_SUCCESS).getMap();
  }


}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.controllers;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;
import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.physics.labware.item.services.PhysicsLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreatePhysicsLabwareItemCondemnationAlertController {
  @Autowired
  private PhysicsLabwareItemService physicsLabwareItemService;

  @MonitorRequest(status=EItemStatus.CONDEMNATION)
  @RequestMapping(value = "physics/labware/item/condemnation-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody PhysicsLabwareItem physicsLabwareItem,
HttpServletRequest request)
      throws Exception {

    physicsLabwareItemService.createCondemnationAlert(physicsLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.CONDEMNATION_REQUEST_CREATE_SUCCESS).getMap(
);
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.controllers;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.itemstatus.model.EItemStatus;
import edu.upm.dpsm.cims.maintenancerequest.aop.annotation.MonitorRequest;
import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.physics.labware.item.services.PhysicsLabwareItemService;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;

/**
 * CreateChemLabwareItemRepairAlertController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class CreatePhysicsLabwareItemRepairAlertController {
  @Autowired
  private PhysicsLabwareItemService physicsLabwareItemService;

  @MonitorRequest(status=EItemStatus.REPAIR)
  @RequestMapping(value = "physics/labware/item/repair-alert/create", method =
RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody PhysicsLabwareItem physicsLabwareItem,
HttpServletRequest request)
      throws Exception {

    physicsLabwareItemService.createRepairAlert(physicsLabwareItem);

    return
JSONMapUtils.createJSONSuccess(EMessages.REPAIR_REQUEST_CREATE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.controllers;

import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.physics.labware.item.services.PhysicsLabwareItemService;

/**
 * SearchChemicalLabwareItemController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class SearchPhysicsChemicalLabwareItemController {
  @Autowired
  private PhysicsLabwareItemService physicsLabwareItemService;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
  }

  @RequestMapping(value = "physics/labware/item/search", method = RequestMethod.POST)
  public @ResponseBody
  Object doAddLabwareItem(@RequestBody PhysicsLabwareItem physicsLabwareItem,
HttpServletRequest request)
      throws Exception {

    List<PhysicsLabwareItem> list =
physicsLabwareItemService.searchLabwareItem(physicsLabwareItem);

    return DTOUtil.copy(PhysicsLabwareItem.class, list, new String[] {
PhysicsLabwareItem.CREATED_BY,
        PhysicsLabwareItem.UPDATED_BY, PhysicsLabwareItem.PHYSICS_LABWARE });
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.dao.impl;
```

```
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.labware.item.dao.PhysicsLabwareItemDAO;
import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;

/**
 * PhysicsLabwareItemDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class PhysicsLabwareItemDAOImpl extends JPADao<PhysicsLabwareItem> implements
PhysicsLabwareItemDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.dao;

import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.core.app.dao.DAO;

/**
 * PhysicsLabwareItemDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface PhysicsLabwareItemDAO extends DAO<PhysicsLabwareItem> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.models;

import java.io.Serializable;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.maintenancerequest.model.Maintainable;
import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;

/**
 * PhysicsLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "physics_labware_item")
public class PhysicsLabwareItem extends JPAEntityModel implements Serializable,
Maintainable {
  public final static String LABWARE_ID        = "labwareId";
  public final static String PHYSICS_LABWARE = "physicsLabware";
  private static final long  serialVersionUID  = 3011167584400262957L;

  @Column(name = "labware_id")
  private Long               labwareId;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "labware_id", updatable = false, insertable = false)
  private PhysicsLabware   physicsLabware;

  @Column(name = "labware_status")
  private Integer            labwareStatus;

  @NotEmpty
  @Transient
  private String             labwareName;

  @Transient
  private Integer            quantity;

  @Column(name="serial_number")
  private String serialNumber;

  public String getSerialNumber() {
    return serialNumber;
  }
```

```
  public void setSerialNumber(String serialNumber) {
    this.serialNumber = serialNumber;
  }

  public Integer getQuantity() {
    return quantity;
  }

  public void setQuantity(Integer quantity) {
    this.quantity = quantity;
  }

  public String getLabwareName() {
    return labwareName;
  }

  public void setLabwareName(String labwareName) {
    this.labwareName = labwareName;
  }

  public Long getLabwareId() {
    return labwareId;
  }

  public void setLabwareId(Long labwareId) {
    this.labwareId = labwareId;
  }

  public PhysicsLabware getPhysicsLabware() {
    return physicsLabware;
  }

  public void setPhysicsLabware(PhysicsLabware physicsLabware) {
    this.physicsLabware = physicsLabware;
  }

  public Integer getLabwareStatus() {
    return labwareStatus;
  }

  public void setLabwareStatus(Integer labwareStatus) {
    this.labwareStatus = labwareStatus;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.search;
```

```
import static
edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem.LABWARE_ID;
import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
/**
 * SearchPhysicsLabwareItem
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public class SearchPhysicsLabwareItem extends JPASearchModel<PhysicsLabwareItem> {

  public SearchPhysicsLabwareItem(PhysicsLabwareItem model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(PhysicsLabwareItem model) throws Exception {
   setSearch(new SearchCondition()
     .add(LABWARE_ID,ERestrictionType.EQ, model.getLabwareId()));
  }


}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 5, 2014
 */
package edu.upm.dpsm.cims.physics.labware.item.services.impl;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.physics.labware.definition.models.PhysicsLabware;
import edu.upm.dpsm.cims.physics.labware.definition.services.PhysicsLabwareService;
import edu.upm.dpsm.cims.physics.labware.item.dao.PhysicsLabwareItemDAO;
import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.physics.labware.item.search.SearchPhysicsLabwareItem;
import edu.upm.dpsm.cims.physics.labware.item.services.PhysicsLabwareItemService;
import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.labstatus.model.ELabwareStatus;
```

```
import edu.upm.dpsm.cims.maintenancerequest.service.MaintenanceRequestService;
import edu.upm.dpsm.cims.modules.model.EModules;

/**
 * PhysicsLabwareItemServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class PhysicsLabwareItemServiceImpl extends JPAService<PhysicsLabwareItem>
implements PhysicsLabwareItemService {

  @Autowired
  private PhysicsLabwareItemDAO     dao;

  @Autowired
  private PhysicsLabwareService     physicsLabwareService;

  @Autowired
  private MaintenanceRequestService maintenanceRequestService;

  @Override
  public DAO<PhysicsLabwareItem> getDAO() {
    return dao;
  }

  @Override
  public PhysicsLabwareItem saveItems(PhysicsLabwareItem physicsLabwareItem) throws
Exception {
    int quantity = physicsLabwareItem.getQuantity();
    physicsLabwareItem.setLabwareStatus(ELabwareStatus.OK.asInt());
    int row = quantity;
    while (row > 0) {
      PhysicsLabwareItem item = new PhysicsLabwareItem();
      PropertyUtils.copyProperties(item, physicsLabwareItem);
      super.save(item);
      row--;
    }

    // update parent LabwareDefinition total quantity
    PhysicsLabware ccd = new PhysicsLabware();
    ccd.setId(physicsLabwareItem.getLabwareId());
    ccd = physicsLabwareService.get(ccd);
    ccd.setTotalQuantity(ccd.getTotalQuantity() + quantity);
    ccd.setOkQuantity(ccd.getOkQuantity() + quantity);
    physicsLabwareService.update(ccd);

    return physicsLabwareItem;
  }
```

```
  @Override
  public List<PhysicsLabwareItem> searchLabwareItem(PhysicsLabwareItem
physicsLabwareItem) throws Exception {
    SearchPhysicsLabwareItem searchLabwareItem = new
SearchPhysicsLabwareItem(physicsLabwareItem);
    physicsLabwareItem.setSearchModel(searchLabwareItem);
    return super.search(physicsLabwareItem);
  }

  @Override
  public PhysicsLabwareItem createRepairAlert(PhysicsLabwareItem physicsLabwareItem)
throws Exception {
    PhysicsLabwareItem labwareItem = super.get(physicsLabwareItem);
    String labwareName = labwareItem.getPhysicsLabware().getName();
    maintenanceRequestService.createRepairAlert(labwareItem, labwareName,
EModules.PHYSICS_LABWARE_ITEM.asLong());
    return labwareItem;
  }

  @Override
  public PhysicsLabwareItem createCondemnationAlert(PhysicsLabwareItem
physicsLabwareItem) throws Exception {
    PhysicsLabwareItem labwareItem = super.get(physicsLabwareItem);
    String labwareName = labwareItem.getPhysicsLabware().getName();
    maintenanceRequestService.createCondemnationAlert(labwareItem, labwareName,
EModules.PHYSICS_LABWARE_ITEM.asLong());
    return labwareItem;
  }

}
package edu.upm.dpsm.cims.physics.labware.item.services;

import java.util.List;

import edu.upm.dpsm.cims.physics.labware.item.models.PhysicsLabwareItem;
import edu.upm.dpsm.cims.core.app.service.BusinessService;

/**
 * @author Miguelino San Miguel
 */
public interface PhysicsLabwareItemService extends
BusinessService<PhysicsLabwareItem> {
  public PhysicsLabwareItem saveItems(PhysicsLabwareItem physicsLabwareItem) throws
Exception;

  public List<PhysicsLabwareItem> searchLabwareItem(PhysicsLabwareItem
physicsLabwareItem) throws Exception;

  public PhysicsLabwareItem createRepairAlert(PhysicsLabwareItem physicsLabwareItem)
throws Exception;

  public PhysicsLabwareItem createCondemnationAlert(PhysicsLabwareItem
physicsLabwareItem) throws Exception;
```

```
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 3, 2014
 */
package edu.upm.dpsm.cims.requeststatus.model;

/**
 * ERequestStatus
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum ERequestStatus {
  ALERT(1),
  UNDER_REQUEST(2),
  APPROVED(3),
  REJECTED(4),
  CLEAR(5);

  private int status;

  private ERequestStatus(int status) {
    this.status = status;
  }

  public static ERequestStatus valueOf(int statusInt) {
    for (ERequestStatus status : ERequestStatus.values()) {
      if (status.asInt() == statusInt) {
        return status;
      }
    }

    throw new RuntimeException("No matching ERequestStatus for the given integer");
  }

  public int asInt() {
    return this.status;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.controller;
```

```
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.chemistry.chemical.definition.models.ChemistryChemical;
import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.semester.model.AcademicYear;
import edu.upm.dpsm.cims.semester.service.AcademicYearService;

/**
 * GetAcademicYearListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetAcademicYearListController {
  @Autowired
  private AcademicYearService academicYearService;

  @RequestMapping(value = "academic-year/list", method = RequestMethod.POST)
  public @ResponseBody
  Object semesterListPage(HttpServletRequest request) throws Exception {
    return DTOUtil.copy(AcademicYear.class, academicYearService.search(new
AcademicYear()), new String[] {
        ChemistryChemical.CREATED_BY, ChemistryChemical.UPDATED_BY });
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.semester.model.Semester;
import edu.upm.dpsm.cims.semester.service.SemesterService;
```

```
/**
 * GetSemesterListController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetSemesterListController {

  @Autowired
  private SemesterService semesterService;

  @RequestMapping(value = "semester/list", method = RequestMethod.POST)
  public @ResponseBody
  Object semesterListPage(HttpServletRequest request) throws Exception {
    return semesterService.search(new Semester());
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.semester.model.AcademicYear;

/**
 * AcademicYearDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface AcademicYearDAO extends DAO<AcademicYear> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.dao;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
```

```
import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.semester.model.AcademicYear;

/**
 * AcademicYearDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class AcademicYearDAOImpl extends JPADao<AcademicYear> implements
AcademicYearDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.semester.model.Semester;

/**
 * SemesterDAO
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface SemesterDAO extends DAO<Semester> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.dao;

import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.semester.model.Semester;
```

```
/**
 * SemesterDAOImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Repository
@Transactional
public class SemesterDAOImpl extends JPADao<Semester> implements SemesterDAO {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * AcademicYear
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name = "academic_year")
public class AcademicYear extends JPALookupModel {
  @Column(name = "start_year")
  private Long startYear;
  @Column(name = "end_year")
  private Long endYear;

  public Long getStartYear() {
    return startYear;
  }

  public void setStartYear(Long startYear) {
    this.startYear = startYear;
  }

  public Long getEndYear() {
    return endYear;
  }
```

```
  public void setEndYear(Long endYear) {
    this.endYear = endYear;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;

/**
 * Semester
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Entity
@Table(name="semester")
public class Semester extends JPALookupModel {

  @Column(name="name")
  private String name;


  public String getName() {
    return name;
  }


  public void setName(String name) {
    this.name = name;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.service;
```

```
import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.semester.model.AcademicYear;

/**
 * AcademicYearService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface AcademicYearService extends BusinessService<AcademicYear> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.semester.dao.AcademicYearDAO;
import edu.upm.dpsm.cims.semester.model.AcademicYear;

/**
 * AcademicYearServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class AcademicYearServiceImpl extends JPAService<AcademicYear> implements
AcademicYearService {

  @Autowired
  private AcademicYearDAO dao;
  @Override
  public DAO<AcademicYear> getDAO() {
    return dao;
  }

}
/**
 * LIMS
```

```
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.service;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.semester.model.Semester;

/**
 * SemesterService
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public interface SemesterService extends BusinessService<Semester> {

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 2, 2014
 */
package edu.upm.dpsm.cims.semester.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.semester.dao.SemesterDAO;
import edu.upm.dpsm.cims.semester.model.Semester;

/**
 * SemesterServiceImpl
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Service
@Transactional
public class SemesterServiceImpl extends JPAService<Semester> implements
SemesterService {
  @Autowired
  private SemesterDAO dao;

  @Override
```

```
  public DAO<Semester> getDAO() {
    return dao ;
  }

}
package edu.upm.dpsm.cims.settings.controllers;

import javax.validation.ValidationException;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.exception.AuthorizationException;
import edu.upm.dpsm.cims.core.html.HtmlConstants;
import edu.upm.dpsm.cims.core.html.json.ErrorStatus;
import edu.upm.dpsm.cims.core.html.json.JSONErrorMap;

@ControllerAdvice
public class ApplicationExceptionHandler {

  @ExceptionHandler({ Exception.class })
  @ResponseBody
  @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
  public Object handlePersonNotFound(Exception pe) {
    return new JSONErrorMap(ErrorStatus.SERVER, pe.getMessage()).getMap();
  }

  @ExceptionHandler({ AuthorizationException.class })
  @ResponseBody
  @ResponseStatus(value = HttpStatus.UNAUTHORIZED)
  public Object handleAuthException(AuthorizationException pe) {
    return new JSONErrorMap(ErrorStatus.AUTH,
pe.getMessage()).addProp(HtmlConstants.AUTH_TYPE, pe.getAuthType())
        .getMap();
  }

  @ExceptionHandler({ ValidationException.class })
  @ResponseBody
  @ResponseStatus(value = HttpStatus.BAD_REQUEST)
  public Object handleValidationException(ValidationException ve) {
    return new JSONErrorMap(ErrorStatus.SERVER, ve.getMessage()).getMap();
  }

  @ExceptionHandler({ MethodArgumentNotValidException.class })
  @ResponseBody
  @ResponseStatus(value = HttpStatus.BAD_REQUEST)
  public Object handleValidationException(MethodArgumentNotValidException pe) {
    return ValidationUtils.toMap(pe.getBindingResult());
```

```
  }

}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 24, 2014
 */
package edu.upm.dpsm.cims.settings.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.InitBinder;

/**
 * ControllerValidatorHandler
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@ControllerAdvice
public class ControllerValidatorHandler {
  @Autowired
  private LocalValidatorFactoryBean validator;

  @InitBinder
  protected void bind(WebDataBinder binder) {
    binder.setValidator(validator);
  }
}
package edu.upm.dpsm.cims.settings.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * @author Miguelino San Meekly
 */
@Controller
public class HomeController {

  @RequestMapping(value = "home", method = RequestMethod.GET)
  public String home(HttpServletRequest request, Model model) {
    return "app/home/home.html";
```

```
  }
}
package edu.upm.dpsm.cims.settings.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class IndexController {

	@RequestMapping(value = { "", "index" }, method = RequestMethod.GET)
	public String index() throws Exception {
		return "index";
	}
}
package edu.upm.dpsm.cims.settings.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.settings.forms.LoginForm;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import edu.upm.dpsm.cims.useraccount.validation.validators.ExistingLoginValidator;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class LoginController {

  @Autowired
  private UserAccountService     userAccountService;

  @Autowired
  private ExistingLoginValidator existingLoginValidator;
```

```
  @InitBinder
  protected void init(WebDataBinder dataBinder) {
    dataBinder.addValidators(existingLoginValidator);
  }

  @RequestMapping(value = "login", method = RequestMethod.POST)
  public @ResponseBody
  Object doLogin(@RequestBody @Valid LoginForm loginForm, HttpServletRequest request)
throws Exception {
    UserAccount ua = DTOUtil.copy(UserAccount.class, loginForm);
    UserAccount userAccount = userAccountService.login(ua);
    HttpSession session = request.getSession(true);
    session.setAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION,
userAccount.getId());
    return new JSONSuccessMap().addProp("user",
        DTOUtil.copy(UserAccount.class, userAccount, new String[] {
UserAccount.CREATED_BY, UserAccount.UPDATED_BY }))
        .getMap();
  }
}
package edu.upm.dpsm.cims.settings.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;

/**
 * @author Miguelino San Miguel
 *
 */
@Controller
public class LogoutController {

  @RequestMapping(value = "logout", method = RequestMethod.POST)
  public @ResponseBody
  Object doLogout(HttpServletRequest request) {
    request.getSession(false).invalidate();
    return new JSONSuccessMap().getMap();
  }
}
/**
 * CIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Jan 25, 2014
 */
```

```
package edu.upm.dpsm.cims.settings.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.exception.AuthorizationException;
import edu.upm.dpsm.cims.core.exception.EAuthType;
import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.core.util.HttpSessionUtils;

/**
 * SessionHeartBeatController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class SessionHeartBeatController {

  @RequestMapping(value = "heartbeat", method = RequestMethod.POST)
  public @ResponseBody
  Object doLogin(HttpServletRequest request) throws Exception {
    HttpSession session = request.getSession(false);
    if (!HttpSessionUtils.exists(session, ProjectConstants.USER_ACCOUNT_ID_SESSION))
{
      throw new AuthorizationException(EValidationMessages.LOGIN_AUTHORIZATION,
EAuthType.LOGIN);
    }

    return new JSONSuccessMap().getMap();
  }
}
package edu.upm.dpsm.cims.settings.forms;

import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.core.ProjectConstants;

/**
 * @author Miguelino San Miguel
 */
```

```
public class LoginForm {
  @NotEmpty
  @Size(min = 5, max = 15)
  @Pattern(regexp = ProjectConstants.PATTERN_ALPHA_NUMERIC_SEPARATOR)
  private String username;

  @NotEmpty
  @Size(min = 6, max = 15)
  @Pattern(regexp = ProjectConstants.PATTERN_ALPHA_NUMERIC_SEPARATOR)
  private String password;

  /**
   * @return the username
   */
  public String getUsername() {
    return username;
  }

  /**
   * @param username
   *          the username to set
   */
  public void setUsername(String username) {
    this.username = username;
  }

  /**
   * @return the password
   */
  public String getPassword() {
    return password;
  }

  /**
   * @param password
   *          the password to set
   */
  public void setPassword(String password) {
    this.password = password;
  }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.validation.ValidationException;
import javax.validation.groups.Default;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
```

```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.RoleService;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import
edu.upm.dpsm.cims.useraccount.validation.validators.AvailableUsernameValidator;
import edu.upm.dpsm.cims.useraccount.validation.validators.ConfirmPasswordValidator;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AddUserAccountController {

	@Autowired
	public UserAccountService userAccountService;
	@Autowired
	public MessageSource messageSource;
	@Autowired
	public RoleService roleService;

	@Autowired
	private AvailableUsernameValidator availableUsernameValidator;

	@Autowired
	private ConfirmPasswordValidator confirmPasswordValidator;

	@InitBinder
	protected void initBinder(WebDataBinder binder) {
		binder.addValidators(availableUsernameValidator,
				confirmPasswordValidator);
	}

	@RequestMapping(value = "addUser", method = RequestMethod.POST)
	public @ResponseBody
	Object addUserAccount(
			@RequestBody @Validated(Default.class) UserAccount userAccount,
			HttpServletRequest request) throws Exception {
		UserAccount ua = userAccountService.save(userAccount);

		if (ua == null) {
			throw new ValidationException(messageSource.getMessage(
					"Failed.userAccount.create", null, "", null));
		}

		return new JSONSuccessMap(messageSource.getMessage(
				"Success.userAccount.create", null, "", null)).getMap();
```

```
	}

}
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.RoleService;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import edu.upm.dpsm.cims.useraccount.validation.groups.ChangePassword;
import edu.upm.dpsm.cims.useraccount.validation.validators.ConfirmPasswordValidator;
import edu.upm.dpsm.cims.useraccount.validation.validators.OldPasswordValidator;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AdminChangePasswordController {
	@Autowired
	private UserAccountService userAccountService;
	@Autowired
	private MessageSource messageSource;
	@Autowired
	private RoleService roleService;

	@Autowired
	private ConfirmPasswordValidator confirmPasswordValidator;

	@Autowired
	private OldPasswordValidator oldPasswordValidator;

	@InitBinder
	protected void initBinder(WebDataBinder binder) {
		binder.addValidators(confirmPasswordValidator, oldPasswordValidator);
	}

	@RequestMapping(value = "adminChangePassword", method = RequestMethod.POST)
	public @ResponseBody
	Object doUpdate(
```

```
                    @RequestBody @Validated(ChangePassword.class) UserAccount
userAccount,
                    HttpServletRequest request) throws Exception{

             UserAccount userAccountDB = userAccountService.get(userAccount);

             BeanUtils.copyProperties(userAccountDB, userAccount, new String[] {
                          UserAccount.PASSWORD, UserAccount.CONFIRMED_PASSWORD,
                          UserAccount.OLD_PASSWORD });

             userAccountDB.setPassword(userAccount.getPassword());

             userAccountDB = userAccountService.update(userAccountDB);

             return new JSONSuccessMap(messageSource.getMessage(
                          "Success.changePassword.update", null, "",
null)).getMap();

       }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import edu.upm.dpsm.cims.useraccount.validation.groups.AdminUpdateAccount;
import edu.upm.dpsm.cims.useraccount.validation.validators.ConfirmPasswordValidator;
import
edu.upm.dpsm.cims.useraccount.validation.validators.CurrentAdminUpdateValidator;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AdminUpdateUserAccountController {
  @Autowired
  public UserAccountService           userAccountService;

  @Autowired
  private ConfirmPasswordValidator    confirmPasswordValidator;
```

```
  @Autowired
  private CurrentAdminUpdateValidator currentAdminUpdateValidator;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
    binder.addValidators(confirmPasswordValidator, currentAdminUpdateValidator);
  }

  @RequestMapping(value = "user/admin/update", method = RequestMethod.POST)
  public @ResponseBody
  Object doUpdate(@RequestBody @Validated(AdminUpdateAccount.class) UserAccount
userAccount, HttpServletRequest request)
      throws Exception {

    userAccountService.adminUpdate(userAccount);

    return
JSONMapUtils.createJSONSuccess(EMessages.USER_ACCOUNT_UPDATE_SUCCESS).getMap();

  }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import edu.upm.dpsm.cims.useraccount.validation.groups.AdminUpdateAccount;
import edu.upm.dpsm.cims.useraccount.validation.validators.ConfirmPasswordValidator;
import
edu.upm.dpsm.cims.useraccount.validation.validators.CurrentAdminUpdateValidator;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class AdminUpdateUserAccountController {
  @Autowired
  public UserAccountService           userAccountService;
```

```
  @Autowired
  private ConfirmPasswordValidator    confirmPasswordValidator;

  @Autowired
  private CurrentAdminUpdateValidator currentAdminUpdateValidator;

  @InitBinder
  protected void initBinder(WebDataBinder binder) {
    binder.addValidators(confirmPasswordValidator, currentAdminUpdateValidator);
  }

  @RequestMapping(value = "user/admin/update", method = RequestMethod.POST)
  public @ResponseBody
  Object doUpdate(@RequestBody @Validated(AdminUpdateAccount.class) UserAccount
userAccount, HttpServletRequest request)
      throws Exception {

    userAccountService.adminUpdate(userAccount);

    return
JSONMapUtils.createJSONSuccess(EMessages.USER_ACCOUNT_UPDATE_SUCCESS).getMap();

  }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.validation.ValidationException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class DeleteUserAccountController {

  @Autowired
  public UserAccountService userAccountService;

  @RequestMapping(value = "useraccount/delete", method = RequestMethod.POST)
```

```
  public @ResponseBody
  Object deleteUser(@RequestBody UserAccount userAccount, HttpServletRequest request)
throws Exception {
    HttpSession session = request.getSession(false);
    UserAccount currentUser = (UserAccount)
session.getAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION);
    if (currentUser.getId().equals(userAccount.getId())) {
      throw new ValidationException("You can't delete your own account");
    }
    userAccountService.delete(userAccount);

    return
JSONMapUtils.createJSONSuccess(EMessages.USER_ACCOUNT_DELETE_SUCCESS).getMap();
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.html.json.JSONSuccessMap;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * GetMyRoleController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetMyRoleController {
  @Autowired
  public UserAccountService userAccountService;

  @RequestMapping(value = "/user/role", method = { RequestMethod.POST,
RequestMethod.GET })
  public @ResponseBody
```

```
  Object searchUsersPage(HttpServletRequest request) throws Exception {
    HttpSession session = request.getSession(false);

    Long userAccountId = (Long)
session.getAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION);
    UserAccount userAccount = new UserAccount();
    userAccount.setId(userAccountId);
    UserAccount userAccountDetails = userAccountService.get(userAccount);

    return new JSONSuccessMap().addProp("role",
userAccountDetails.getRole().getName())
        .addProp("laboratoryUnit",
userAccountDetails.getLaboratoryUnit().getName()).getMap();

  }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import java.util.Collections;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.useraccount.models.Role;
import edu.upm.dpsm.cims.useraccount.services.RoleService;

/**
 * @author Miguelino San Miguel
 *
 */
@Controller
public class GetRoleListController {

      @Autowired
      private RoleService roleService;

      @RequestMapping(value = "/role/list", method = RequestMethod.GET)
      public @ResponseBody
      Object getRoleList(HttpServletRequest request, HttpServletResponse response) {
            try {
                  List<Role> userTypeList = roleService.search(new Role());
                  return userTypeList;
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return Collections.EMPTY_LIST;
```

```
      }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Feb 18, 2014
 */
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.useraccount.dto.UserAccountDTO;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * GetUserDetailController
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
@Controller
public class GetUserDetailController {
  @Autowired
  public UserAccountService userAccountService;

  @RequestMapping(value = "/user/details", method = RequestMethod.POST)
  public @ResponseBody
  Object searchUsersPage(@RequestBody UserAccount userAccount, Model model,
HttpServletRequest request)
      throws Exception {

    if (ObjectUtils.isEmpty(userAccount.getId())) {
      HttpSession session = request.getSession(false);
      userAccount.setId((Long)
session.getAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION));
    }
    UserAccount userAccountDetails = userAccountService.get(userAccount);
```

```
    return DTOUtil.copy(UserAccountDTO.class, userAccountDetails, new String[] {
UserAccount.CREATED_BY,
        UserAccount.UPDATED_BY });

  }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import java.util.Collections;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.util.DTOUtil;
import edu.upm.dpsm.cims.useraccount.dto.UserAccountDTO;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import edu.upm.dpsm.cims.useraccount.services.RoleService;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class SearchUserAccountController {

  @Autowired
  public UserAccountService userAccountService;

  @Autowired
  public MessageSource messageSource;

  @Autowired
  public RoleService roleService;

  @RequestMapping(value = "/user/list", method = { RequestMethod.POST,
      RequestMethod.GET })
  public@ResponseBody Object searchUsersPage(@RequestBody UserAccount userAccount,
      Model model, HttpServletRequest request) {
    try {
      List<UserAccount> userAccountList = userAccountService
          .search(userAccount);

      return DTOUtil.copy(UserAccountDTO.class, userAccountList);
```

```
    } catch (Exception e) {
      e.printStackTrace();
    }
    return Collections.emptyList();
  }
}
package edu.upm.dpsm.cims.useraccount.controllers;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.html.json.JSONMapUtils;
import edu.upm.dpsm.cims.core.message.EMessages;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
import edu.upm.dpsm.cims.useraccount.validation.groups.UpdateAccount;

/**
 * @author Miguelino San Miguel
 */
@Controller
public class UpdateUserAccountController {
  @Autowired
  public UserAccountService userAccountService;

  @RequestMapping(value = "myAccount", method = RequestMethod.POST)
  public @ResponseBody
  Object doUpdate(@RequestBody @Validated(UpdateAccount.class) UserAccount
userAccount, HttpServletRequest request)
      throws Exception {

    HttpSession session = request.getSession(false);
    Long userAccountId = (Long)
session.getAttribute(ProjectConstants.USER_ACCOUNT_ID_SESSION);
    userAccount.setId(userAccountId);
    userAccountService.update(userAccount);

    return
JSONMapUtils.createJSONSuccess(EMessages.USER_ACCOUNT_UPDATE_SUCCESS).getMap();
  }
}
package edu.upm.dpsm.cims.useraccount.dao.impl;

import org.springframework.stereotype.Repository;
```

```
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.useraccount.dao.RoleDAO;
import edu.upm.dpsm.cims.useraccount.models.Role;

/**
 * @author Miguelino San Miguel
 */
@Repository
@Transactional(rollbackFor=Exception.class)
public class RoleDAOImpl extends JPADao<Role> implements RoleDAO {

}
package edu.upm.dpsm.cims.useraccount.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.jpa.dao.JPADao;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.useraccount.dao.UserAccountDAO;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */
@Repository
@Transactional
public class UserAccountDAOImpl extends JPADao<UserAccount> implements UserAccountDAO
{

  @Override
  public UserAccount login(UserAccount userAccount) throws Exception {
    Session session = this.hibernateUtil.getSession();
    Criteria criteria = session.createCriteria(UserAccount.class);
    if (userAccount != null) {
      prepareSearchCriteria(userAccount, criteria);
    }
    List<UserAccount> result = criteria.list();

    if (!ObjectUtils.isEmpty(result)) {
      return result.get(0);
    }

    return null;
  }
}
package edu.upm.dpsm.cims.useraccount.dao;
```

```
import java.util.List;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.useraccount.models.Role;

/**
 * @author Miguelino San Miguel
 */
public interface RoleDAO extends DAO<Role> {

  @Override
  public List<Role> search(Role userType)
      throws Exception;
}
package edu.upm.dpsm.cims.useraccount.dao;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */

public interface  UserAccountDAO extends DAO<UserAccount> {

  public abstract UserAccount login(UserAccount userAccount)
      throws Exception;
}
package edu.upm.dpsm.cims.useraccount.dto;

import edu.upm.dpsm.cims.core.dto.EntityDTO;
import edu.upm.dpsm.cims.useraccount.models.Role;

public class UserAccountDTO extends EntityDTO{

       private String username;

       private String employeeId;

       private String firstname;

       private String lastname;

       private Long departmentId;

       private Long laboratoryUnitId;

       public Long getDepartmentId() {
    return departmentId;
  }

  public void setDepartmentId(Long departmentId) {
```

```
  this.departmentId = departmentId;
}

private Role role;

    private Long roleId;

    public String getUsername() {
          return username;
    }

    public void setUsername(String username) {
          this.username = username;
    }

    public String getEmployeeId() {
          return employeeId;
    }

    public void setEmployeeId(String employeeId) {
          this.employeeId = employeeId;
    }

    public String getFirstname() {
          return firstname;
    }

    public void setFirstname(String firstname) {
          this.firstname = firstname;
    }

    public String getLastname() {
          return lastname;
    }

    public void setLastname(String lastname) {
          this.lastname = lastname;
    }

    public Role getRole() {
          return role;
    }

    public void setRole(Role role) {
          this.role = role;
    }

    public Long getRoleId() {
          return roleId;
    }

    public void setRoleId(Long roleId) {
          this.roleId = roleId;
```

```
      }

  public Long getLaboratoryUnitId() {
    return laboratoryUnitId;
  }

  public void setLaboratoryUnitId(Long laboratoryUnitId) {
    this.laboratoryUnitId = laboratoryUnitId;
  }
}
/**
 * LIMS
 *
 * License :
 *
 * author: Miguelino San Miguel  Mar 9, 2014
 */
package edu.upm.dpsm.cims.useraccount.models;

/**
 * ERoles
 *
 * Description:
 *
 * author : Miguelino San Miguel
 */
public enum ERoles {
  SYSTEM_ADMIN(1L),
  LAB_ATTENDANT(4L),
  LAB_FACULTY_COORDINATOR(3L),
  SUPPLY_OFFICER(2L);

  private Long id;

  private ERoles(long roleId) {
    this.id = Long.valueOf(roleId);
  }

  public Long getId() {
    return this.id;
  }
}
package edu.upm.dpsm.cims.useraccount.models;

import java.io.Serializable;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import edu.upm.dpsm.cims.core.jpa.model.JPALookupModel;
```

```
/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "role")
@Access(AccessType.FIELD)
public class Role extends JPALookupModel implements Serializable {
  private static final long serialVersionUID = 9307741154845174 75L;
  public static final String NAME = "name";

  @Column(name="name",nullable = false, length = 50)
  private String name;

  public Role() {}

  public String getName() {
    return this.name;
  }

  public void setName(String title) {
    this.name = title;
  }

}
package edu.upm.dpsm.cims.useraccount.models;

import java.io.Serializable;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.validation.groups.Default;

import org.hibernate.validator.constraints.NotEmpty;

import edu.upm.dpsm.cims.core.ProjectConstants;
import edu.upm.dpsm.cims.core.jpa.model.JPAEntityModel;
import edu.upm.dpsm.cims.department.model.Department;
import edu.upm.dpsm.cims.laboratoryunit.model.LaboratoryUnit;
import edu.upm.dpsm.cims.useraccount.validation.groups.AdminUpdateAccount;
import edu.upm.dpsm.cims.useraccount.validation.groups.ChangePassword;
import edu.upm.dpsm.cims.useraccount.validation.groups.UpdateAccount;
```

```
/**
 * @author Miguelino San Miguel
 */
@Entity
@Table(name = "user_account")
@Access(AccessType.FIELD)
public class UserAccount extends JPAEntityModel implements Serializable {

  public static final String ROLE_ID            = "roleId";

  public static final String ROLE               = "role";

  public static final String EMPLOYEE_ID        = "employeeId";

  public static final String LASTNAME           = "lastname";

  public static final String FIRSTNAME          = "firstname";

  public static final String USERNAME           = "username";

  public static final String PASSWORD           = "password";

  public static final String CONFIRMED_PASSWORD = "confirmedPassword";

  public static final String OLD_PASSWORD       = "oldPassword";

  public static final String DEPARTMENT_ID      = "departmentId";

  private static final long  serialVersionUID   = 1L;

  @Pattern(regexp = ProjectConstants.PATTERN_ALPHA_NUMERIC_SEPARATOR)
  @Column(name = "username")
  private String             username;

  @NotEmpty(groups = { ChangePassword.class })
  @Transient
  private String             oldPassword;

  @NotEmpty(groups = { Default.class, ChangePassword.class })
  @Size(min = 6, max = 15, groups = { Default.class, ChangePassword.class })
  @Pattern(regexp = ProjectConstants.PATTERN_ALPHA_NUMERIC_SEPARATOR, groups = {
Default.class, ChangePassword.class })
  @Column(name = "password")
  private String             password;

  @NotEmpty(groups = { Default.class, ChangePassword.class })
  @Transient
  private String             confirmedPassword;

  @NotEmpty
  @Column(name = "employee_id")
  private String             employeeId;
```

```
  @Pattern(regexp = ProjectConstants.PATTERN_NAME, groups = { Default.class,
UpdateAccount.class,
      AdminUpdateAccount.class })
  @Column(name = "firstname")
  private String             firstname;

  @Pattern(regexp = ProjectConstants.PATTERN_NAME, groups = { Default.class,
UpdateAccount.class,
      AdminUpdateAccount.class })
  @Column(name = "lastname")
  private String             lastname;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "primary_role_id", updatable = false, insertable = false)
  private Role               role;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "department_id", updatable = false, insertable = false)
  private Department         department;

  @Column(name = "department_id")
  private Long               departmentId;

  @NotNull(groups = { Default.class, AdminUpdateAccount.class })
  @Min(value = 1L, groups = { Default.class, AdminUpdateAccount.class })
  @Column(name = "primary_role_id")
  private Long               roleId;

  @Column(name = "laboratory_unit_id")
  private Long laboratoryUnitId;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "laboratory_unit_id", updatable = false, insertable = false)
  private LaboratoryUnit laboratoryUnit;

  public LaboratoryUnit getLaboratoryUnit() {
    return laboratoryUnit;
  }

  public void setLaboratoryUnit(LaboratoryUnit laboratoryUnit) {
    this.laboratoryUnit = laboratoryUnit;
  }

  public Department getDepartment() {
    return department;
  }

  public void setDepartment(Department department) {
    this.department = department;
  }

  public Long getDepartmentId() {
```

```
  return departmentId;
}

public void setDepartmentId(Long departmentId) {
  this.departmentId = departmentId;
}

public String getUsername() {
  return username;
}

public void setUsername(String username) {
  this.username = username;
}

public String getPassword() {
  return password;
}

public void setPassword(String password) {
  this.password = password;
}

public String getEmployeeId() {
  return employeeId;
}

public void setEmployeeId(String employeeId) {
  this.employeeId = employeeId;
}

public String getFirstname() {
  return firstname;
}

public void setFirstname(String firstname) {
  this.firstname = firstname;
}

public String getLastname() {
  return lastname;
}

public void setLastname(String lastname) {
  this.lastname = lastname;
}

public Role getRole() {
  return role;
}

public void setRole(Role userType) {
  this.role = userType;
```

```
  }

  public String getOldPassword() {
    return oldPassword;
  }

  public void setOldPassword(String oldPassword) {
    this.oldPassword = oldPassword;
  }

  public String getConfirmedPassword() {
    return confirmedPassword;
  }

  public void setConfirmedPassword(String confirmedPassword) {
    this.confirmedPassword = confirmedPassword;
  }

  public Long getRoleId() {
    return roleId;
  }

  public void setRoleId(Long userTypeId) {
    this.roleId = userTypeId;
  }

  public Long getLaboratoryUnitId() {
    return laboratoryUnitId;
  }

  public void setLaboratoryUnitId(Long laboratoryUnitId) {
    this.laboratoryUnitId = laboratoryUnitId;
  }
}
package edu.upm.dpsm.cims.useraccount.search;

import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */
public class SearchLogin extends JPASearchModel<UserAccount> {

  public SearchLogin(UserAccount model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(UserAccount userAccount) throws Exception {
```

```
    setSearch(new SearchCondition().add(UserAccount.USERNAME, ERestrictionType.EQ,
userAccount.getUsername()).add(
        UserAccount.PASSWORD, ERestrictionType.EQ, userAccount.getPassword()));
  }

}
package edu.upm.dpsm.cims.useraccount.search;

import org.hibernate.criterion.MatchMode;

import edu.upm.dpsm.cims.core.jpa.search.JPASearchModel;
import edu.upm.dpsm.cims.core.jpa.search.criterion.ERestrictionType;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCondition;
import edu.upm.dpsm.cims.core.jpa.search.criterion.SearchCriterion;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */
public class SearchUserAccount extends JPASearchModel<UserAccount> {
  public SearchUserAccount(UserAccount model) throws Exception {
    super(model);
  }

  @Override
  protected void createSearchCriteria(UserAccount userAccount) throws Exception {

    setSearch(new SearchCondition()
        .add(SearchCondition
            .conjunction()
            .add(UserAccount.USERNAME, ERestrictionType.LIKE,
userAccount.getUsername(), MatchMode.START)
            .and(
                SearchCondition
                    .disjunction()
                    .add(UserAccount.EMPLOYEE_ID, ERestrictionType.EQ,
userAccount.getEmployeeId())
                    .or((new SearchCriterion(UserAccount.FIRSTNAME,
ERestrictionType.LIKE, userAccount.getFirstname()))
                        .and((new SearchCriterion(UserAccount.LASTNAME,
ERestrictionType.LIKE, userAccount
                            .getLastname())))).and(UserAccount.ROLE_ID,
ERestrictionType.EQ, userAccount.getRoleId()))));
  }
}
package edu.upm.dpsm.cims.useraccount.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
```

```
import edu.upm.dpsm.cims.useraccount.dao.RoleDAO;
import edu.upm.dpsm.cims.useraccount.models.Role;
import edu.upm.dpsm.cims.useraccount.services.RoleService;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional(rollbackFor = Exception.class)
public class RoleServiceImpl extends JPAService<Role> implements RoleService {

	@Autowired
	private RoleDAO	dao;

	@Override
	public DAO<Role> getDAO() {
		return this.dao;
	}
}
package edu.upm.dpsm.cims.useraccount.services.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.dao.DAO;
import edu.upm.dpsm.cims.core.jpa.search.pagination.Pagination;
import edu.upm.dpsm.cims.core.jpa.service.JPAService;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.useraccount.dao.UserAccountDAO;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.search.SearchLogin;
import edu.upm.dpsm.cims.useraccount.search.SearchUserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * @author Miguelino San Miguel
 */
@Service
@Transactional(rollbackFor = Exception.class)
public class UserAccountServiceImpl extends JPAService<UserAccount> implements
UserAccountService {

  @Autowired
  private UserAccountDAO dao;

  @Override
  public UserAccount save(UserAccount userAccount) throws Exception {
    String username = this.generateUsername(userAccount);

    List<UserAccount> existingAccounts = verifyConflictingUsername(username);
```

```
    if (!ObjectUtils.isEmpty(existingAccounts)) {
      username = resolveConflictingUsernames(username, existingAccounts);
    }
    userAccount.setUsername(username);
    return super.save(userAccount);
  }

  @Override
  public String generateUsername(UserAccount userAccount) throws Exception {
    String firstname = userAccount.getFirstname();
    String lastname = userAccount.getLastname();
    String username = firstname.substring(0, 1).toLowerCase() +
lastname.toLowerCase().replaceAll("\\s", "");
    return username;
  }

  private List<UserAccount> verifyConflictingUsername(String username) throws
Exception {
    UserAccount checkUsernameAccount = new UserAccount();
    checkUsernameAccount.setUsername(username);
    SearchUserAccount searchUserAccount = new
SearchUserAccount(checkUsernameAccount);
    checkUsernameAccount.setSearchModel(searchUserAccount);
    List<UserAccount> existingAccounts = this.search(checkUsernameAccount);
    return existingAccounts;
  }

  private String resolveConflictingUsernames(String username, List<UserAccount>
existingAccounts) {
    Integer size = existingAccounts.size();
    username = username + (size + 1);
    return username;
  }

  @Override
  public List<UserAccount> deleteAndSearch(UserAccount userAccount) throws Exception
{
    this.delete(userAccount);
    userAccount.setId(null);
    return this.search(userAccount);
  }

  @Override
  public List<UserAccount> search(UserAccount userAccount) throws Exception {

    SearchUserAccount sua = new SearchUserAccount(userAccount);
    userAccount.setSearchModel(sua);
    Pagination pagination = userAccount.getPagination();
    pagination = (pagination == null) ? (new Pagination()) : pagination;
    pagination.setEnable(true);
    userAccount.setPagination(pagination);
    return this.dao.search(userAccount);
```

```
}

@Override
public UserAccount login(UserAccount userAccount) throws Exception {
  SearchLogin sua = new SearchLogin(userAccount);
  userAccount.setSearchModel(sua);

  Pagination pagination = userAccount.getPagination();
  pagination = (pagination == null) ? (new Pagination()) : pagination;
  pagination.setEnable(true);
  userAccount.setPagination(pagination);
  return dao.login(userAccount);
}

@Override
public DAO<UserAccount> getDAO() {
  return this.dao;
}

@Override
public UserAccount update(UserAccount userAccount) throws Exception {
  UserAccount userAccountDB = get(userAccount);
  userAccountDB.setFirstname(userAccount.getFirstname());
  userAccountDB.setLastname(userAccount.getLastname());
  return super.update(userAccountDB);
}

@Override
public UserAccount adminUpdate(UserAccount userAccount) throws Exception {
  UserAccount userAccountDB = get(userAccount);
  userAccountDB.setFirstname(userAccount.getFirstname());
  userAccountDB.setLastname(userAccount.getLastname());
  userAccountDB.setEmployeeId(userAccount.getEmployeeId());
  userAccountDB.setRoleId(userAccount.getRoleId());
  userAccountDB.setDepartmentId(userAccount.getDepartmentId());
  userAccountDB.setLaboratoryUnitId(userAccount.getLaboratoryUnitId());
  userAccountDB = super.update(userAccountDB);
  return userAccountDB;
}

@Override
public boolean isAccountExist(UserAccount userAccount) throws Exception {
  SearchLogin sua = new SearchLogin(userAccount);
  userAccount.setSearchModel(sua);
  return super.getTotalSize(userAccount) > Long.valueOf(0L);
}

@Override
public UserAccount get(Long id) throws Exception {
  UserAccount userAccount = new UserAccount();
  userAccount.setId(id);
  return super.get(userAccount);
}
```

```
}
package edu.upm.dpsm.cims.useraccount.services;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.useraccount.models.Role;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface RoleService extends BusinessService<Role> {}
package edu.upm.dpsm.cims.useraccount.services;

import java.util.List;

import org.springframework.transaction.annotation.Transactional;

import edu.upm.dpsm.cims.core.app.service.BusinessService;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */
@Transactional
public interface UserAccountService extends BusinessService<UserAccount> {

  public UserAccount get(Long id) throws Exception;
  public List<UserAccount> deleteAndSearch(UserAccount userAccount) throws Exception;

  public UserAccount login(UserAccount userAccount) throws Exception;

  public boolean isAccountExist(UserAccount userAccount) throws Exception;

  public UserAccount adminUpdate(UserAccount userAccount) throws Exception;

  public String generateUsername(UserAccount userAccount) throws Exception;
}
package edu.upm.dpsm.cims.useraccount.validation.groups;

public interface AdminUpdateAccount {}
package edu.upm.dpsm.cims.useraccount.validation.groups;

public interface ChangePassword {}
package edu.upm.dpsm.cims.useraccount.validation.groups;

public interface UpdateAccount {}
package edu.upm.dpsm.cims.useraccount.validation.validators;

import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * @author Miguelino San Miguel
 */
@Component
public class AvailableUsernameValidator implements Validator {

	@Autowired
	private MessageSource messageSource;

	@Autowired
	private UserAccountService userAccountService;

	@Override
	public boolean supports(Class<?> user) {
		return UserAccount.class.equals(user);
	}

	@Override
	public void validate(Object target, Errors errors) {
		try {
			UserAccount userAccount = (UserAccount) target;
			String username = userAccount.getUsername();
			if (!ObjectUtils.isEmpty(username)) {
				BindingResult result = (BindingResult) errors;
				UserAccount uaCount = new UserAccount();
				uaCount.setUsername(username);
				if (userAccountService.isAccountExist(uaCount)) {
					ValidationUtils.addBindingError(result,
								UserAccount.USERNAME,
EValidationMessages.UNAVAILABLE_USERNAME);
				}
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
package edu.upm.dpsm.cims.useraccount.validation.validators;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
```

```
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */
@Component
public class ConfirmPasswordValidator implements Validator {

	@Autowired
	private MessageSource messageSource;

	@Override
	public boolean supports(Class<?> user) {
		return UserAccount.class.equals(user);
	}

	@Override
	public void validate(Object target, Errors errors) {
		UserAccount userAccount = (UserAccount) target;
		BindingResult result = (BindingResult) errors;
		String confirmedPassword = userAccount.getConfirmedPassword();
		String password = userAccount.getPassword();
		if (!ObjectUtils.isEmpty(password)
				&& !ObjectUtils.isEmpty(confirmedPassword)) {
			if (!password.equals(confirmedPassword)) {
				ValidationUtils.addBindingError(result,
						UserAccount.CONFIRMED_PASSWORD,

	EValidationMessages.INVALID_CONFIRMED_PASSWORD);
			}
		}

	}

}
package edu.upm.dpsm.cims.useraccount.validation.validators;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
```

```
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.UserAccountUtil;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;

/**
 * @author Miguelino San Miguel
 */
@Component
public class CurrentAdminUpdateValidator implements Validator {

  @Autowired
  private MessageSource messageSource;

  @Override
  public boolean supports(Class<?> clazz) {
    return UserAccount.class.equals(clazz);
  }

  @Override
  public void validate(Object target, Errors errors) {
    try {
      UserAccount userAccount = (UserAccount) target;
      BindingResult result = (BindingResult) errors;
      Long userAccountId = UserAccountUtil.getUserAccountId();
      UserAccount userAccountSession = new UserAccount();
      userAccountSession.setId(userAccountId);
      if (userAccountSession.getId().equals(userAccount.getId()) &&
userAccountSession.getRoleId().equals(4L)
          && !userAccount.getRoleId().equals(4L)) {
        userAccount.setRoleId(4L);
        ValidationUtils.addBindingError(result, UserAccount.ROLE_ID,
EValidationMessages.INVALID_SELF_UPDATE_ROLE);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
package edu.upm.dpsm.cims.useraccount.validation.validators;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.core.util.ObjectUtils;
import edu.upm.dpsm.cims.settings.forms.LoginForm;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;
```

```
@Component
public class ExistingLoginValidator implements Validator {

	@Autowired
	private UserAccountService userAccountService;

	@Override
	public boolean supports(Class<?> paramClass) {
		return LoginForm.class.equals(paramClass);
	}

	@Override
	public void validate(Object target, Errors errors) {
		try {
			LoginForm login = (LoginForm) target;
			String username = login.getUsername();
			String password = login.getPassword();
			if (!ObjectUtils.isEmpty(username) &&
!ObjectUtils.isEmpty(password)) {
				BindingResult result = (BindingResult) errors;
				UserAccount uaCount = new UserAccount();
				uaCount.setUsername(username);
				if (!userAccountService.isAccountExist(uaCount)) {
					ValidationUtils.addBindingError(result,
							ValidationUtils.GENERAL_FORM_ERRORS,

	EValidationMessages.NON_EXISTING_ACCOUNT);
				}
			}
		} catch (Exception e) {
			e.printStackTrace();
		}

	}

}
package edu.upm.dpsm.cims.useraccount.validation.validators;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Component;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import edu.upm.dpsm.cims.core.app.validation.constant.EValidationMessages;
import edu.upm.dpsm.cims.core.app.validation.util.ValidationUtils;
import edu.upm.dpsm.cims.useraccount.models.UserAccount;
import edu.upm.dpsm.cims.useraccount.services.UserAccountService;

/**
 * @author Miguelino San Miguel
```

```
 */
@Component
public class OldPasswordValidator implements Validator {
      @Autowired
      private MessageSource messageSource;

      @Autowired
      private UserAccountService userAccountService;
  @Override
  public boolean supports(Class<?> clazz) {

    return UserAccount.class.equals(clazz);
  }

  @Override
  public void validate(Object target, Errors errors) {
    try {
      UserAccount userAccount = (UserAccount) target;
      BindingResult result = (BindingResult) errors;
      UserAccount userAccountDB = userAccountService.get(userAccount);

      if (!userAccountDB.getPassword()
          .equals(userAccount.getOldPassword())) {
        ValidationUtils.addBindingError(result, UserAccount.OLD_PASSWORD,
            EValidationMessages.INVALID_OLD_PASSWORD);
      }
    } catch (Exception e) {
      e.printStackTrace();

    }

  }
}
# Direct log messages to stdout
log4j.rootLogger=DEBUG, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.conversionPattern=%5p (%F:%L) - %m%n

# Log everything. Good for troubleshooting
log4j.logger.org.hibernate=DEBUG

log4j.logger.org.springframework.transaction=DEBUG

login.authorization.required = Login credentials required

Success.userAccount.update = Successfully updated the account
Fail.userAccount.update = Failed to update the account

Success.userAccount.delete Successfully deleted the account
Fail.userAccount.delete = Failed to delete the account

Success.chemicalDefinition.add = Successfully created a chemical definition
```

```
Fail.chemicalDefinition.add = Failed to create a chemical definition

Success.chemicalDefinition.update = Successfully updated a chemical definition
Fail.chemicalDefinition.update = Failed to update a chemical definition

Success.chemicalDefinition.delete = Successfully deleted a chemical definition!
Failed.chemicalDefinition.delete = Deletion of chemical definition failed.

Success.chemicalItem.add = New chemical item is successfully added!
Failed.chemicalItem.add = Failed to create new chemical item.

Success.chemicalItem.update = Chemical item is successfully updated!
Failed.chemicalItem.update = Failed to update chemical item.

Success.chemicalItem.delete = Successfully deleted a chemical item!
Failed.chemicalItem.delete = Deletion of chemical item failed.

Success.labwareDefinition.add = Successfully created a labware definition
Fail.labwareDefinition.add = Failed to create a labware definition

Success.labwareDefinition.update = Successfully updated a labware definition
Fail.labwareDefinition.update = Failed to update a labware definition

Success.labwareDefinition.delete = Successfully deleted a chemical definition!
Failed.labwareDefinition.delete = Deletion of chemical definition failed.

Success.labwareItem.add = New chemical item is successfully added!
Failed.labwareItem.add = Failed to create new chemical item.

Success.replenishmentRequest.create = Replenishment request has been submitted!

Success.repairRequest.create = Repair request has been submitted!

Success.condemnationRequest.create = Condemnation request has been submitted!

Success.maintenanceRequest.approved = Successfully approved the request!

Success.maintenanceRequest.rejected = Successfully rejected the request!

Success.accountability.add = Successfully create an accountability record!

Success.accountability.clear = Successfully cleared an accountability record!

Success.accountability.unclear = Successfully uncleared an accountability record!
##General Validation Messages
NotEmpty = This field is required
NotNull = This field is required
Size = This field length must be between {2} and {1} characters
Pattern = The value has invalid format
Min = The minimum value for this field is {1}
Max = The maximum value for this field is {1}

#Chemical
```

```
NotExisting.chemicalName = Chemical definition is not existing
NotEmpty.name = Name is required
Size.name = Name must have {2} - {1} characters
NotEmpty.molFormula = Molecular Formula is required
Size.molFormula = Molecular Formula must have {2} - {1} characters
Unavailable.chemical.quantity = The chemical being requested is unavailable.
Lacking.chemical.quantity = The chemical being requested is beyond the available
quantity.

##CHEMISTRY
#CHEMICAL ENTITY

#validation errors
NotEmpty.chemistryChemicalDefinition.name = Chemical name is required
Size.chemistryChemicalDefinition.name = Chemical name must have {2} - {1} characters
NotEmpty.chemistryChemicalDefinition.molFormula = Molecular Formula is required
Size.chemistryChemicalDefinition.molFormula = Molecular Formula must have {2} - {1}
characters
NotExisting.chemistryChemicalDefinition.chemicalName = Chemical definition is not
existing


#CHEMICAL ITEM ENTITY
#validation errors



##PHYSICS
#CHEMICAL ENTITY
#validation errors
NotEmpty.physicsChemicalDefinition.name = Chemical name is required
Size.physicsChemicalDefinition.name = Chemical name must have {2} - {1} characters
Size.physicsChemicalDefinition.molFormula = Molecular Formula must have {2} - {1}
characters


##LABWARE
labwareDefinition.add.success = New labware definition is successfully added!
labwareDefinition.add.fail = Failed to create new labware definition.

labwareDefinition.update.success = Labware definition is successfully updated!
labwareDefinition.update.fail = Failed to update labware definition.

#UserAccount
NotAvailable.userAccount.username = Username is not available
NotEmpty.userAccount.username = Username is required
Size.userAccount.username = Username must have {2} - {1} characters
Pattern.userAccount.username = Username must start with a letter
(letter,number,underscore)
NotEmpty.userAccount.password = Password is required
Size.userAccount.password = Password must have {2} - {1} characters
NotEmpty.userAccount.confirmedPassword = Confirm password is required
Invalid.userAccount.confirmedPassword = Confirm password doesn't match
```

```
Pattern.userAccount.password = Password must start with a letter
(letter,number,underscore)
Pattern.userAccount.firstname = Firstname must contain only letters
Pattern.userAccount.lastname = Lastname must contain only letters
Min.userAccount.roleId = Please provide the user role
NotNull.userAccount.roleId = Please provide the user role

#ChangePassword Validation
Invalid.userAccount.oldPassword = Incorrect old password
NotEmpty.userAccount.oldPassword = Old password is required

Success.changePassword.update = Successfully updated your password!
Failed.changePassword.update = Updating of your password failed.

#Admin Access
Access.admin.required =  Access required admin credentials!
Success.userAccount.create = Successfully created a new account!
Failed.userAccount.create = Creation of new account failed.

Success.userAccount.delete = Successfully deleted an account!
Failed.userAccount.delete = Deletion of account failed.

Success.userAccount.update = Successfully updated the account!
Failed.userAccount.update = Updating of an account failed.

Invalid.userAccount.admin.update.self = You can't change your admin credentials!

#Login validation
NotEmpty.loginForm.username = Username is required
Size.loginForm.username = Username must have {2} - {1} characters
Pattern.loginForm.username = Username must start with a letter
(letter,number,underscore)
NotEmpty.loginForm.password = Password is required
Size.loginForm.password = Password must have {2} - {1} characters
Pattern.loginForm.password = Password must start with a letter
(letter,number,underscore)
NotFound.loginForm.account = Account is non-existing

<div ng-controller="NewAccountabilityCtrl" class="row col-lg-10">
  <br> <br>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add Accountability</h4>
    </div>
    <div class="panel-body">
      <form id="newAccountabilityForm" class="form-horizontal" role="form" ui-
validate ui-model="newAccountability"
        ng-submit="create()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="studentNumber">Student Number<span
class="required">*</span></label>
```

```
          </div>
          <div class="col-lg-6">
            <input type="text" name="studentNumber" id="studentNumber" class="form-
control"
              ng-model="newAccountability.studentNumber"
            />
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="studentName">Student Name<span
class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" name="studentName" id="studentName" class="form-
control"
              ng-model="newAccountability.studentName"
            />
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="subject">Subject</label>
          </div>
          <div class="col-lg-6">
            <input type="text" name="subject" id="subject" class="form-control" ng-
model="newAccountability.subject" />
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="section">Section</label>
          </div>
          <div class="col-lg-6">
            <input type="text" name="section" id="section" class="form-control" ng-
model="newAccountability.section" />
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="professor">Professor</label>
          </div>
          <div class="col-lg-6">
            <input type="text" name="professor" id="professor" class="form-control"
              ng-model="newAccountability.professor"
            />
          </div>
        </div>
```

```
        <div class="form-group">
          <div class="col-lg-2">
            <label for="semesterId">Semester<span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <select class="form-control" ng-model="newAccountability.semesterId"
id="semesterId" name="semesterId"
              ng-options="semester.id as semester.name for semester in semesters"
            >
            </select>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="academicYearId">Academic Year<span
class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <select class="form-control" ng-model="newAccountability.academicYearId"
id="academicYearId"
              name="academicYearId"
            >
              <option ng-repeat="academicYear in academicYears"
value="{{academicYear.id}}">{{academicYear.startYear}}
                - {{academicYear.endYear}}</option>
            </select>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="isCleared">Cleared</label>
          </div>
          <div class="col-lg-6">
            <span class="btn-group"> <label class="btn btn-primary"
              ng-class="{active:(definition.isCleared == true)}"
            > <input type="radio" ng-value="true" ng-
model="newAccountability.isCleared" name="isCleared"
                class="ng-pristine ng-valid"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.isCleared == false)}"> <input
                type="radio" ng-model="newAccountability.isCleared" ng-value="false"
name="isCleared"
                class="ng-pristine ng-valid"
              >No
            </label>
            </span>
          </div>
        </div>

        <div class="form-group">
```

```
          <div class="col-lg-2">
            <label for="accountability">Add Laboratory Item</label>
          </div>
          <div class="col-lg-6">
            <select class="form-control" ng-model="autocomplete.moduleId"
id="moduleId" name="moduleId"
              ng-options="module.id as module.name for module in modules"
placeholder="Select Laboratory Module"
            >
            </select>
          </div>
          <div class="col-lg-6">

            <div class="input-group">
              <input type="text" id="newAccountability.accountabilityLineItems"
                name="newAccountability.accountabilityLineItems" class="form-control"
ng-model="autocomplete.lineItem"
                placeholder="Autocomplete Laboratory Item"
                typeahead="lineItem as lineItem.name for lineItem in
autocompleteLineItem($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-select="autocomplete.lineItem =
$model" on-enter="addLineItem()"
                autocomplete="off"
              /> <span class="input-group-addon btn btn-default" ng-
click="addLineItem()"> Add</span>
            </div>
          </div>
        </div>

        <div class="
                  table-responsive
                l-lg-10">
          <table ng-table="lineItemTable" class="table  table-bordered">
            <tbody>
              <tr ng-repeat="item in $data">
                <td data-title="'Module'">{{item.moduleName}}</td>
                <td data-title="'Item Name'"><a
class="clickable">{{item.itemName}}</a></td>
                <td data-title="'Quantity'"><a class="clickable"><input type="text"
                    class="form-control input-sm" ng-model="item.quantity"
                  /></a></td>
                <td data-title="'Available Quantity'">{{item.availableQuantity}}</td>
                <td data-title="'Consumable'">{{item.isConsumable}}</td>
                <td data-title="'Action'"><a class="clickable" ng-
click="removeLineItem(item)">Remove</a></td>
              </tr>
              <tr ng-if="!$data || !$data.length">
                <td colspan="6"><div class="text-center">
                    <strong>No item added yet</strong>
                  </div></td>
              </tr>
            </tbody>
            <tfoot>
```

```
              <tr>
                <td>Page: {{lineItemTable.page()}}</td>
              </tr>
            </tfoot>
          </table>
        </div>

        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Save</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
(function($) {
       angular.module("cims").controller("NewAccountabilityCtrl", function($scope,
$rootScope, $http, ngTableParams) {
              $.extend($scope, {
                     semesters : [],
                     academicYears : [],
                     autocomplete : {},
                     init : function() {
                           $scope.newAccountability = {
                                  isCleared : false,
                                  accountabilityLineItems : []
                           };
                           $scope.lineItemTable.reload();
                           if ($scope.modules) {
                                  $scope.newAccountability.moduleId =
$scope.modules[0].id;
                           }

                     },
                     create : function() {
                           $http.post("accountability/new",
$scope.newAccountability).success(function(data) {
                                  $scope.init();
                                  alert(data.message);
                                  $rootScope.$broadcast("created-accountability",{});
                           });
                     },

                     getSemesters : function() {
                           $http.post("semester/list", {}).success(function(data) {
                                  $scope.semesters = data;
                           });
                     },
                     getAcademicYears : function() {
```

```
                        $http.post("academic-year/list",
{}).success(function(data) {
                                $scope.academicYears = data;
                        });
                },
                addLineItem : function() {
                        var lineItem = $scope.autocomplete.lineItem;
                        if(!lineItem || !lineItem.name){
                                return;
                        }
                        var isChemical = [ 3, 7, 13
                        ].indexOf($scope.autocomplete.moduleId) > -1;
                        $scope.newAccountability.accountabilityLineItems.push({

                                definitionId : lineItem.id,

                                itemName : lineItem.name,
                                moduleId : $scope.autocomplete.moduleId,
                                moduleName :
$scope.asName($scope.autocomplete.moduleId),
                                quantity : 1,
                                availableQuantity : lineItem.totalQuantity,
                                isConsumable : isChemical
                        });
                        $scope.lineItemTable.reload();
                        $scope.autocomplete.lineItem = null;
                },
                removeLineItem : function(item) {
                        var array =
$scope.newAccountability.accountabilityLineItems,
                          index = array.indexOf(item);
                        if (index > -1) {
                                array.splice(index, 1);
                                $scope.lineItemTable.reload();
                        }
                },

                autocompleteLineItem : function(key) {
                        return $http.post("accountability/lineitem/autocomplete",
{
                                moduleId : $scope.autocomplete.moduleId,
                                laboratoryItemName : key
                        }).then(function(response) {
                                return response.data;
                        });
                }
        });

        $scope.getSemesters();
        $scope.getAcademicYears();

        $scope.lineItemTable = new ngTableParams({
                page : 1,
```

```
                        count : 10
                }, {
                        total : 0,
                        getData : function($defer, params) {
                                var data =
$scope.newAccountability.accountabilityLineItems;
                                params.total(data.length);
                                $defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
                        }
                });

                $scope.$watch("autocomplete.moduleId", function() {
                        $scope.autocomplete.lineItem = null;
                });

                $scope.init();
        });
})(jQuery);
<div ng-controller="ReportAccountabilityCtrl">
  <br> <br>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Generate Report</h4>
    </div>
    <div class="panel-body">
      <form id="reportForm" class="form-horizontal" role="form" ui-validate ui-
model="accountabilityReport"
        ng-submit="generate()">

        <div class="form-group">
          <div class="col-lg-2">
            <label for="semesterId">Semester<span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <select class="form-control" ng-model="accountabilityReport.semesterId"
id="semesterId" name="semesterId"
              ng-options="semester.id as semester.name for semester in semesters"
            >
            </select>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="academicYearId">Academic Year<span
class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <select class="form-control" ng-
model="accountabilityReport.academicYearId" id="academicYearId"
              name="academicYearId"
            >
```

```
              <option ng-repeat="academicYear in academicYears"
value="{{academicYear.id}}">{{academicYear.startYear}}
                - {{academicYear.endYear}}</option>
            </select>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-8">
            <button type="submit" class="btn btn-primary">Generate</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("ReportAccountabilityCtrl", function($scope,
$rootScope, $http, ngTableParams) {
             $.extend($scope, {

                   init: function(){
                         $scope.accountabilityReport = {
                         };
                   },
                   getSemesters : function() {
                         $http.post("semester/list", {}).success(function(data) {
                               $scope.semesters = data;
                         });
                   },
                   getAcademicYears : function() {
                         $http.post("academic-year/list",
{}).success(function(data) {
                               $scope.academicYears = data;
                         });
                   },
                   generate : function() {
                         $http.post("accountability/report/generate",
$scope.accountabilityReport).success(function(data) {
                               // TODO
                               var confirmDownload = confirm("Download the pdf
report?");
                               if(confirmDownload){
                                     window.open(data.url, '_blank');
                               }
                         });
                   }
             });
             $scope.init();
             $scope.getSemesters();
             $scope.getAcademicYears();
```

```
      });
})(jQuery);
<div ng-controller="SearchAccountabilityCtrl">
  <br> <br>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Search Accountability</h4>
    </div>
    <div class="panel-body">
      <form id="searchForm" class="form-horizontal" role="form" ng-
submit="accountabilityTable.reload()"
        ui-model="searchAccountabilityForm" ui-validate
      >
        <div class="form-group">
          <div class="col-lg-1">
            <label for="studentNumber">Student Number</label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control" id="studentNumber"
name="studentNumber"
              ng-model="searchAccountabilityForm.studentNumber" placeholder="Name"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-1">
            <label for="isCleared">Cleared</label>
          </div>
          <div class="col-lg-6">
            <span class="btn-group"> <label class="btn btn-primary"
              ng-class="{active:(definition.isCleared == true)}"
            > <input type="radio" ng-value="true" ng-
model="searchAccountabilityForm.isCleared" checked="checked"
                name="isCleared" class="ng-pristine ng-valid"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.isCleared == false)}"> <input
                type="radio" ng-model="searchAccountabilityForm.isCleared" ng-
value="false" name="isCleared"
                class="ng-pristine ng-valid"
              >No
            </label> <label class="btn btn-primary"
              ng-class="{active:(definition.isCleared == null || definition.isCleared
== '')}"
            > <input type="radio" ng-model="searchAccountabilityForm.isCleared" ng-
value="" name="isCleared"
                class="ng-pristine ng-valid"
              >Both
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
```

```
          <div class="col-lg-7">
            <button type="submit" class="btn btn-primary">Search</button>
          </div>
        </div>
      </form>
    </div>
  </div>
  <div class="table-responsive col-lg-12">
    <table ng-table="accountabilityTable" class="table  table-bordered">
      <tbody ng-repeat="accountability in $data">
        <tr>
          <td data-title="'Student Name'"
sortable="'studentName'">{{accountability.studentName}}</td>
          <td data-title="'Student Number'">{{accountability.studentNumber}}</td>
          <td data-title="'Date'"
sortable="'createdDate'">{{accountability.createdDate | date:'medium'}}</td>
          <td data-title="'Cleared'"
sortable="'isCleared'">{{accountability.isCleared}}</td>
          <td data-title="'Action'"><span ng-if="!accountability.isCleared"><a
class="clickable"
              ng-click="clear(accountability)"
            >Clear</a> </span>
            <span ng-if="accountability.isCleared">
            <a class="clickable"
              ng-click="unclear(accountability)"
            >Unclear</a>
            </span>
            |
            <span><a class="clickable" ng-init="toggleLineItem = false"
              ng-click="toggleLineItem = !toggleLineItem"
            >
            View Line Items <i ng-if="!toggleLineItem" class="glyphicon glyphicon-
chevron-down"></i><i
                ng-if="toggleLineItem" class="glyphicon glyphicon-chevron-up"
              ></i></a></span></td>
        </tr>
        <tr ng-if="toggleLineItem" ng-show="toggleLineItem">
          <td colspan="5">
            <table class="table  table-bordered">
              <tbody>
               <tr>
                  <td>Item Name</td>
                  <td>Quantity</td>
                  <td>Consumable</td>
                </tr>
                <tr ng-repeat="lineItem in accountability.accountabilityLineItems">
                  <td>{{lineItem.itemName}}</td>
                  <td>{{lineItem.quantity}}</td>
                  <td>{{lineItem.isConsumable}}</td>
                </tr>
              </tbody>
            </table>
          </td>
```

```
        </tr>
      </tbody>
      <tbody ng-if="!$data || !$data.length">
        <tr>
          <td colspan="5"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Page: {{accountabilityTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>
</div>
(function($) {
      angular.module("cims").controller("SearchAccountabilityCtrl", function($scope,
$rootScope, $http, ngTableParams,$filter) {
            $.extend($scope, {
                  searchAccountabilityForm : {
                  },
                  clear : function(accountability){
                        $http.post("accountability/clear", {id:accountability.id})
                          .success(function(data) {
                              alert(data.message);
                              $scope.accountabilityTable.reload();
                          });
                  },
                  unclear : function(accountability){
                        $http.post("accountability/unclear",
{id:accountability.id})
                          .success(function(data) {
                              alert(data.message);
                              $scope.accountabilityTable.reload();
                          });
                  }
            });

            $scope.accountabilityTable = new ngTableParams({
                  page : 1,
                  count : 10,
                  sorting : {
                        isCleared : 'asc'
                  }
            }, {
                  total : 0,
                  getData : function($defer, params) {
                        $http.post("accountability/search",
$scope.searchAccountabilityForm).success(function(data) {
                              var orderedData = params.sorting() ?
             $filter('orderBy')(data, params.orderBy()) :
```

```
                data;
                                params.total(data.length);
                                $defer.resolve(orderedData.slice((params.page() - 1)
* params.count(), params.page() * params.count()));
                            });
                    }
                });
                $scope.$on("created-accountability",function(event,data){
                        $scope.accountabilityTable.reload();
                });

        });
})(jQuery);
<div ng-controller="TodayAccountabilityCtrl">

  <div class="table-responsive col-lg-12">
    <table ng-table="todayAccountabilityTable" class="table  table-bordered">
      <tbody ng-repeat="accountability in $data">
        <tr>
          <td data-title="'Student Name'"
sortable="'studentName'">{{accountability.studentName}}</td>
          <td data-title="'Student Number'">{{accountability.studentNumber}}</td>
          <td data-title="'Date'"
sortable="'createdDate'">{{accountability.createdDate | date:'medium'}}</td>
          <td data-title="'Cleared'"
sortable="'isCleared'">{{accountability.isCleared}}</td>
          <td data-title="'Action'"><span ng-if="!accountability.isCleared"><a
class="clickable"
              ng-click="clear(accountability)"
            >Clear</a> |</span> <span><a class="clickable" ng-init="toggleLineItem =
false"
              ng-click="toggleLineItem = !toggleLineItem"
            > View Line Items <i ng-if="!toggleLineItem" class="glyphicon glyphicon-
chevron-down"></i><i
                ng-if="toggleLineItem" class="glyphicon glyphicon-chevron-up"
              ></i></a></span></td>
        </tr>
        <tr ng-show="toggleLineItem">
          <td colspan="5">
            <table class="table  table-bordered">
              <tbody>
                <tr>
                  <td>Item Name</td>
                  <td>Quantity</td>
                  <td>Consumable</td>
                </tr>
                <tr ng-repeat="lineItem in accountability.accountabilityLineItems">
                  <td>{{lineItem.itemName}}</td>
                  <td>{{lineItem.quantity}}</td>
                  <td>{{lineItem.isConsumable}}</td>
                </tr>
              </tbody>
            </table>
```

```
          </td>
        </tr>
      </tbody>
      <tbody ng-if="!$data || !$data.length">
        <tr>
          <td colspan="5"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Page: {{todayAccountabilityTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>
</div>
(function($) {
       angular.module("cims").controller("TodayAccountabilityCtrl", function($scope,
$rootScope, $http, ngTableParams,$filter) {
             $.extend($scope, {
                    clear : function(accountability){
                           $http.post("accountability/clear", {id:accountability.id})
                             .success(function(data) {
                                  alert(data.message);
                                  $scope.todayAccountabilityTable.reload();
                             });
                    }
             });

             $scope.todayAccountabilityTable = new ngTableParams({
                    page : 1,
                    count : 10,
                    sorting : {
                           isCleared : 'asc'
                    }
             }, {
                    total : 0,
                    getData : function($defer, params) {
                           $http.post("accountability/today/search",
$scope.searchAccountabilityForm).success(function(data) {
                                  var orderedData = params.sorting() ?
              $filter('orderBy')(data, params.orderBy()) :
              data;
                                  params.total(data.length);
                                  $defer.resolve(orderedData.slice((params.page() - 1)
* params.count(), params.page() * params.count()));
                           });
                    }
             });

             $scope.$on("created-accountability",function(event,data){
```

```
                $scope.todayAccountabilityTable.reload();
            });

    });
})(jQuery);
<div ng-controller="biologyAddChemicalDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add Biology Definition</h4>
    </div>
    <div class="panel-body">
      <form id="searchbiologyForm" class="form-horizontal" role="form" ui-validate
ui-model="definition"
        ng-submit="add()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Name">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="molFormula">Molecular Formula</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="molFormula" ng-
model="definition.molFormula"
              placeholder="Molecular Formula"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="color">Color</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="color" ng-
model="definition.color" placeholder="Color">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="texture">Texture</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="texture" ng-
model="definition.texture" placeholder="Texture">
          </div>
        </div>
```

```
        <div class="form-group">
          <div class="col-lg-2">
            <label for="grade">Grade</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="grade" ng-
model="definition.grade" placeholder="Grade">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="hygroscopic">Hygroscopic</label>
          </div>
          <div class="col-lg-8">
            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.hygroscopic == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.hygroscopic"
checked="checked"
                name="hygroscopic" class="ng-pristine ng-valid"
            > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.hygroscopic == false)}"> <input
                type="radio" ng-model="definition.hygroscopic" ng-value="false"
name="hygroscopic"
                class="ng-pristine ng-valid"
            > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="explosive">Explosive</label>
          </div>
          <div class="col-lg-8">

            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.explosive == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.explosive"
checked="checked" name="explosive"
                class="ng-pristine ng-valid"
            > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.explosive == false)}"> <input
                type="radio" ng-model="definition.explosive" ng-value="false"
name="explosive"
                class="ng-pristine ng-valid"
            > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
```

```
          <div class="col-lg-2">
            <label for="thresh">Thresh</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="thresh" ng-
model="definition.thresh" placeholder="Thresh">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("biologyAddChemicalDefCtrl",
function($scope, $rootScope, $http) {
            $.extend($scope, {
                  init : function() {
                        $scope.definition = {
                              minimum : 0,
                              hygroscopic : true,
                              explosive : true
                        };
                  },
                  add : function() {
                        $http.post("biology/chemical/add",
$scope.definition).success(function(data) {
                              alert(data.message);
                              if (data.success) {
                                    $scope.init();
                              }
                        });
                  }
            });

            $scope.init();
```

```
      });
})(jQuery);
<div ng-controller="biologyAddItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title" ng-click="alert('');">Add Biology Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchbiologyForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="chemicalName"> Chemical Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="chemicalId" ng-
model="item.chemicalId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="chemicalName" name="chemicalName"
                ng-model="item.chemicalName" placeholder="Autocomplete Chemical Name"
                typeahead="chemical as chemical.name for chemical in
autocomplete($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-
select="item.chemicalId=$model.id;item.chemicalName=$model.name"
              >
            </div>
          </div>

          <div class="form-group">
            <div class="col-lg-3">
              <label for="expDate"> Expiration Date</label>
            </div>
            <div class="col-lg-6">
              <div class="input-group">
                <input type="text" class="form-control" ng-model="item.expDate"
placeholder="Expiration Date"
                  datepicker-popup="dd-MMMM-yyyy" is-open="expDateOpen"
                /> <span class="input-group-addon btn btn-default" ng-
click="$event.preventDefault();$event.stopPropagation();expDateOpen=true;"
                  ng-init="expDateOpen=false"
                > <i class="glyphicon glyphicon-calendar"></i>
                </span>

              </div>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-3">
```

```
            <label for="quantity">Quantity<span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control {required: true}"
name="quantity" ng-model="item.quantity"
              placeholder=""
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
(function($) {
	angular.module("cims").controller("biologyAddItemCtrl", function($scope,
$rootScope, $http,$state) {

		var chemicalId = $scope.$stateParams.chemicalId;

		$.extend($scope, {
			init : function() {
				$scope.item = {
					quantity : 1
				};
			},
			autocomplete : function(key) {
				return $http.post("biology/chemical/search", {
					name : key
				}).then(function(response) {
					return response.data;
				});
			},
			addItem : function() {
				$http.post("biology/chemical/add-item",
$scope.item).success(function(data) {
					alert(data.message);
					$state.go("^.view-definition",{chemicalId :
$scope.item.chemicalId});
				});
			},
			getChemicalName : function() {
				$http.post("biology/chemical/get", {
					id : chemicalId
				}).success(function(data) {
					$scope.item.chemicalName = data.name;
					$scope.item.chemicalId = chemicalId;
				});
```

```
                }

            });

            if (chemicalId) {
                $scope.getChemicalName();
            }

            $scope.init();

        });
})(jQuery);
<div>
  <div ui-view></div>
</div>
<div ng-controller="SearchbiologyChemicalCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Search Biology</h4>
      </div>
      <div class="panel-body">
        <form id="searchbiologyForm" class="form-horizontal" role="form" ng-
submit="search()">
          <div class="form-group">
            <div class="col-lg-1">
              <label for="name">Name</label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control" id="name" name="name" ng-
model="biologyForm.name"
                placeholder="Name"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-10">
    <table ng-table="chemicalTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="chemical in $data">
          <td data-title="'Name'"><a class="clickable"
            ui-sref="biology.chemical.update-definition({chemicalId : chemical.id})"
          >{{chemical.name}}</a></td>
          <td data-title="'Available Quantity'">{{chemical.totalQuantity}}</td>
          <td data-title="'Minimum Quantity'">{{chemical.minimum}}</td>
```

```
          <td data-title="'Action'"><a class="clickable"
            ui-sref="biology.chemical.update-definition({chemicalId : chemical.id})"
          >Update</a> | <a class="clickable"
            ui-sref="biology.chemical.add-item({chemicalId : chemical.id})"
          > Add Quantity</a></td>
        </tr>
        <tr ng-if="!$data || !$data.length">
          <td colspan="3"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Page: {{chemicalTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>
</div>
(function($) {
      angular.module("cims").controller("SearchbiologyChemicalCtrl",
                  function($scope, $rootScope, $http, ngTableParams) {
                        $.extend($scope, {
                              biologyForm : {},
                              search : function() {
                                    $scope.chemicalTable.reload();
                              }

                        });

                        $scope.chemicalTable = new ngTableParams({
                              page : 1,
                              count : 10
                        }, {
                              total : 0,
                              getData : function($defer, params) {
                                    $http.post("biology/chemical/search",
$scope.biologyForm).success(function(data) {
                                          params.total(data.length);

      $defer.resolve(data.slice((params.page() - 1) * params.count(), params.page()
                                                      * params.count()));
                                    });
                              }
                        });

                  });
})(jQuery);
<div ng-controller="UpdatebiologyChemicalCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
```

```
        <h4 class="panel-title">Update Biology Chemical</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="biology.chemical.add-
item({chemicalId : definition.id})"> Add Quantity</a>
        <br>
        <br>
        <br>
        <form id="searchbiologyForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Name">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="molFormula">Molecular Formula</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="molFormula" ng-
model="definition.molFormula"
                placeholder="Molecular Formula"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="color">Color</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="color" ng-
model="definition.color" placeholder="Color">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="texture">Texture</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="texture" ng-
model="definition.texture" placeholder="Texture">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="grade">Grade</label>
            </div>
            <div class="col-lg-8">
```

```
              <input type="text" class="form-control" name="grade" ng-
model="definition.grade" placeholder="Grade">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="hygroscopic">Hygroscopic</label>
            </div>
            <div class="col-lg-8">
              <span class="btn-group"> <label class="btn btn-primary active"
                ng-class="{active:(definition.hygroscopic == true)}"
              > <input type="radio" ng-value="true" ng-model="definition.hygroscopic"
checked="checked"
                  name="hygroscopic" class="ng-pristine ng-valid" value="true"
                > Yes
              </label> <label class="btn btn-primary" ng-
class="{active:(definition.hygroscopic == false)}"> <input
                  type="radio" ng-model="definition.hygroscopic" ng-value="false"
name="hygroscopic"
                  class="ng-pristine ng-valid" value="false"
                > No
              </label>
              </span>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="explosive">Explosive</label>
            </div>
            <div class="col-lg-8">

              <span class="btn-group"> <label class="btn btn-primary active"
                ng-class="{active:(definition.explosive == true)}"
              > <input type="radio" ng-value="true" ng-model="definition.explosive"
checked="checked"
                  name="explosive" class="ng-pristine ng-valid" value="true"
                > Yes
              </label> <label class="btn btn-primary" ng-
class="{active:(definition.explosive == false)}"> <input
                  type="radio" ng-model="definition.explosive" ng-value="false"
name="explosive"
                  class="ng-pristine ng-valid" value="false"
                > No
              </label>
              </span>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="thresh">Thresh</label>
            </div>
            <div class="col-lg-8">
```

```
              <input type="text" class="form-control" name="thresh" ng-
model="definition.thresh" placeholder="Thresh">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum" placeholder="Minimum Quantity">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Save</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("UpdatebiologyChemicalCtrl",
function($scope, $rootScope, $http) {

            var chemicalId = $scope.$stateParams.chemicalId;

            $.extend($scope, {
                  definition : {},
                  get : function() {
                        $http.post("biology/chemical/get", {
                              id : chemicalId
                        }).success(function(data) {
                              $scope.definition = data;
                        });
                  },
                  update : function() {
                        $http.post("biology/chemical/update-definition",
$scope.definition).success(function(data) {
                              alert(data.message);
                        });
                  }
            });

            $scope.get();
      });
})(jQuery);
<div ng-controller="ViewBiologyChemicalDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
```

```
  <h4 class="panel-title">Biology Chemical - {{definition.name}}</h4>
</div>
<div class="panel-body">
  <div class="container-fluid">
    <div class="row">
      <div class="col-lg-2">
        <label>Name</label>
      </div>
      <div class="col-lg-8">{{definition.name}}</div>
    </div>
    <br>
    <div class="row">
      <div class="col-lg-2">
        <label>Available Quantity</label>
      </div>
      <div class="col-lg-8">{{definition.totalQuantity}}</div>
    </div>
    <br>
    <div class="row">
      <div class="col-lg-2">
        <label>Minimum Quantity</label>
      </div>
      <div class="col-lg-8">{{definition.minimum}}</div>
    </div>
    <br>
    <div class="row">
      <div class="col-lg-2">
        <label>Molecular Formula</label>
      </div>
      <div class="col-lg-8">{{definition.molFormula}}</div>
    </div>
    <br>
    <div class="row">
      <div class="col-lg-2">
        <label>Color</label>
      </div>
      <div class="col-lg-8">{{definition.color}}</div>
    </div>
    <br>
    <div class="row">
      <div class="col-lg-2">
        <label>Texture</label>
      </div>
      <div class="col-lg-8">{{definition.texture}}</div>
    </div>
    <br>
    <div class="row">
      <div class="col-lg-2">
        <label>Grade</label>
      </div>
      <div class="col-lg-8">{{definition.grade}}</div>
    </div>
    <br>
```

```
          <div class="row">
            <div class="col-lg-2">
              <label>Hygroscopic</label>
            </div>
            <div class="col-lg-8">{{definition.hygroscopic}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Explosive</label>
            </div>
            <div class="col-lg-8">{{definition.explosive}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Updated Date</label>
            </div>
            <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("ViewBiologyChemicalDefCtrl",
                  function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
                        var definitionId = $scope.$stateParams.chemicalId;

                        $.extend($scope, {
                              definition : {},
                              get : function() {
                                    $http.post("biology/chemical/get", {id :
definitionId})
                                      .success(function(data) {
                                        $scope.definition  = data;
                                    });
                              }

                        });

                        $scope.get();

                  });
})(jQuery);
<div ng-controller="biologyAddLabwareDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add Biology Labware Definition</h4>
    </div>
    <div class="panel-body">
```

```
      <form id="searchbiologyForm" class="form-horizontal" role="form" ui-validate
ui-model="definition"
        ng-submit="add()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="Name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="type">Type</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="type" ng-
model="definition.type"
              placeholder="Type"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="size">Size</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
```

```
  </div>
</div>
(function($) {
	angular.module("cims").controller("biologyAddLabwareDefCtrl", function($scope,
$rootScope, $http) {
		$.extend($scope, {
			init : function() {
				$scope.definition = {
					minimum : 0,
					hygroscopic : true,
					explosive : true
				};
			},
			add : function() {
				$http.post("biology/labware/add",
$scope.definition).success(function(data) {
					alert(data.message);
					if (data.success) {
						$scope.init();
					}
				});
			}
		});

		$scope.init();

	});
})(jQuery);
<div ng-controller="biologyAddLabwareItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Add Biology Labware Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchbiologyForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="chemicalName"> Labware Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="labwareId" ng-
model="item.labwareId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="labwareName" name="labwareName"
                ng-model="item.labwareName" placeholder="Autocomplete Labware Name"
                typeahead="labware as labware.name for labware in
autocomplete($viewValue)  | limitTo:8"
```

```
                typeahead-wait-ms="500" typeahead-on-
select="item.labwareId=$model.id;item.labwareName=$model.name"
                >
              </div>
            </div>
            <input type="text" class="hide {required: true}" name="quantity" ng-
model="item.quantity"
              placeholder="Serial Number" ng-value="1"
            >
            <div class="form-group">
              <div class="col-lg-3">
                <label for="serialNumber">Serial Number<span
class="required">*</span></label>
              </div>
              <div class="col-lg-6">
                <input type="text" class="form-control {required: true}"
name="serialNumber" ng-model="item.serialNumber"
                  placeholder="Serial Number"
                >
              </div>
            </div>

            <div class="form-group">
              <div class="col-lg-12">
                <button type="submit" class="btn btn-primary">Add</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
</div>
(function($){
       angular.module("cims").

controller("biologyAddLabwareItemCtrl",function($scope,$rootScope,$http,$state){

              var labwareId = $scope.$stateParams.labwareId;

                $.extend($scope,{
                       init : function(){
                            $scope.item = {
                                     quantity : 1
                              };
                       },
                       autocomplete : function(key){
                            return $http.post("biology/labware/search", {name : key} )
                              .then(function(response) {
                                     return response.data;
                            });
                       },
                       addItem : function(){
                            $http.post("biology/labware/add-item",$scope.item)
```

```
                        .success(function(data){
                            alert(data.message);
                            $state.go("^.view-
definition",{labwareId:$scope.item.labwareId});
                        });
                },
                    getLabwareName : function() {
                        $http.post("biology/labware/get", {
                            id : labwareId
                        }).success(function(data) {
                            $scope.item.labwareName = data.name;
                            $scope.item.labwareId = labwareId;
                        });
                    }
            });

            if(labwareId){
                $scope.getLabwareName();
            }

            $scope.init();

        });
})(jQuery);
<div>
  <div ui-view></div>
</div>
<div>
  <div id="searchLabwareItemModal" class="modal fade">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
          <h3 class="modal-title">Biology Labware Items -
{{selectedLabware.name}}</h3>
        </div>
        <div class="modal-body">
          <table ng-table="labwareItemTable" class="table  table-bordered">
            <tbody>
              <tr ng-repeat="labwareItem in $data">
                <td data-title="'Serial Number'">{{labwareItem.serialNumber}}</td>
                <td data-title="'Created Date'">{{labwareItem.createdDate |
amDateFormat:'dddd, MMMM Do YYYY,
                    h:mm:ss a'}}</td>
                <td data-title="'Action'"><a class="clickable" ng-
click="sendRepairRequest(labwareItem)"> For
                    repair</a> | <a class="clickable" ng-
click="sendCondemnationRequest(labwareItem)"> For condemnation</a></td>
              </tr>
              <tr ng-if="!$data || !$data.length">
                <td colspan="3"><div class="text-center">
                    <strong>No Result Found</strong>
```

```
                </div></td>
              </tr>
            </tbody>
            <tfoot>
              <tr>
                <td>Page: {{labwareItemTable.page()}}</td>
              </tr>
            </tfoot>
          </table>

        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
        </div>
      </div>
    </div>
  </div>
</div>
(function($) {
       angular.module("cims").controller("SearchbiologyLabwareCtrl", function($scope,
$rootScope, $http, ngTableParams) {
             $.extend($scope, {
                    biologyForm : {},
                    selectedLabware : {},
                    search : function() {
                           $scope.labwareTable.reload();
                    },
                    selectLabware: function(labware){
                           $scope.selectedLabware = labware;
                           $scope.searchItem();
                    },
                    searchItem : function(){
       $scope.labwareItemTable.reload();
      },
                    templates : $rootScope.extendTemplates({
                           searchLabwareItemModal :
"app/biology/labware/biology.labware.item.search.html"
                    }),
                    sendRepairRequest : function(labwareItem){
                           $http.post("biology/labware/item/repair-alert/create", {id
: labwareItem.id})
                              .success(function(data) {
                                    alert(data.message);
                           });
                    },
                    sendCondemnationRequest: function(labwareItem){
                           $http.post("biology/labware/item/condemnation-
alert/create", {id : labwareItem.id})
                       .success(function(data) {
                             alert(data.message);
                    });
                    }
```

```
		});

		$scope.labwareTable = new ngTableParams({
			page : 1,
			count : 10
		}, {
			total : 0,
			getData : function($defer, params) {
				$http.post("biology/labware/search",
$scope.biologyForm).success(function(data) {
					params.total(data.length);
					$defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
				});
			}
		});

		$scope.labwareItemTable = new ngTableParams({
			page : 1,
			count : 10
		}, {
			total : 0,
			getData : function($defer, params) {
				$http.post("biology/labware/item/search", {labwareId :
$scope.selectedLabware.id}).success(function(data) {
					params.total(data.length);
					$defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
				});
			}
		});

	});
})(jQuery);
<div ng-controller="UpdatebiologyLabwareCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Update Biology Labware</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="biology.labware.add-
item({labwareId : definition.id})">
          Add Quantity</a> <br> <br> <br>
        <form id="searchbiologyForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="Name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
```

```
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="type">Type</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="type" ng-
model="definition.type" placeholder="Type">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="size">Size</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="brokenQuantity">Broken Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="brokenQuantity" ng-
model="definition.brokenQuantity"
                placeholder="Broken Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="stainedQuantity">Stained Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="stainedQuantity" ng-
model="definition.stainedQuantity"
                placeholder="Stained Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
                placeholder="Minimum Quantity"
              >
```

```
            </div>
            <div class="form-group">
              <div class="col-lg-7">
                <button type="submit" class="btn btn-primary">Update</button>
              </div>
            </div>
        </form>
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("UpdatebiologyLabwareCtrl", function($scope,
$rootScope, $http) {

            var labwareId = $scope.$stateParams.labwareId;

            $.extend($scope, {
                  definition : {},
                  get : function() {
                        $http.post("biology/labware/get", {
                              id : labwareId
                        }).success(function(data) {
                              $scope.definition = data;
                        });
                  },
                  update : function() {
                        $http.post("biology/labware/update-definition",
$scope.definition).success(function(data) {
                              alert(data.message);
                        });
                  }
            });

            $scope.get();
      });
})(jQuery);
<div ng-controller="ViewBiologyLabwareDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Biology Labware - {{definition.name}}</h4>
      </div>
      <div class="panel-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-lg-2">
              <label>Name</label>
            </div>
            <div class="col-lg-8">{{definition.name}}</div>
          </div>
          <br>
          <div class="row">
```

```
            <div class="col-lg-2">
              <label>Available Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.totalQuantity}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Minimum Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.minimum}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Type</label>
            </div>
            <div class="col-lg-8">{{definition.type}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Size</label>
            </div>
            <div class="col-lg-8">{{definition.size}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Updated Date</label>
            </div>
            <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("ViewBiologyLabwareDefCtrl",
                  function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
                        var definitionId = $scope.$stateParams.labwareId;

                        $.extend($scope, {
                              definition : {},
                              get : function() {
                                    $http.post("biology/labware/get", {id :
definitionId})
                                      .success(function(data) {
                                        $scope.definition  = data;
                                    });
                              }
```

```
                        });

                        $scope.get();

                });
})(jQuery);
(function($) {
      $.extend(true, window.routeConfig, {
            states : {
                  'biology' : {
                        url : '/biology',
                        templateUrl : 'app/biology/biology.html'
                  },
                  'biology.chemical' : {
                        url : '/chemical',
                        templateUrl : 'app/biology/chemical/biology.chemical.html'
                  },
                  'biology.chemical.search' : {
                        url : '/search',
                        templateUrl :
'app/biology/chemical/biology.chemical.search.html'
                  },
                  'biology.chemical.add-definition' : {
                        url : '/add-definition',
                        templateUrl : 'app/biology/chemical/biology.chemical.add-
definition.html'
                  },
                  'biology.chemical.update-definition' : {
                        url : '/update-definition?chemicalId',
                        templateUrl :
'app/biology/chemical/biology.chemical.update-definition.html'
                  },
                  'biology.chemical.add-item' : {
                        url : '/add-item?chemicalId',
                        templateUrl : 'app/biology/chemical/biology.chemical.add-
item.html'
                  },
                  'biology.chemical.view-definition' : {
                        url : '/view-definition?chemicalId',
                        templateUrl : 'app/biology/chemical/biology.chemical.view-
definition.html'
                  },
                  'biology.labware' : {
                        url : '/labware',
                        templateUrl : 'app/biology/labware/biology.labware.html'
                  },
                  'biology.labware.add-definition' : {
                        url : '/add-definition',
                        templateUrl : 'app/biology/labware/biology.labware.add-
definition.html'
                  },
                  'biology.labware.search' : {
```

```
                        url : '/search',
                        templateUrl :
'app/biology/labware/biology.labware.search.html'
                  },
                  'biology.labware.update-definition' : {
                        url : '/update-definition?labwareId',
                        templateUrl : 'app/biology/labware/biology.labware.update-
definition.html'
                  },
                  'biology.labware.add-item' : {
                        url : '/add-item?labwareId',
                        templateUrl : 'app/biology/labware/biology.labware.add-
item.html'
                  },
                  'biology.labware.view-definition' : {
                        url : '/view-definition?labwareId',
                        templateUrl : 'app/biology/labware/biology.labware.view-
definition.html'
                  },

            }
      });
})(jQuery);
<div>
  <div ui-view></div>
</div>
<div ng-controller="ChemistryAddChemicalDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add Chemistry Definition</h4>
    </div>
    <div class="panel-body">
      <form id="searchChemistryForm" class="form-horizontal" role="form" ui-validate
ui-model="definition"
        ng-submit="add()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Name">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="molFormula">Molecular Formula</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="molFormula" ng-
model="definition.molFormula"
              placeholder="Molecular Formula"
```

```
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="color">Color</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="color" ng-
model="definition.color" placeholder="Color">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="texture">Texture</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="texture" ng-
model="definition.texture" placeholder="Texture">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="grade">Grade</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="grade" ng-
model="definition.grade" placeholder="Grade">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="hygroscopic">Hygroscopic</label>
          </div>
          <div class="col-lg-8">
            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.hygroscopic == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.hygroscopic"
checked="checked"
                name="hygroscopic" class="ng-pristine ng-valid"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.hygroscopic == false)}"> <input
                type="radio" ng-model="definition.hygroscopic" ng-value="false"
name="hygroscopic"
                class="ng-pristine ng-valid"
              > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
```

```
            <label for="explosive">Explosive</label>
          </div>
          <div class="col-lg-8">

            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.explosive == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.explosive"
checked="checked" name="explosive"
                class="ng-pristine ng-valid"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.explosive == false)}"> <input
                type="radio" ng-model="definition.explosive" ng-value="false"
name="explosive"
                class="ng-pristine ng-valid"
              > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="thresh">Thresh</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="thresh" ng-
model="definition.thresh" placeholder="Thresh">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
(function($) {
```

```
	angular.module("cims").controller("ChemistryAddChemicalDefCtrl",
function($scope, $rootScope, $http) {
		$.extend($scope, {
			init : function() {
				$scope.definition = {
					minimum : 0,
					hygroscopic : true,
					explosive : true
				};
			},
			add : function() {
				$http.post("chemistry/chemical/add",
$scope.definition).success(function(data) {
					alert(data.message);
					if (data.success) {
						$scope.init();
					}
				});
			}
		});

		$scope.init();

	});
})(jQuery);
<div ng-controller="ChemistryAddItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title" ng-click="alert('');">Add Chemistry Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchChemistryForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="chemicalName"> Chemical Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="chemicalId" ng-
model="item.chemicalId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="chemicalName" name="chemicalName"
                ng-model="item.chemicalName" placeholder="Autocomplete Chemical Name"
                typeahead="chemical as chemical.name for chemical in
autocomplete($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-
select="item.chemicalId=$model.id;item.chemicalName=$model.name"
              >
            </div>
```

```
          </div>

          <div class="form-group">
            <div class="col-lg-3">
              <label for="expDate"> Expiration Date</label>
            </div>
            <div class="col-lg-6">
              <div class="input-group">
                <input type="text" class="form-control" ng-model="item.expDate"
placeholder="Expiration Date"
                  datepicker-popup="dd-MMMM-yyyy" is-open="expDateOpen"
                /> <span class="input-group-addon btn btn-default" ng-
click="$event.preventDefault();$event.stopPropagation();expDateOpen=true;"
                  ng-init="expDateOpen=false"
                > <i class="glyphicon glyphicon-calendar"></i>
                </span>

              </div>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-3">
              <label for="quantity">Quantity<span class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control {required: true}"
name="quantity" ng-model="item.quantity"
                placeholder=""
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-12">
              <button type="submit" class="btn btn-primary">Add</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
(function($) {
       angular.module("cims").controller("ChemistryAddItemCtrl", function($scope,
$rootScope, $http,$state,$stateParams) {

             var chemicalId = $scope.$stateParams.chemicalId;

             $.extend($scope, {
                   init : function() {
                         $scope.item = {
                               quantity : 1
                         };
                   },
```

```
                        autocomplete : function(key) {
                                return $http.post("chemistry/chemical/search", {
                                        name : key
                                }).then(function(response) {
                                        return response.data;
                                });
                        },
                        addItem : function() {
                                $http.post("chemistry/chemical/add-item",
$scope.item).success(function(data) {
                                        alert(data.message);
                                        $state.go("^.view-definition",{chemicalId :
$scope.item.chemicalId});
                                        $scope.init();
                                });
                        },
                        getChemicalName : function() {
                                $http.post("chemistry/chemical/get", {
                                        id : chemicalId
                                }).success(function(data) {
                                        $scope.item.chemicalName = data.name;
                                        $scope.item.chemicalId = chemicalId;
                                });
                        }

                });

                if (chemicalId) {
                        $scope.getChemicalName();
                }

                $scope.init();

        });
})(jQuery);
<div>
  <div ui-view></div>
</div>
<div ng-controller="SearchChemistryChemicalCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Search Chemistry</h4>
      </div>
      <div class="panel-body">
        <form id="searchChemistryForm" class="form-horizontal" role="form" ng-
submit="search()">
          <div class="form-group">
            <div class="col-lg-1">
              <label for="name">Name</label>
            </div>
            <div class="col-lg-6">
```

```
              <input type="text" class="form-control" id="name" name="name" ng-
model="chemistryForm.name"
                placeholder="Name"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-10">
    <table ng-table="chemicalTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="chemical in $data">
          <td data-title="'Name'"><a class="clickable"
            ui-sref="chemistry.chemical.update-definition({chemicalId :
chemical.id})"
          >{{chemical.name}}</a></td>
          <td data-title="'Available Quantity'">{{chemical.totalQuantity}}</td>
          <td data-title="'Minimum Quantity'">{{chemical.minimum}}</td>
          <td data-title="'Action'"><a class="clickable"
            ui-sref="chemistry.chemical.update-definition({chemicalId :
chemical.id})"
          >Update</a> | <a class="clickable"
            ui-sref="chemistry.chemical.add-item({chemicalId : chemical.id})"
          > Add Quantity</a></td>
        </tr>
        <tr ng-if="!$data || !$data.length">
          <td colspan="3"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Page: {{chemicalTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>
</div>
(function($) {
      angular.module("cims").controller("SearchChemistryChemicalCtrl",
                  function($scope, $rootScope, $http, ngTableParams) {
                        $.extend($scope, {
                              chemistryForm : {},
                              search : function() {
                                    $scope.chemicalTable.reload();
```

```
                        }

                    });

                    $scope.chemicalTable = new ngTableParams({
                        page : 1,
                        count : 10
                    }, {
                        total : 0,
                        getData : function($defer, params) {
                            $http.post("chemistry/chemical/search",
$scope.chemistryForm).success(function(data) {
                                params.total(data.length);

        $defer.resolve(data.slice((params.page() - 1) * params.count(), params.page()
                                    * params.count()));
                            });
                        }
                    });

                });
})(jQuery);
<div ng-controller="UpdateChemistryChemicalCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Update Chemistry Chemical</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="chemistry.chemical.add-
item({chemicalId : definition.id})"> Add Quantity</a>
        <br>
        <br>
        <br>
        <form id="searchChemistryForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Name">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="molFormula">Molecular Formula</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="molFormula" ng-
model="definition.molFormula"
                placeholder="Molecular Formula"
```

```
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="color">Color</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="color" ng-
model="definition.color" placeholder="Color">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="texture">Texture</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="texture" ng-
model="definition.texture" placeholder="Texture">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="grade">Grade</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="grade" ng-
model="definition.grade" placeholder="Grade">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="hygroscopic">Hygroscopic</label>
          </div>
          <div class="col-lg-8">
            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.hygroscopic == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.hygroscopic"
checked="checked"
                name="hygroscopic" class="ng-pristine ng-valid" value="true"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.hygroscopic == false)}"> <input
                type="radio" ng-model="definition.hygroscopic" ng-value="false"
name="hygroscopic"
                class="ng-pristine ng-valid" value="false"
              > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
```

```
              <label for="explosive">Explosive</label>
            </div>
            <div class="col-lg-8">

              <span class="btn-group"> <label class="btn btn-primary active"
                ng-class="{active:(definition.explosive == true)}"
              > <input type="radio" ng-value="true" ng-model="definition.explosive"
checked="checked"
                  name="explosive" class="ng-pristine ng-valid" value="true"
                > Yes
              </label> <label class="btn btn-primary" ng-
class="{active:(definition.explosive == false)}"> <input
                  type="radio" ng-model="definition.explosive" ng-value="false"
name="explosive"
                  class="ng-pristine ng-valid" value="false"
                > No
              </label>
              </span>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="thresh">Thresh</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="thresh" ng-
model="definition.thresh" placeholder="Thresh">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum" placeholder="Minimum Quantity">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Save</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("UpdateChemistryChemicalCtrl",
function($scope, $rootScope, $http) {

            var chemicalId = $scope.$stateParams.chemicalId;
```

```
		$.extend($scope, {
			definition : {},
			get : function() {
				$http.post("chemistry/chemical/get", {
					id : chemicalId
				}).success(function(data) {
					$scope.definition = data;
				});
			},
			update : function() {
				$http.post("chemistry/chemical/update-definition",
$scope.definition).success(function(data) {
					alert(data.message);
				});
			}
		});

		$scope.get();
	});
})(jQuery);
<div ng-controller="ViewChemistryChemicalDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Chemistry Chemical - {{definition.name}}</h4>
      </div>
      <div class="panel-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-lg-2">
              <label>Name</label>
            </div>
            <div class="col-lg-8">{{definition.name}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Available Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.totalQuantity}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Minimum Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.minimum}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Molecular Formula</label>
```

```
            </div>
            <div class="col-lg-8">{{definition.molFormula}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Color</label>
            </div>
            <div class="col-lg-8">{{definition.color}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Texture</label>
            </div>
            <div class="col-lg-8">{{definition.texture}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Grade</label>
            </div>
            <div class="col-lg-8">{{definition.grade}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Hygroscopic</label>
            </div>
            <div class="col-lg-8">{{definition.hygroscopic}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Explosive</label>
            </div>
            <div class="col-lg-8">{{definition.explosive}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Updated Date</label>
            </div>
            <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("ViewChemistryChemicalDefCtrl",
```

```
                    function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
                            var definitionId = $scope.$stateParams.chemicalId;

                            $.extend($scope, {
                                   definition : {},
                                   get : function() {
                                          $http.post("chemistry/chemical/get", {id :
definitionId})
                                            .success(function(data) {
                                              $scope.definition  = data;
                                          });
                                   }

                            });

                            $scope.get();

                    });
})(jQuery);
<div ng-controller="ChemistryAddLabwareDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add Chemistry Labware Definition</h4>
    </div>
    <div class="panel-body">
      <form id="searchChemistryForm" class="form-horizontal" role="form" ui-validate
ui-model="definition"
        ng-submit="add()"
       >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="Name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="type">Type</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="type" ng-
model="definition.type"
              placeholder="Type"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="size">Size</label>
```

```
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("ChemistryAddLabwareDefCtrl",
function($scope, $rootScope, $http) {
            $.extend($scope, {
                  init : function() {
                        $scope.definition = {
                              minimum : 0,
                              hygroscopic : true,
                              explosive : true
                        };
                  },
                  add : function() {
                        $http.post("chemistry/labware/add",
$scope.definition).success(function(data) {
                              alert(data.message);
                              if (data.success) {
                                    $scope.init();
                              }
                        });
                  }
            });

            $scope.init();

      });
```

```
})(jQuery);
<div ng-controller="ChemistryAddLabwareItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Add Chemistry Labware Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchChemistryForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="chemicalName"> Labware Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="labwareId" ng-
model="item.labwareId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="labwareName" name="labwareName"
                ng-model="item.labwareName" placeholder="Autocomplete Labware Name"
                typeahead="labware as labware.name for labware in
autocomplete($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-
select="item.labwareId=$model.id;item.labwareName=$model.name"
              >
            </div>
          </div>
          <input type="text" class="hide {required: true}" name="quantity" ng-
model="item.quantity"
            placeholder="Serial Number" ng-value="1"
          >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="serialNumber">Serial Number<span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control {required: true}"
name="serialNumber" ng-model="item.serialNumber"
                placeholder="Serial Number"
              >
            </div>
          </div>

          <div class="form-group">
            <div class="col-lg-12">
              <button type="submit" class="btn btn-primary">Add</button>
            </div>
          </div>
        </form>
```

```
      </div>
    </div>
  </div>
</div>
(function($){
       angular.module("cims").

controller("ChemistryAddLabwareItemCtrl",function($scope,$rootScope,$http,$state){

             var labwareId = $scope.$stateParams.labwareId;

               $.extend($scope,{
                      init : function(){
                           $scope.item = {
                                    quantity : 1
                             };
                      },
                      autocomplete : function(key){
                           return $http.post("chemistry/labware/search", {name : key}
)
                             .then(function(response) {
                                    return response.data;
                           });
                      },
                      addItem : function(){
                           $http.post("chemistry/labware/add-item",$scope.item)
                             .success(function(data){
                                  alert(data.message);
                                  $state.go("^.view-
definition",{labwareId:$scope.item.labwareId});
                             });
                      },
                           getLabwareName : function() {
                                  $http.post("chemistry/labware/get", {
                                         id : labwareId
                                  }).success(function(data) {
                                         $scope.item.labwareName = data.name;
                                         $scope.item.labwareId = labwareId;
                                  });
                           }
               });

               if(labwareId){
                    $scope.getLabwareName();
               }

               $scope.init();

        });
})(jQuery);
<div>
  <div ui-view></div>
</div>
```

```
<div>
  <div id="searchLabwareItemModal" class="modal fade">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
          <h3 class="modal-title">Chemistry Labware Items -
{{selectedLabware.name}}</h3>
        </div>
        <div class="modal-body">
          <table ng-table="labwareItemTable" class="table  table-bordered">
            <tbody>
              <tr ng-repeat="labwareItem in $data">
                <td data-title="'Serial Number'">{{labwareItem.serialNumber}}</td>
                <td data-title="'Created Date'">{{labwareItem.createdDate |
amDateFormat:'dddd, MMMM Do YYYY,
                    h:mm:ss a'}}</td>
                <td data-title="'Action'"><a class="clickable" ng-
click="sendRepairRequest(labwareItem)"> For
                    repair</a> | <a class="clickable" ng-
click="sendCondemnationRequest(labwareItem)"> For condemnation</a></td>
              </tr>
              <tr ng-if="!$data || !$data.length">
                <td colspan="3"><div class="text-center">
                    <strong>No Result Found</strong>
                  </div></td>
              </tr>
            </tbody>
            <tfoot>
              <tr>
                <td>Page: {{labwareItemTable.page()}}</td>
              </tr>
            </tfoot>
          </table>

        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
        </div>
      </div>
    </div>
  </div>
</div>
<div ng-controller="SearchChemistryLabwareCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Search Chemistry Labware {{count}}</h4>
      </div>
      <div class="panel-body">
```

```
        <form id="searchChemistryForm" class="form-horizontal" role="form" ng-
submit="search()">
          <div class="form-group">
            <div class="col-lg-1">
              <label for="name">Name</label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control" id="name" name="name" ng-
model="chemistryForm.name"
                placeholder="Name"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-10">
    <table ng-table="labwareTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="labware in $data">
          <td data-title="'Name'"><a class="clickable"
            ui-sref="chemistry.labware.update-definition({labwareId : labware.id})"
          >{{labware.name}}</a></td>
          <td data-title="'Available Quantity'">{{labware.totalQuantity}}</td>
          <td data-title="'Minimum Quantity'">{{labware.minimum}}</td>
          <td data-title="'Action'"><a class="clickable"
            ui-sref="chemistry.labware.update-definition({labwareId : labware.id})"
          >Update</a>
          | <a class="clickable"
            ui-sref="chemistry.labware.add-item({labwareId : labware.id})"
          > Add Quantity</a>
          | <a class="clickable" data-toggle="modal" data-
target="#searchLabwareItemModal"

          > <span ng-click="selectLabware(labware)">View Items</span></a>


          </td>
        </tr>
        <tr ng-if="!$data || !$data.length">
          <td colspan="3"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
```

```
          <td>Page: {{labwareTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>

  <div ng-include src="templates.searchLabwareItemModal">
  </div>
</div>
(function($) {
      angular.module("cims").controller("SearchChemistryLabwareCtrl",
function($scope, $rootScope, $http, ngTableParams) {
            $.extend($scope, {
                  chemistryForm : {},
                  selectedLabware : {},
                  search : function() {
                        $scope.labwareTable.reload();
                  },
                  selectLabware: function(labware){
                        $scope.selectedLabware = labware;
                        $scope.searchItem();
                  },
                  searchItem : function(){
      $scope.labwareItemTable.reload();
      },
                  templates : $rootScope.extendTemplates({
                        searchLabwareItemModal :
"app/chemistry/labware/chemistry.labware.item.search.html"
                  }),
                  sendRepairRequest : function(labwareItem){
                        $http.post("chemistry/labware/item/repair-alert/create",
{id : labwareItem.id})
                           .success(function(data) {
                                 alert(data.message);
                        });
                  },
                  sendCondemnationRequest: function(labwareItem){
                        $http.post("chemistry/labware/item/condemnation-
alert/create", {id : labwareItem.id})
                     .success(function(data) {
                           alert(data.message);
                  });
                  }

            });

            $scope.labwareTable = new ngTableParams({
                  page : 1,
                  count : 10
            }, {
                  total : 0,
                  getData : function($defer, params) {
```

```
                        $http.post("chemistry/labware/search",
$scope.chemistryForm).success(function(data) {
                                params.total(data.length);
                                $defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
                        });
                }
        });

        $scope.labwareItemTable = new ngTableParams({
                page : 1,
                count : 10
        }, {
                total : 0,
                getData : function($defer, params) {
                        $http.post("chemistry/labware/item/search", {labwareId :
$scope.selectedLabware.id}).success(function(data) {
                                params.total(data.length);
                                $defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
                        });
                }
        });

    });
})(jQuery);
<div ng-controller="UpdateChemistryLabwareCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Update Chemistry Labware</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="chemistry.labware.add-
item({labwareId : definition.id})"> Add
          Quantity</a> <br> <br> <br>
        <form id="searchChemistryForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="Name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="type">Type</label>
            </div>
            <div class="col-lg-8">
```

```
              <input type="text" class="form-control" name="type" ng-
model="definition.type" placeholder="Type">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="size">Size</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="brokenQuantity">Broken Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="brokenQuantity" ng-
model="definition.brokenQuantity"
                placeholder="Broken Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="stainedQuantity">Stained Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="stainedQuantity" ng-
model="definition.stainedQuantity"
                placeholder="Stained Quantity"
              >
            </div>
          </div>

          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
                placeholder="Minimum Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Update</button>
            </div>
          </div>
        </form>
```

```
      </div>
    </div>
  </div>
</div>
(function($) {
      angular.module("cims").controller("UpdateChemistryLabwareCtrl",
function($scope, $rootScope, $http) {

            var labwareId = $scope.$stateParams.labwareId;

            $.extend($scope, {
                  definition : {},
                  get : function() {
                        $http.post("chemistry/labware/get", {
                              id : labwareId
                        }).success(function(data) {
                              $scope.definition = data;
                        });
                  },
                  update : function() {
                        $http.post("chemistry/labware/update-definition",
$scope.definition).success(function(data) {
                              alert(data.message);
                        });
                  }
            });

            $scope.get();
      });


})(jQuery);
<div ng-controller="ViewChemistryLabwareDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Chemistry Labware - {{definition.name}}</h4>
      </div>
      <div class="panel-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-lg-2">
              <label>Name</label>
            </div>
            <div class="col-lg-8">{{definition.name}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Available Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.totalQuantity}}</div>
```

```
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Minimum Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.minimum}}</div>
          </div>
          <br>          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Type</label>
            </div>
            <div class="col-lg-8">{{definition.type}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Size</label>
            </div>
            <div class="col-lg-8">{{definition.size}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Updated Date</label>
            </div>
            <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
          </div>
        </div>
      </div>
    </div>
  </div>


</div>
(function($) {
      angular.module("cims").controller("ViewChemistryLabwareDefCtrl",
                  function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
                        var definitionId = $scope.$stateParams.labwareId;

                        $.extend($scope, {
                              definition : {},
                              get : function() {
                                    $http.post("chemistry/labware/get", {id :
definitionId})
                                      .success(function(data) {
                                        $scope.definition  = data;
                                    });
                              }
```

```
                        });

                        $scope.get();

                  });


})(jQuery);
(function($) {
      $.extend(true, window.routeConfig, {
            states : {
                  'chemistry' : {
                        url : '/chemistry',
                        templateUrl : 'app/chemistry/chemistry.html'
                  },
                  'chemistry.chemical' : {
                        url : '/chemical',
                        templateUrl :
'app/chemistry/chemical/chemistry.chemical.html'
                  },
                  'chemistry.chemical.search' : {
                        url : '/search',
                        templateUrl :
'app/chemistry/chemical/chemistry.chemical.search.html'
                  },
                  'chemistry.chemical.add-definition' : {
                        url : '/add-definition',
                        templateUrl :
'app/chemistry/chemical/chemistry.chemical.add-definition.html'
                  },
                  'chemistry.chemical.update-definition' : {
                        url : '/update-definition?chemicalId',
                        templateUrl :
'app/chemistry/chemical/chemistry.chemical.update-definition.html'
                  },
                  'chemistry.chemical.view-definition' : {
                        url : '/view-definition?chemicalId',
                        templateUrl :
'app/chemistry/chemical/chemistry.chemical.view-definition.html'
                  },
                  'chemistry.chemical.add-item' : {
                        url : '/add-item?chemicalId',
                        templateUrl :
'app/chemistry/chemical/chemistry.chemical.add-item.html'
                  },
                  'chemistry.labware' : {
                        url : '/labware',
                        templateUrl :
'app/chemistry/labware/chemistry.labware.html'
                  },
                  'chemistry.labware.add-definition' : {
                        url : '/add-definition',
```

```
                        templateUrl :
'app/chemistry/labware/chemistry.labware.add-definition.html'
                  },
                  'chemistry.labware.search' : {
                        url : '/search',
                        templateUrl :
'app/chemistry/labware/chemistry.labware.search.html'
                  },
                  'chemistry.labware.update-definition' : {
                        url : '/update-definition?labwareId',
                        templateUrl :
'app/chemistry/labware/chemistry.labware.update-definition.html'
                  },
                  'chemistry.labware.add-item' : {
                        url : '/add-item?labwareId',
                        templateUrl :
'app/chemistry/labware/chemistry.labware.add-item.html'
                  },
                  'chemistry.labware.view-definition' : {
                        url : '/view-definition?labwareId',
                        templateUrl :
'app/chemistry/labware/chemistry.labware.view-definition.html'
                  },

            }
      });
})(jQuery);

<div>
  <div ui-view></div>
</div>

<ol class="breadcrumb">
  <li ng-repeat="breadcrumb in breadcrumbs.getAll()">
    <a href="#{{breadcrumb.path}}" ng-hide="$last">{{breadcrumb.label}}</a>
    <span ng-show="$last">{{breadcrumb.label}}</span>
  </li>
</ol>

<footer id="footer">
    <p class="text-muted credit"><center><small>University of the Philippines Manila
Copyright 2014 ver 1.0.0</small></center></p>
</footer>

<div id="header" class="navbar navbar-inverse">
  <div class="container">
    <div class="row-fluid">
      <div class="col-md-1 col-sm-2 pagination-centered">
        <img src="assets/img/logo_small.jpg" class="img-responsive centered"
          title="College of Arts and Sciences"
        />
      </div>
```

```
      <div class="col-md-8 col-sm-7" style="vertical-align: middle">
        <h3 class="site-title">Laboratory Inventory Monitoring System</h3>
      </div>
      <div class="col-md-2" style="vertical-align: middle">
        <span ng-if="user && user.firstname" style="vertical-align: middle">Hello
{{user.firstname}}</span>
      </div>
      <div class="col-md-1">
        <a ng-show="isLoggedIn" class="btn btn-primary" ui-sref="login" ng-
click="doLogout()">Logout</a>
      </div>
    </div>
  </div>
</div>

<div id="sidebar" ng-controller="SideBarCtrl" ng-show="isLoggedIn">
  <div class="panel-group">
    <div class="panel panel-default" ui-restricted-
view="lab_attendant,lab_faculty_coordinator,supply_officer">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a ui-sref="dashboard"> Dashboard </a>
        </h4>
      </div>
    </div>
  </div>

  <div class="panel-group" id="accordion">
    <div class="panel panel-default" ui-restricted-view="system_admin">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a class="accordion-toggle" data-toggle="collapse" data-parent="#sidebar
#accordion" href="#userCollapse">
            Manage Users </a>
        </h4>
      </div>
      <div id="userCollapse" class="panel-collapse collapse">
        <div class="panel-body">
          <ul class="nav nav-pills nav-stacked">
            <li><a ui-sref="user.search">Search</a></li>
            <li><a ui-sref="user.add">Add User</a></li>
          </ul>
        </div>
      </div>
    </div>

    <div class="panel panel-default" ui-restricted-view="chemistry_lab_attendant">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a class="accordion-toggle" data-toggle="collapse" data-parent="#sidebar
#accordion" href="#chemistryCollapse">
            Chemistry </a>
        </h4>
```

```
      </div>
      <div id="chemistryCollapse" class="panel-collapse collapse">
        <div class="panel-body">
          <ul class="nav nav-pills nav-stacked">
            <li><a ui-sref="chemistry.chemical.search">Chemical - Search</a></li>
            <li><a ui-sref="chemistry.chemical.add-definition">Chemical - Add
Definition</a></li>
            <li><a ui-sref="chemistry.chemical.add-item">Chemical - Add Item
Quantity</a></li>
            <li><a ui-sref="chemistry.labware.search">Labware - Search</a></li>
            <li><a ui-sref="chemistry.labware.add-definition">Labware - Add
Definition</a></li>
            <li><a ui-sref="chemistry.labware.add-item">Labware - Add Item</a></li>
          </ul>
        </div>
      </div>
    </div>

    <div class="panel panel-default" ui-restricted-view="physics_lab_attendant">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a class="accordion-toggle" data-toggle="collapse" data-parent="#sidebar
#accordion" href="#physicsCollapse">
            Physics </a>
        </h4>
      </div>
      <div id="physicsCollapse" class="panel-collapse collapse">
        <div class="panel-body">
          <ul class="nav nav-pills nav-stacked">
            <li><a ui-sref="physics.chemical.search">Chemical - Search</a></li>
            <li><a ui-sref="physics.chemical.add-definition">Chemical - Add
Definition</a></li>
            <li><a ui-sref="physics.chemical.add-item">Chemical - Add Item
Quantity</a></li>
            <li><a ui-sref="physics.labware.search">Labware - Search</a></li>
            <li><a ui-sref="physics.labware.add-definition">Labware - Add
Definition</a></li>
            <li><a ui-sref="physics.labware.add-item">Labware - Add Item</a></li>
          </ul>
        </div>
      </div>
    </div>

    <div class="panel panel-default" ui-restricted-view="biology_lab_attendant">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a class="accordion-toggle" data-toggle="collapse" data-parent="#sidebar
#accordion" href="#biologyCollapse">
            Biology </a>
        </h4>
      </div>
      <div id="biologyCollapse" class="panel-collapse collapse">
        <div class="panel-body">
```

```
          <ul class="nav nav-pills nav-stacked">
            <li><a ui-sref="biology.chemical.search">Chemical - Search</a></li>
            <li><a ui-sref="biology.chemical.add-definition">Chemical - Add
Definition</a></li>
            <li><a ui-sref="biology.chemical.add-item">Chemical - Add Item
Quantity</a></li>
            <li><a ui-sref="biology.labware.search">Labware - Search</a></li>
            <li><a ui-sref="biology.labware.add-definition">Labware - Add
Definition</a></li>
            <li><a ui-sref="biology.labware.add-item">Labware - Add Item</a></li>
          </ul>
        </div>
      </div>
    </div>

    <div class="panel panel-default" ui-restricted-view="mathcs_lab_attendant">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a class="accordion-toggle" data-toggle="collapse" data-parent="#sidebar
#accordion" href="#mathcsCollapse">
            Math CS </a>
        </h4>
      </div>
      <div id="mathcsCollapse" class="panel-collapse collapse">
        <div class="panel-body">
          <ul class="nav nav-pills nav-stacked">
            <li><a ui-sref="mathcs.labware.search">Machine - Search</a></li>
            <li><a ui-sref="mathcs.labware.add-definition">Machine - Add
Definition</a></li>
            <li><a ui-sref="mathcs.labware.add-item">Machine - Add Item</a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>


</div>

(function($){
       var cims = angular.module("cims");
             cims.controller('SideBarCtrl',['$scope', '$rootScope',
function($scope,$rootScope){

             }]);
})(jQuery);

<div ng-controller="DashboardCtrl">
  <div class="row col-lg-11">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h3 class="panel-title">Dashboard</h3>
```

```
      </div>
      <div class="panel-body">
        <div class="alert alert-info" ui-restricted-
view="lab_attendant,lab_faculty_coordinator,supply_officer">
          <span class="glyphicon glyphicon-bullhorn" style="margin-right:
5px;"></span> <strong>Notifications</strong>
        </div>
        <br>
        <div id="notificationView" ui-restricted-
view="lab_attendant,lab_faculty_coordinator,supply_officer">
          <ul id="myTab" class="nav nav-tabs nav-justified">
            <li class="active" ui-restricted-
view="lab_attendant,lab_faculty_coordinator"><a
              href="#alertTab" data-toggle="tab"
            >Alerts<span class="badge pull-
right">{{requestNotifications[RequestStatus.ALERT]}}</span></a></li>
            <li ui-restricted-view="supply_officer,lab_faculty_coordinator"><a
              href="#pendingRequestTab" data-toggle="tab"
            >Pending Request<span class="badge pull-
right">{{requestNotifications[RequestStatus.UNDER_REQUEST]}}</span></a></li>
            <li ui-restricted-view="supply_officer,lab_faculty_coordinator"><a
href="#approvedRequestTab"
              data-toggle="tab"
            >Approved<span class="badge pull-
right">{{requestNotifications[RequestStatus.APPROVED]}}</span></a></li>
            <li ui-restricted-view="supply_officer,lab_faculty_coordinator"><a
href="#rejectedRequestTab"
              data-toggle="tab"
            >Rejected<span class="badge pull-
right">{{requestNotifications[RequestStatus.REJECTED]}}</span></a></li>
          </ul>
          <div id="myTabContent" class="tab-content">
            <div class="tab-pane fade in active" id="alertTab">
              <div ng-include src="templates.alertTable"></div>
            </div>
            <div class="tab-pane fade" id="pendingRequestTab">
              <div ng-include src="templates.pendingRequestTable"></div>
            </div>
            <div class="tab-pane fade" id="approvedRequestTab">
              <div ng-include src="templates.approvedRequestTable"></div>
            </div>
            <div class="tab-pane fade" id="rejectedRequestTab">
              <div ng-include src="templates.rejectedRequestTable"></div>
            </div>
          </div>
        </div>
        <br>
        <hr>
        <div id="accountabilityView" ui-restricted-view="lab_attendant">
          <h5>
            <strong>Accountability Transactions</strong>
          </h5>
          <br>
```

```
          <div class="bs-example bs-example-tabs">
            <ul id="myTab" class="nav nav-tabs">
              <li><a href="#todayAccountability" data-toggle="tab">Today</a></li>
              <li><a href="#searchAccountability" data-toggle="tab">Search</a></li>
              <li><a href="#reportAccountability" data-toggle="tab">Report</a></li>
              <li class="active"><a href="#newAccountability" data-
toggle="tab">New</a></li>

            </ul>
            <div id="myTabContent" class="tab-content">
              <div class="tab-pane fade" id="todayAccountability">
                <div ng-include src="templates.todayAccountabilityTable"></div>
              </div>
              <div class="tab-pane fade" id="searchAccountability">
                <div ng-include src="templates.searchAccountabilityTable"></div>
              </div>
              <div class="tab-pane fade" id="reportAccountability">
                <div ng-include src="templates.reportAccountability"></div>
              </div>
              <div class="tab-pane fade in active" id="newAccountability">
                <div ng-include src="templates.newAccountabilityForm"></div>
              </div>
            </div>
          </div>
        </div>

    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller('DashboardCtrl', function($scope,
$rootScope, $storage, $http, $window, $location) {

            $.extend($scope, {
                  newAccountability : {},
                  RequestStatus : {
                        ALERT : 1,
                        UNDER_REQUEST : 2,
                        APPROVED : 3,
                        REJECTED : 4,
                        CLEAR : 5
                  },
                  ItemStatus : {
                        REPLENISHMENT : 2,
                        CONDEMNATION : 3,
                        REPAIR : 4
                  },
                  statusName : function(statusId) {
                        switch (statusId) {
                        case 2:
                              return "Replenishment";
```

```
                        case 3:
                                return "Condemnation";
                        case 4:
                        default:
                                return "Repair";
                        }
                },
                getLabModules : function() {
                        $http.post("modules/list", {}).success(function(data) {
                                $scope.modules = data;
                                $scope.newAccountability.moduleId =
$scope.modules[0].id;
                        });
                },
                MODULES : {
                        CHEMISTRY_CHEMICAL : 3,
                        CHEMISTRY_CHEMICAL_ITEM : 4,
                        CHEMISTRY_LABWARE : 5,
                        CHEMISTRY_LABWARE_ITEM : 6,
                        PHYSICS_CHEMICAL : 7,
                        PHYSICS_CHEMICAL_ITEM : 8,
                        PHYSICS_LABWARE : 9,
                        PHYSICS_LABWARE_ITEM : 10,
                        MATHCS_MACHINE : 11,
                        MATHCS_MACHINE_ITEM : 12,
                        BIOLOGY_CHEMICAL : 13,
                        BIOLOGY_CHEMICAL_ITEM : 14,
                        BIOLOGY_LABWARE : 15,
                        BIOLOGY_LABWARE_ITEM : 16,
                },
                asName : function(moduleId) {
                        switch (moduleId) {
                        case 3:
                                return "Chemistry Chemical";
                        case 4:
                                return "Chemistry Chemical Item";
                        case 5:
                                return "Chemistry Labware";
                        case 6:
                                return "Chemistry Labware Item";
                        case 7:
                                return "Physics Chemical";
                        case 8:
                                return "Physics Chemical Item";
                        case 9:
                                return "Physics Labware";
                        case 10:
                                return "Physics Labware Item";
                        case 11:
                                return "Math CS Machine";
                        case 12:
                                return "Math CS Machine Item";
                        case 13:
```

```
                        return "Biology Chemical";
                    case 14:
                        return "Biology Chemical Item";
                    case 15:
                        return "Biology Labware";
                    case 16:
                        return "Biology Labware Item";
                    }
                    return "";
                },
                requestNotifications : {},
                setRequestNotification : function(status, totalCount) {
                    if (totalCount == 0) {
                        $scope.requestNotifications[status] = null;
                        return;
                    }
                    $scope.requestNotifications[status] = totalCount;
                },
                templates : $rootScope.extendTemplates({
                    newAccountabilityForm :
"app/accountability/accountability.new.html",
                    reportAccountability :
"app/accountability/accountability.report.html",
                    searchAccountabilityTable :
"app/accountability/accountability.search.html",
                    todayAccountabilityTable :
"app/accountability/accountability.today.html",
                    alertTable : "app/maintenancerequest/alert-tpl.html",
                    pendingRequestTable : "app/maintenancerequest/pending-
request-tpl.html",
                    approvedRequestTable : "app/maintenancerequest/approved-
request-tpl.html",
                    rejectedRequestTable : "app/maintenancerequest/rejected-
request-tpl.html"
                }),
            });

            $scope.getLabModules();

        });
})(jQuery);

<div ng-controller="AlertCtrl">
  <br> <br>
  <div class="container">
    <div class="row">
      <div class="panel panel-default col-lg-12">
        <div class="panel-body">
          <form id="searchForm" class="form-horizontal" role="form" ng-
submit="alertTable.reload()"
            ui-model="searchRequestForm" ui-validate
          >
            <div class="form-group">
```

```
              <div class="col-lg-1">
                <label for="itemStatus">Item Status</label>
              </div>
              <div class="col-lg-6">
                <select  class="form-control" ng-
model="searchRequestForm.itemStatus">
                  <option ng-repeat="status in ItemStatus"
value="{{status}}">{{statusName(status)}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="moduleId">Module</label>
              </div>
              <div class="col-lg-6">
                <select  class="form-control" ng-model="searchRequestForm.moduleId">
                  <option ng-repeat="module in modules"
value="{{module.id}}">{{module.name}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="itemName">Definition Name</label>
              </div>
              <div class="col-lg-6">
                <input  class="form-control" type="text" id="itemName"
name="itemName" ng-model="searchRequestForm.itemName">
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-7">
                <button type="submit" class="btn btn-primary">Search</button>
                <button type="button" class="btn btn-primary" ng-
click="searchRequestForm = {}">Reset</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>

    <div class="table-responsive col-lg-12">
      <table ng-table="alertTable" class="table  table-bordered">
        <tbody>
          <tr ng-repeat="alert in $data">
            <td data-title="'Serial Number'">{{alert.serialNumber}}</td>
            <td data-title="'Module'"
sortable="'moduleId'">{{asName(alert.moduleId)}}</td>
            <td data-title="'Item Name'"><a
              ng-if="alert.moduleId == MODULES.CHEMISTRY_CHEMICAL || alert.moduleId
== MODULES.CHEMISTRY_CHEMICAL_ITEM"
```

```
              ui-sref="chemistry.chemical.view-
definition({chemicalId:alert.definitionId})"
            >{{alert.itemName}}</a> <a
              ng-if="alert.moduleId == MODULES.BIOLOGY_CHEMICAL || alert.moduleId ==
MODULES.BIOLOGY_CHEMICAL_ITEM"
              ui-sref="biology.chemical.view-
definition({chemicalId:alert.definitionId})"
            >{{alert.itemName}}</a> <a
              ng-if="alert.moduleId == MODULES.PHYSICS_CHEMICAL || alert.moduleId ==
MODULES.PHYSICS_CHEMICAL_ITEM"
              ui-sref="physics.chemical.view-
definition({chemicalId:alert.definitionId})"
            >{{alert.itemName}}</a> <a
              ng-if="alert.moduleId == MODULES.CHEMISTRY_LABWARE || alert.moduleId ==
MODULES.CHEMISTRY_LABWARE_ITEM"
              ui-sref="chemistry.labware.view-
definition({labwareId:alert.definitionId})"
            >{{alert.itemName}}</a> <a
              ng-if="alert.moduleId == MODULES.BIOLOGY_LABWARE || alert.moduleId ==
MODULES.BIOLOGY_LABWARE_ITEM"
              ui-sref="biology.labware.view-
definition({labwareId:alert.definitionId})"
            >{{alert.itemName}}</a> <a
              ng-if="alert.moduleId == MODULES.PHYSICS_LABWARE || alert.moduleId ==
MODULES.PHYSICS_LABWARE_ITEM"
              ui-sref="physics.labware.view-
definition({labwareId:alert.definitionId})"
            >{{alert.itemName}}</a> <a
              ng-if="alert.moduleId == MODULES.MATHCS_MACHINE || alert.moduleId ==
MODULES.MATHCS_MACHINE_ITEM"
              ui-sref="mathcs.labware.view-
definition({labwareId:alert.definitionId})"
            >{{alert.itemName}}</a></td>
            <td data-title="'Message'" sortable="'itemStatus'"><span
              ng-if="alert.itemStatus == ItemStatus.REPLENISHMENT &&
alert.requestStatus == RequestStatus.ALERT"
            ><span class="label label-danger large" style="font-size: small; margin-
right: 5px"><span
                  class="glyphicon glyphicon-exclamation-sign"
                ></span></span>Minimum reached!</span> <span
              ng-if="alert.itemStatus == ItemStatus.REPLENISHMENT &&
alert.requestStatus == RequestStatus.UNDER_REQUEST"
            > <span class="label label-warning large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-flag"
                ></span>
              </span>Pending replenishment request
            </span> <span ng-if="alert.itemStatus == ItemStatus.REPLENISHMENT &&
alert.requestStatus == RequestStatus.APPROVED">
                <span class="label label-success large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-check"
                ></span>
```

```
              </span>Approved replenishment request
            </span> <span ng-if="alert.itemStatus == ItemStatus.REPAIR &&
alert.requestStatus == RequestStatus.ALERT"><span
                class="label label-danger large" style="font-size: small; margin-
right: 5px"
              ><span class="glyphicon glyphicon-exclamation-sign"></span></span>For
Repair!</span> <span
              ng-if="alert.itemStatus == ItemStatus.REPAIR && alert.requestStatus ==
RequestStatus.UNDER_REQUEST"
            > <span class="label label-warning large" style="font-size: small;
margin-right: 5px"> <span
                class="glyphicon glyphicon-flag"
              ></span>
              </span>Pending repair request
            </span> <span ng-if="alert.itemStatus == ItemStatus.REPAIR &&
alert.requestStatus == RequestStatus.APPROVED">
                <span class="label label-success large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-check"
                ></span>
              </span>Approved repair request
            </span> <span ng-if="alert.itemStatus == ItemStatus.CONDEMNATION &&
alert.requestStatus == RequestStatus.ALERT"><span
                class="label label-danger large" style="font-size: small; margin-
right: 5px"
              ><span class="glyphicon glyphicon-exclamation-sign"></span></span>For
condemnation!</span> <span
              ng-if="alert.itemStatus == ItemStatus.CONDEMNATION &&
alert.requestStatus == RequestStatus.UNDER_REQUEST"
            > <span class="label label-warning large" style="font-size: small;
margin-right: 5px"> <span
                class="glyphicon glyphicon-flag"
              ></span>
              </span>Pending condemnation request
            </span> <span ng-if="alert.itemStatus == ItemStatus.CONDEMNATION &&
alert.requestStatus == RequestStatus.APPROVED">
                <span class="label label-success large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-check"
                ></span>
              </span>Approved condemnation request
            </span></td>
            <td ui-restricted-view="system_admin,lab_faculty_coordinator" data-
title="'Action'"><a
              ng-if="alert.itemStatus == ItemStatus.REPLENISHMENT &&
alert.requestStatus == RequestStatus.ALERT"
              class="clickable" ng-click="sendReplenishmentRequest(alert)"
            > Request for replenishment </a> <a
              ng-if="alert.itemStatus == ItemStatus.REPAIR && alert.requestStatus ==
RequestStatus.ALERT"
              class="clickable" ng-click="sendRepairRequest(alert)"
            > Request for repair </a> <a
```

```
              ng-if="alert.itemStatus == ItemStatus.CONDEMNATION &&
alert.requestStatus == RequestStatus.ALERT"
              class="clickable" ng-click="sendCondemnationRequest(alert)"
            > Request for condemnation </a></td>
          </tr>
          <tr ng-if="!$data || !$data.length">
            <td colspan="5"><div class="text-center">
                <strong>No Result Found</strong>
              </div></td>
          </tr>
        </tbody>
        <tfoot>
          <tr>
            <td>Page: {{alertTable.page()}}</td>
          </tr>
        </tfoot>
      </table>
    </div>
  </div>

(function($) {
      angular.module("cims").controller('AlertCtrl', function($scope, $rootScope,
$http, ngTableParams,$filter) {

            $.extend($scope, {
                  searchRequestForm : {},
                  getTotalCount : function() {
                        $http.post("maintenance-request/total",
{requestStatus:$scope.RequestStatus.ALERT}).success(function(data) {

      $scope.setRequestNotification($scope.RequestStatus.ALERT, data.totalCount);
                        });
                  },
                  sendReplenishmentRequest : function(item){
                        $http.post("maintenance-request/replenishment-
request/create", {id:item.id})
                          .success(function(data) {
                                alert(data.message);
                                item.requestStatus =
$scope.RequestStatus.UNDER_REQUEST;
                                $rootScope.$broadcast("change-maintenance-request-
status",{previousRequestStatus : $scope.RequestStatus.ALERT});
                        });
                  },
                  sendRepairRequest : function(item){
                        $http.post("maintenance-request/repair-request/create",
{id:item.id})
                          .success(function(data) {
                                alert(data.message);
                                item.requestStatus =
$scope.RequestStatus.UNDER_REQUEST;
                                $rootScope.$broadcast("change-maintenance-request-
status",{previousRequestStatus : $scope.RequestStatus.ALERT});
```

```
                        });
                },
                sendCondemnationRequest : function(item){
                        $http.post("maintenance-request/condemnation-
request/create", {id:item.id})
                          .success(function(data) {
                                alert(data.message);
                                item.requestStatus =
$scope.RequestStatus.UNDER_REQUEST;
                                $rootScope.$broadcast("change-maintenance-request-
status",{previousRequestStatus : $scope.RequestStatus.ALERT});
                        });
                },
        });

        $scope.getTotalCount();

        $scope.alertTable = new ngTableParams({
                page : 1,
                count : 10,
                sorting : {
                        moduleId : 'asc'
                }
        }, {
                total : 0,
                getData : function($defer, params) {
                        $http.post("maintenance-request/alert/list",
$scope.searchRequestForm).success(function(data) {
                                var orderedData = params.sorting() ?
         $filter('orderBy')(data, params.orderBy()) :
         data;

                                params.total(data.length);
                                $defer.resolve(orderedData.slice((params.page() - 1)
* params.count(), params.page() * params.count()));
                        });
                }
        });

        $scope.$on("change-maintenance-request-status",function(event,data){
                if(data.previousRequestStatus != $scope.RequestStatus.ALERT){
                        $scope.getTotalCount();
                        $scope.alertTable.reload();
                }
        });

    });
})(jQuery);

<div ng-controller="ApprovedRequestCtrl">

  <br> <br>
  <div class="container">
```

```
    <div class="row">
      <div class="panel panel-default col-lg-12">
        <div class="panel-body">
          <form id="searchForm" class="form-horizontal" role="form" ng-
submit="approvedRequestTable.reload()"
            ui-model="searchRequestForm" ui-validate
          >
            <div class="form-group">
              <div class="col-lg-1">
                <label for="itemStatus">Item Status</label>
              </div>
              <div class="col-lg-6">
                <select class="form-control" ng-model="searchRequestForm.itemStatus">
                  <option ng-repeat="status in ItemStatus"
value="{{status}}">{{statusName(status)}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="moduleId">Module</label>
              </div>
              <div class="col-lg-6">
                <select class="form-control" ng-model="searchRequestForm.moduleId">
                  <option ng-repeat="module in modules"
value="{{module.id}}">{{module.name}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="itemName">Definition Name</label>
              </div>
              <div class="col-lg-6">
                <input class="form-control" type="text" id="itemName" name="itemName"
                  ng-model="searchRequestForm.itemName"
                >
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-7">
                <button type="submit" class="btn btn-primary">Search</button>
                <button type="button" class="btn btn-primary" ng-
click="searchRequestForm = {}">Reset</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>

    <div class="table-responsive col-lg-12">
      <table ng-table="approvedRequestTable" class="table  table-bordered">
```

```
        <tbody>
          <tr ng-repeat="request in $data">
            <td data-title="'Serial Number'"><span
              ng-if="request.itemStatus == ItemStatus.REPAIR || request.itemStatus ==
ItemStatus.CONDEMNATION"
            >{{request.serialNumber}}</span></td>
            <td data-title="'Module'"
sortable="'moduleId'">{{asName(request.moduleId)}}</td>
            <td data-title="'Item Name'"><a
              ng-if="request.moduleId == MODULES.CHEMISTRY_CHEMICAL ||
request.moduleId == MODULES.CHEMISTRY_CHEMICAL_ITEM"
              ui-sref="chemistry.chemical.view-
definition({chemicalId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.BIOLOGY_CHEMICAL || request.moduleId
== MODULES.BIOLOGY_CHEMICAL_ITEM"
              ui-sref="biology.chemical.view-
definition({chemicalId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.PHYSICS_CHEMICAL || request.moduleId
== MODULES.PHYSICS_CHEMICAL_ITEM"
              ui-sref="physics.chemical.view-
definition({chemicalId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.CHEMISTRY_LABWARE ||
request.moduleId == MODULES.CHEMISTRY_LABWARE_ITEM"
              ui-sref="chemistry.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.BIOLOGY_LABWARE || request.moduleId
== MODULES.BIOLOGY_LABWARE_ITEM"
              ui-sref="biology.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.PHYSICS_LABWARE || request.moduleId
== MODULES.PHYSICS_LABWARE_ITEM"
              ui-sref="physics.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.MATHCS_MACHINE || request.moduleId
== MODULES.MATHCS_MACHINE_ITEM"
              ui-sref="mathcs.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a></td>
            <td data-title="'Message'" sortable="'itemStatus'"><span
              ng-if="request.itemStatus == ItemStatus.REPLENISHMENT"
            > <span class="label label-success large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-check"
                ></span>
              </span> Approved replenishment request
            </span> <span ng-if="request.itemStatus == ItemStatus.CONDEMNATION">
<span class="label label-success large"
```

```
                style="font-size: small; margin-right: 5px"
               > <span class="glyphicon glyphicon-check"></span></span> Approved
condemnation request
             </span> <span ng-if="request.itemStatus == ItemStatus.REPAIR"> <span
class="label label-success large"
                style="font-size: small; margin-right: 5px"
               > <span class="glyphicon glyphicon-check"></span></span> Approved
repair request
             </span></td>
           </tr>
           <tr ng-if="!$data || !$data.length">
             <td colspan="4"><div class="text-center">
                 <strong>No Result Found</strong>
               </div></td>
           </tr>
         </tbody>
         <tfoot>
           <tr>
             <td>Page: {{approvedRequestTable.page()}}</td>
           </tr>
         </tfoot>
       </table>
     </div>
   </div>


(function($) {
       angular.module("cims").controller('ApprovedRequestCtrl', function($scope,
$rootScope, $http, ngTableParams,$filter) {

              $.extend($scope, {
                     searchRequestForm : {},
                     getTotalCount : function() {
                            $http.post("maintenance-request/total",
{requestStatus:$scope.RequestStatus.APPROVED}).success(function(data) {

       $scope.setRequestNotification($scope.RequestStatus.APPROVED, data.totalCount);
                            });
                     }

              });

              $scope.getTotalCount();

              $scope.approvedRequestTable = new ngTableParams({
                     page : 1,
                     count : 10,
                     sorting : {
                            moduleId : 'asc'
                     }
              }, {
                     total : 0,
                     getData : function($defer, params) {
```

```
                        $http.post("maintenance-request/list",
$.extend({requestStatus
:$scope.RequestStatus.APPROVED},$scope.searchRequestForm)).success(function(data) {
                                var orderedData = params.sorting() ?
             $filter('orderBy')(data, params.orderBy()) :
             data;
                                params.total(data.length);
                                $defer.resolve( orderedData.slice((params.page() -
1) * params.count(), params.page() * params.count()));
                        });
                  }
             });

             $scope.$on("change-maintenance-request-status",function(event,data){
                  if(data.previousRequestStatus != $scope.RequestStatus.APPROVED){
                        $scope.getTotalCount();
                        $scope.approvedRequestTable.reload();
                  }
             });

       });
})(jQuery);

<div ng-controller="PendingRequestCtrl">
  <br> <br>
  <div class="container">
    <div class="row">
      <div class="panel panel-default col-lg-12">
        <div class="panel-body">
          <form id="searchForm" class="form-horizontal" role="form" ng-
submit="pendingRequestTable.reload()"
            ui-model="searchRequestForm" ui-validate
          >
            <div class="form-group">
              <div class="col-lg-1">
                <label for="itemStatus">Item Status</label>
              </div>
              <div class="col-lg-6">
                <select class="form-control" ng-model="searchRequestForm.itemStatus">
                  <option ng-repeat="status in ItemStatus"
value="{{status}}">{{statusName(status)}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="moduleId">Module</label>
              </div>
              <div class="col-lg-6">
                <select class="form-control" ng-model="searchRequestForm.moduleId">
                  <option ng-repeat="module in modules"
value="{{module.id}}">{{module.name}}</option>
                </select>
```

```
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-1">
              <label for="itemName">Definition Name</label>
            </div>
            <div class="col-lg-6">
              <input class="form-control" type="text" id="itemName" name="itemName"
                ng-model="searchRequestForm.itemName"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
              <button type="button" class="btn btn-primary" ng-
click="searchRequestForm = {}">Reset</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-12">
    <table ng-table="pendingRequestTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="request in $data">
          <td data-title="'Serial Number'">{{request.serialNumber}}</td>
          <td data-title="'Module'"
sortable="'moduleId'">{{asName(request.moduleId)}}</td>
          <td data-title="'Item Name'"><a
            ng-if="request.moduleId == MODULES.CHEMISTRY_CHEMICAL ||
request.moduleId == MODULES.CHEMISTRY_CHEMICAL_ITEM"
            ui-sref="chemistry.chemical.view-
definition({chemicalId:request.definitionId})"
          >{{request.itemName}}</a> <a
            ng-if="request.moduleId == MODULES.BIOLOGY_CHEMICAL || request.moduleId
== MODULES.BIOLOGY_CHEMICAL_ITEM"
            ui-sref="biology.chemical.view-
definition({chemicalId:request.definitionId})"
          >{{request.itemName}}</a> <a
            ng-if="request.moduleId == MODULES.PHYSICS_CHEMICAL || request.moduleId
== MODULES.PHYSICS_CHEMICAL_ITEM"
            ui-sref="physics.chemical.view-
definition({chemicalId:request.definitionId})"
          >{{request.itemName}}</a> <a
            ng-if="request.moduleId == MODULES.CHEMISTRY_LABWARE ||
request.moduleId == MODULES.CHEMISTRY_LABWARE_ITEM"
            ui-sref="chemistry.labware.view-
definition({labwareId:request.definitionId})"
          >{{request.itemName}}</a> <a
            ng-if="request.moduleId == MODULES.BIOLOGY_LABWARE || request.moduleId
== MODULES.BIOLOGY_LABWARE_ITEM"
```

```
              ui-sref="biology.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.PHYSICS_LABWARE || request.moduleId
== MODULES.PHYSICS_LABWARE_ITEM"
              ui-sref="physics.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.MATHCS_MACHINE || request.moduleId
== MODULES.MATHCS_MACHINE_ITEM"
              ui-sref="mathcs.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a></td>

            <td data-title="'Message'" sortable="'itemStatus'"><span
              ng-if="request.itemStatus == ItemStatus.REPLENISHMENT &&
request.requestStatus == RequestStatus.UNDER_REQUEST"
            > <span class="label label-warning large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-flag"
                ></span></span> Pending replenishment request
            </span> <span
              ng-if="request.itemStatus == ItemStatus.REPAIR && request.requestStatus
== RequestStatus.UNDER_REQUEST"
            > <span class="label label-warning large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-flag"
                ></span></span> Pending repair request
            </span> <span
              ng-if="request.itemStatus == ItemStatus.CONDEMNATION &&
request.requestStatus == RequestStatus.UNDER_REQUEST"
            > <span class="label label-warning large" style="font-size: small;
margin-right: 5px"> <span
                  class="glyphicon glyphicon-flag"
                ></span></span> Pending condemnation request
            </span> <span ng-if="request.requestStatus == RequestStatus.APPROVED">
<span
              class="label label-success large" style="font-size: small; margin-
right: 5px"
            > <span class="glyphicon glyphicon-check"></span></span>Approved
request
            </span> <span ng-if="request.requestStatus == RequestStatus.REJECTED">
<span
              class="label label-danger large" style="font-size: small; margin-
right: 5px"
            > <span class="glyphicon glyphicon-thumbs-down"></span></span>Rejected
request
            </span></td>
            <td data-title="'Action'" ui-restricted-
view="system_admin,supply_officer"><span
              ng-if="request.requestStatus == RequestStatus.UNDER_REQUEST"
            > <a class="clickable" ng-
click="sendApproval(request,RequestStatus.APPROVED)"> Approve </a> | <a
```

```
                class="clickable" ng-
click="sendApproval(request,RequestStatus.REJECTED)"
                > Reject </a>
              </span></td>
            </tr>
            <tr ng-if="!$data || !$data.length">
              <td colspan="5"><div class="text-center">
                  <strong>No Result Found</strong>
                </div></td>
            </tr>
          </tbody>
          <tfoot>
            <tr>
              <td>Page: {{pendingRequestTable.page()}}</td>
            </tr>
          </tfoot>
        </table>
      </div>
    </div>

(function($) {
       angular.module("cims").controller('PendingRequestCtrl', function($scope,
$rootScope, $http, ngTableParams,$filter) {

              $.extend($scope, {
                     searchRequestForm : {},
                     getTotalCount : function() {
                           $http.post("maintenance-request/total",
{requestStatus:$scope.RequestStatus.UNDER_REQUEST}).success(function(data) {

       $scope.setRequestNotification($scope.RequestStatus.UNDER_REQUEST,
data.totalCount);
                           });
                     },
                     sendApproval : function(requestItem, approvalStatus) {
                           var confirmAction = true;
                           if (approvalStatus == $scope.RequestStatus.REJECTED) {
                                 confirmAction = confirm("Are you sure you want to
reject this request?");
                           }

                           if (confirmAction) {
                                 $http.post("maintenance-request/pending-
request/approval", {
                                       id : requestItem.id,
                                       requestStatus : approvalStatus
                                 }).success(function(data) {
                                       alert(data.message);
                                       $scope.getTotalCount();
                                       //requestItem.requestStatus = approvalStatus;
                                       $scope.getTotalCount();
                                       $scope.pendingRequestTable.reload();
```

```
                                    $rootScope.$broadcast("change-maintenance-
request-status",{previousRequestStatus : $scope.RequestStatus.UNDER_REQUEST});
                                });
                        }

                }

        });

        $scope.getTotalCount();

        $scope.pendingRequestTable = new ngTableParams({
                page : 1,
                count : 10,
                sorting : {
                        moduleId : 'asc'
                }
        }, {
                total : 0,
                getData : function($defer, params) {
                        $http.post("maintenance-request/list",
$.extend({requestStatus
:$scope.RequestStatus.UNDER_REQUEST},$scope.searchRequestForm)).success(function(data
) {
                                var orderedData = params.sorting() ?
             $filter('orderBy')(data, params.orderBy()) :
             data;
                                params.total(data.length);
                                $defer.resolve(orderedData.slice((params.page() - 1)
* params.count(), params.page() * params.count()));
                        });
                }
        });

        $scope.$on("change-maintenance-request-status",function(event,data){
                if(data.previousRequestStatus !=
$scope.RequestStatus.UNDER_REQUEST){
                        $scope.getTotalCount();
                        $scope.pendingRequestTable.reload();
                }
        });

    });
})(jQuery);

<div ng-controller="RejectedRequestCtrl">

  <br> <br>
  <div class="container">
    <div class="row">
      <div class="panel panel-default col-lg-12">
        <div class="panel-body">
```

```
          <form id="searchForm" class="form-horizontal" role="form" ng-
submit="rejectedRequestTable.reload()"
            ui-model="searchRequestForm" ui-validate
          >
            <div class="form-group">
              <div class="col-lg-1">
                <label for="itemStatus">Item Status</label>
              </div>
              <div class="col-lg-6">
                <select class="form-control" ng-model="searchRequestForm.itemStatus">
                  <option ng-repeat="status in ItemStatus"
value="{{status}}">{{statusName(status)}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="moduleId">Module</label>
              </div>
              <div class="col-lg-6">
                <select class="form-control" ng-model="searchRequestForm.moduleId">
                  <option ng-repeat="module in modules"
value="{{module.id}}">{{module.name}}</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-1">
                <label for="itemName">Definition Name</label>
              </div>
              <div class="col-lg-6">
                <input class="form-control" type="text" id="itemName" name="itemName"
                  ng-model="searchRequestForm.itemName"
                >
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-7">
                <button type="submit" class="btn btn-primary">Search</button>
                <button type="button" class="btn btn-primary" ng-
click="searchRequestForm = {}">Reset</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>

    <div class="table-responsive col-lg-12">
      <table ng-table="rejectedRequestTable" class="table  table-bordered">
        <tbody>
          <tr ng-repeat="request in $data">
            <td data-title="'Serial Number'"><span
```

```
              ng-if="request.itemStatus == ItemStatus.REPAIR || request.itemStatus ==
ItemStatus.CONDEMNATION"
            >{{request.serialNumber}}</span></td>
            <td data-title="'Module'"
sortable="'moduleId'">{{asName(request.moduleId)}}</td>
            <td data-title="'Item Name'"><a
              ng-if="request.moduleId == MODULES.CHEMISTRY_CHEMICAL ||
request.moduleId == MODULES.CHEMISTRY_CHEMICAL_ITEM"
              ui-sref="chemistry.chemical.view-
definition({chemicalId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.BIOLOGY_CHEMICAL || request.moduleId
== MODULES.BIOLOGY_CHEMICAL_ITEM"
              ui-sref="biology.chemical.view-
definition({chemicalId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.PHYSICS_CHEMICAL || request.moduleId
== MODULES.PHYSICS_CHEMICAL_ITEM"
              ui-sref="physics.chemical.view-
definition({chemicalId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.CHEMISTRY_LABWARE ||
request.moduleId == MODULES.CHEMISTRY_LABWARE_ITEM"
              ui-sref="chemistry.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.BIOLOGY_LABWARE || request.moduleId
== MODULES.BIOLOGY_LABWARE_ITEM"
              ui-sref="biology.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.PHYSICS_LABWARE || request.moduleId
== MODULES.PHYSICS_LABWARE_ITEM"
              ui-sref="physics.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a> <a
              ng-if="request.moduleId == MODULES.MATHCS_MACHINE || request.moduleId
== MODULES.MATHCS_MACHINE_ITEM"
              ui-sref="mathcs.labware.view-
definition({labwareId:request.definitionId})"
            >{{request.itemName}}</a></td>
            <td data-title="'Message'" sortable="'itemStatus'"><span
              ng-if="request.itemStatus == ItemStatus.REPLENISHMENT"
            > <span class="label label-danger large" style="font-size: small; margin-
right: 5px"> <span
                  class="glyphicon glyphicon-thumbs-down"
                ></span></span> Rejected replenishment request
            </span> <span ng-if="request.itemStatus == ItemStatus.CONDEMNATION">
<span class="label label-danger large"
              style="font-size: small; margin-right: 5px"
            > <span class="glyphicon glyphicon-thumbs-down"></span></span> Rejected
condemnation request
```

```
            </span> <span ng-if="request.itemStatus == ItemStatus.REPAIR"> <span
class="label label-danger large"
                style="font-size: small; margin-right: 5px"
              > <span class="glyphicon glyphicon-thumbs-down"></span></span> Rejected
repair request
            </span></td>
          </tr>
          <tr ng-if="!$data || !$data.length">
            <td colspan="4"><div class="text-center">
                <strong>No Result Found</strong>
              </div></td>
          </tr>
        </tbody>
        <tfoot>
          <tr>
            <td>Page: {{rejectedRequestTable.page()}}</td>
          </tr>
        </tfoot>
      </table>
    </div>
  </div>

(function($) {
      angular.module("cims").controller('RejectedRequestCtrl', function($scope,
$rootScope, $http, ngTableParams,$filter) {

            $.extend($scope, {
                  searchRequestForm : {},
                  getTotalCount : function() {
                        $http.post("maintenance-request/total",
{requestStatus:$scope.RequestStatus.REJECTED}).success(function(data) {

      $scope.setRequestNotification($scope.RequestStatus.REJECTED, data.totalCount);
                        });
                  }

            });

            $scope.getTotalCount();

            $scope.rejectedRequestTable = new ngTableParams({
                  page : 1,
                  count : 10,
                  sorting : {
                        moduleId : 'asc'
                  }
            }, {
                  total : 0,
                  getData : function($defer, params) {
                        $http.post("maintenance-request/list",
$.extend({requestStatus
:$scope.RequestStatus.REJECTED},$scope.searchRequestForm)).success(function(data) {
                              var orderedData = params.sorting() ?
```

```
                $filter('orderBy')(data, params.orderBy()) :
                data;
                                params.total(data.length);
                                $defer.resolve(orderedData.slice((params.page() - 1)
* params.count(), params.page() * params.count()));
                        });
                }
        });

        $scope.$on("change-maintenance-request-status",function(event,data){
                if(data.previousRequestStatus != $scope.RequestStatus.REJECTED){
                        $scope.getTotalCount();
                        $scope.rejectedRequestTable.reload();
                }
        });

    });
})(jQuery);

<div ng-controller="mathcsAddLabwareDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add MathCS Labware Definition</h4>
    </div>
    <div class="panel-body">
      <form id="searchmathcsForm" class="form-horizontal" role="form" ui-validate ui-
model="definition"
        ng-submit="add()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="Name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="type">Type</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="type" ng-
model="definition.type"
              placeholder="Type"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="size">Size</label>
          </div>
```

```
          <div class="col-lg-8">
            <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("mathcsAddLabwareDefCtrl", function($scope,
$rootScope, $http) {
            $.extend($scope, {
                  init : function() {
                        $scope.definition = {
                              minimum : 0,
                              hygroscopic : true,
                              explosive : true
                        };
                  },
                  add : function() {
                        $http.post("mathcs/labware/add",
$scope.definition).success(function(data) {
                              alert(data.message);
                              if (data.success) {
                                    $scope.init();
                              }
                        });
                  }
            });

            $scope.init();

      });
```

```
})(jQuery);

<div ng-controller="mathcsAddLabwareItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Add MathCS Labware Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchmathcsForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="chemicalName"> Labware Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="labwareId" ng-
model="item.labwareId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="labwareName" name="labwareName"
                ng-model="item.labwareName" placeholder="Autocomplete Labware Name"
                typeahead="labware as labware.name for labware in
autocomplete($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-
select="item.labwareId=$model.id;item.labwareName=$model.name"
              >
            </div>
          </div>
          <input type="text" class="hide {required: true}" name="quantity" ng-
model="item.quantity"
            placeholder="Serial Number" ng-value="1"
          >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="serialNumber">Serial Number<span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control {required: true}"
name="serialNumber" ng-model="item.serialNumber"
                placeholder="Serial Number"
              >
            </div>
          </div>

          <div class="form-group">
            <div class="col-lg-12">
              <button type="submit" class="btn btn-primary">Add</button>
            </div>
          </div>
```

```
        </form>
      </div>
    </div>
  </div>
</div>

(function($){
       angular.module("cims").

controller("mathcsAddLabwareItemCtrl",function($scope,$rootScope,$http,$state){

              var labwareId = $scope.$stateParams.labwareId;

                $.extend($scope,{
                       init : function(){
                            $scope.item = {
                                    quantity : 1
                              };
                       },
                       autocomplete : function(key){
                            return $http.post("mathcs/labware/search", {name : key} )
                              .then(function(response) {
                                    return response.data;
                            });
                       },
                       addItem : function(){
                            $http.post("mathcs/labware/add-item",$scope.item)
                              .success(function(data){
                                   alert(data.message);
                                   $state.go("^.view-
definition",{labwareId:$scope.item.labwareId});
                              });
                       },
                            getLabwareName : function() {
                                  $http.post("mathcs/labware/get", {
                                         id : labwareId
                                  }).success(function(data) {
                                         $scope.item.labwareName = data.name;
                                         $scope.item.labwareId = labwareId;
                                  });
                            }
                });

                if(labwareId){
                     $scope.getLabwareName();
                }

                $scope.init();

         });
})(jQuery);
```

```
<div>
  <div ui-view></div>
</div>

<div>
  <div id="searchLabwareItemModal" class="modal fade">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
          <h3 class="modal-title">MathCS Labware Items -
{{selectedLabware.name}}</h3>
        </div>
        <div class="modal-body">
          <table ng-table="labwareItemTable" class="table  table-bordered">
            <tbody>
              <tr ng-repeat="labwareItem in $data">
                <td data-title="'Serial Number'">{{labwareItem.serialNumber}}</td>
                <td data-title="'Created Date'">{{labwareItem.createdDate |
amDateFormat:'dddd, MMMM Do YYYY,
                    h:mm:ss a'}}</td>
                <td data-title="'Action'"><a class="clickable" ng-
click="sendRepairRequest(labwareItem)"> For
                    repair</a> | <a class="clickable" ng-
click="sendCondemnationRequest(labwareItem)"> For condemnation</a></td>
              </tr>
              <tr ng-if="!$data || !$data.length">
                <td colspan="3"><div class="text-center">
                    <strong>No Result Found</strong>
                  </div></td>
              </tr>
            </tbody>
            <tfoot>
              <tr>
                <td>Page: {{labwareItemTable.page()}}</td>
              </tr>
            </tfoot>
          </table>

        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
        </div>
      </div>
    </div>
  </div>
</div>

<div ng-controller="SearchmathcsLabwareCtrl">
  <div class="col-lg-10">
```

```
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Search MathCS Labware {{count}}</h4>
      </div>
      <div class="panel-body">
        <form id="searchmathcsForm" class="form-horizontal" role="form" ng-
submit="search()">
          <div class="form-group">
            <div class="col-lg-1">
              <label for="name">Name</label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control" id="name" name="name" ng-
model="mathcsForm.name"
                placeholder="Name"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-10">
    <table ng-table="labwareTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="labware in $data">
          <td data-title="'Name'"><a class="clickable"
            ui-sref="mathcs.labware.update-definition({labwareId : labware.id})"
          >{{labware.name}}</a></td>
          <td data-title="'Available Quantity'">{{labware.totalQuantity}}</td>
          <td data-title="'Minimum Quantity'">{{labware.minimum}}</td>
          <td data-title="'Action'"><a class="clickable"
            ui-sref="mathcs.labware.update-definition({labwareId : labware.id})"
          >Update</a>
          | <a class="clickable"
            ui-sref="mathcs.labware.add-item({labwareId : labware.id})"
          > Add Quantity</a>
          | <a class="clickable" data-toggle="modal" data-
target="#searchLabwareItemModal"

          > <span ng-click="selectLabware(labware)">View Items</span></a>


          </td>
        </tr>
        <tr ng-if="!$data || !$data.length">
          <td colspan="3"><div class="text-center">
              <strong>No Result Found</strong>
```

```
            </div></td>
          </tr>
        </tbody>
        <tfoot>
          <tr>
            <td>Page: {{labwareTable.page()}}</td>
          </tr>
        </tfoot>
      </table>
    </div>

    <div ng-include src="templates.searchLabwareItemModal">
    </div>
</div>

(function($) {
      angular.module("cims").controller("SearchmathcsLabwareCtrl", function($scope,
$rootScope, $http, ngTableParams) {
             $.extend($scope, {
                    mathcsForm : {},
                    selectedLabware : {},
                    search : function() {
                          $scope.labwareTable.reload();
                    },
                    selectLabware: function(labware){
                          $scope.selectedLabware = labware;
                          $scope.searchItem();
                    },
                    searchItem : function(){
      $scope.labwareItemTable.reload();
      },
                    templates : $rootScope.extendTemplates({
                          searchLabwareItemModal :
"app/mathcs/labware/mathcs.labware.item.search.html"
                    }),
                    sendRepairRequest : function(labwareItem){
                          $http.post("mathcs/labware/item/repair-alert/create", {id
: labwareItem.id})
                              .success(function(data) {
                                    alert(data.message);
                          });
                    },
                    sendCondemnationRequest: function(labwareItem){
                          $http.post("mathcs/labware/item/condemnation-
alert/create", {id : labwareItem.id})
                       .success(function(data) {
                             alert(data.message);
                    });
                    }

             });

             $scope.labwareTable = new ngTableParams({
```

```
                page : 1,
                count : 10
        }, {
                total : 0,
                getData : function($defer, params) {
                        $http.post("mathcs/labware/search",
$scope.mathcsForm).success(function(data) {
                                params.total(data.length);
                                $defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
                        });
                }
        });

        $scope.labwareItemTable = new ngTableParams({
                page : 1,
                count : 10
        }, {
                total : 0,
                getData : function($defer, params) {
                        $http.post("mathcs/labware/item/search", {labwareId :
$scope.selectedLabware.id}).success(function(data) {
                                params.total(data.length);
                                $defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
                        });
                }
        });

    });
})(jQuery);

<div ng-controller="UpdatemathcsLabwareCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Update MathCS Labware</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="mathcs.labware.add-
item({labwareId : definition.id})">
          Add Quantity</a> <br> <br> <br>
        <form id="searchmathcsForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="Name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
            </div>
          </div>
```

```
          <div class="form-group">
            <div class="col-lg-2">
              <label for="type">Type</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="type" ng-
model="definition.type" placeholder="Type">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="size">Size</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="brokenQuantity">Broken Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="brokenQuantity" ng-
model="definition.brokenQuantity"
                placeholder="Broken Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="stainedQuantity">Stained Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="stainedQuantity" ng-
model="definition.stainedQuantity"
                placeholder="Stained Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
                placeholder="Minimum Quantity"
              >
            </div>
            <div class="form-group">
              <div class="col-lg-7">
                <button type="submit" class="btn btn-primary">Update</button>
```

```
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("UpdatemathcsLabwareCtrl", function($scope,
$rootScope, $http) {

            var labwareId = $scope.$stateParams.labwareId;

            $.extend($scope, {
                  definition : {},
                  get : function() {
                        $http.post("mathcs/labware/get", {
                              id : labwareId
                        }).success(function(data) {
                              $scope.definition = data;
                        });
                  },
                  update : function() {
                        $http.post("mathcs/labware/update-definition",
$scope.definition).success(function(data) {
                              alert(data.message);
                        });
                  }
            });

            $scope.get();
      });
})(jQuery);

<div ng-controller="ViewMathCSLabwareDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">MathCS Labware - {{definition.name}}</h4>
      </div>
      <div class="panel-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-lg-2">
              <label>Name</label>
            </div>
            <div class="col-lg-8">{{definition.name}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
```

```
            <label>Available Quantity</label>
          </div>
          <div class="col-lg-8">{{definition.totalQuantity}}</div>
        </div>
        <br>
        <div class="row">
          <div class="col-lg-2">
            <label>Minimum Quantity</label>
          </div>
          <div class="col-lg-8">{{definition.minimum}}</div>
        </div>
        <br>          <br>
        <div class="row">
          <div class="col-lg-2">
            <label>Type</label>
          </div>
          <div class="col-lg-8">{{definition.type}}</div>
        </div>
        <br>
        <div class="row">
          <div class="col-lg-2">
            <label>Size</label>
          </div>
          <div class="col-lg-8">{{definition.size}}</div>
        </div>
        <br>
        <div class="row">
          <div class="col-lg-2">
            <label>Updated Date</label>
          </div>
          <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
        </div>
      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("ViewMathCSLabwareDefCtrl",
                  function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
                        var definitionId = $scope.$stateParams.labwareId;

                        $.extend($scope, {
                              definition : {},
                              get : function() {
                                    $http.post("mathcs/labware/get", {id :
definitionId})
                                      .success(function(data) {
                                        $scope.definition  = data;
                                    });
                              }
```

```
                        });

                        $scope.get();

                });
})(jQuery);

(function($) {
        $.extend(true, window.routeConfig, {
                states : {
                        'mathcs' : {
                                url : '/mathcs',
                                templateUrl : 'app/mathcs/mathcs.html'
                        },
                        'mathcs.labware' : {
                                url : '/labware',
                                templateUrl : 'app/mathcs/labware/mathcs.labware.html'
                        },
                        'mathcs.labware.add-definition' : {
                                url : '/add-definition',
                                templateUrl : 'app/mathcs/labware/mathcs.labware.add-
definition.html'
                        },
                        'mathcs.labware.search' : {
                                url : '/search',
                                templateUrl :
'app/mathcs/labware/mathcs.labware.search.html'
                        },
                        'mathcs.labware.update-definition' : {
                                url : '/update-definition?labwareId',
                                templateUrl : 'app/mathcs/labware/mathcs.labware.update-
definition.html'
                        },
                        'mathcs.labware.add-item' : {
                                url : '/add-item?labwareId',
                                templateUrl : 'app/mathcs/labware/mathcs.labware.add-
item.html'
                        },
                        'mathcs.labware.view-definition' : {
                                url : '/view-definition?labwareId',
                                templateUrl : 'app/mathcs/labware/mathcs.labware.view-
definition.html'
                        },
                        'mathcs.labware.view-definition' : {
                                url : '/view-definition?labwareId',
                                templateUrl : 'app/mathcs/labware/mathcs.labware.view-
definition.html'
                        },

                }
        });
})(jQuery);
```

```
<div>
  <div ui-view></div>
</div>

<div ng-controller="physicsAddChemicalDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add physics Definition</h4>
    </div>
    <div class="panel-body">
      <form id="searchphysicsForm" class="form-horizontal" role="form" ui-validate
ui-model="definition"
        ng-submit="add()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Name">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="molFormula">Molecular Formula</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="molFormula" ng-
model="definition.molFormula"
              placeholder="Molecular Formula"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="color">Color</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="color" ng-
model="definition.color" placeholder="Color">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="texture">Texture</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="texture" ng-
model="definition.texture" placeholder="Texture">
          </div>
        </div>
        <div class="form-group">
```

```
          <div class="col-lg-2">
            <label for="grade">Grade</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="grade" ng-
model="definition.grade" placeholder="Grade">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="hygroscopic">Hygroscopic</label>
          </div>
          <div class="col-lg-8">
            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.hygroscopic == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.hygroscopic"
checked="checked"
                name="hygroscopic" class="ng-pristine ng-valid"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.hygroscopic == false)}"> <input
                type="radio" ng-model="definition.hygroscopic" ng-value="false"
name="hygroscopic"
                class="ng-pristine ng-valid"
              > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="explosive">Explosive</label>
          </div>
          <div class="col-lg-8">

            <span class="btn-group"> <label class="btn btn-primary active"
              ng-class="{active:(definition.explosive == true)}"
            > <input type="radio" ng-value="true" ng-model="definition.explosive"
checked="checked" name="explosive"
                class="ng-pristine ng-valid"
              > Yes
            </label> <label class="btn btn-primary" ng-
class="{active:(definition.explosive == false)}"> <input
                type="radio" ng-model="definition.explosive" ng-value="false"
name="explosive"
                class="ng-pristine ng-valid"
              > No
            </label>
            </span>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
```

```
            <label for="thresh">Thresh</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="thresh" ng-
model="definition.thresh" placeholder="Thresh">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("physicsAddChemicalDefCtrl",
function($scope, $rootScope, $http) {
            $.extend($scope, {
                  init : function() {
                        $scope.definition = {
                              minimum : 0,
                              hygroscopic : true,
                              explosive : true
                        };
                  },
                  add : function() {
                        $http.post("physics/chemical/add",
$scope.definition).success(function(data) {
                              alert(data.message);
                              if (data.success) {
                                    $scope.init();
                              }
                        });
                  }
            });

            $scope.init();
```

```
      });
})(jQuery);

<div ng-controller="physicsAddItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title" ng-click="alert('');">Add physics Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchphysicsForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
            <div class="col-lg-3">
              <label for="chemicalName"> Chemical Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="chemicalId" ng-
model="item.chemicalId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="chemicalName" name="chemicalName"
                ng-model="item.chemicalName" placeholder="Autocomplete Chemical Name"
                typeahead="chemical as chemical.name for chemical in
autocomplete($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-
select="item.chemicalId=$model.id;item.chemicalName=$model.name"
              >
            </div>
          </div>

          <div class="form-group">
            <div class="col-lg-3">
              <label for="expDate"> Expiration Date</label>
            </div>
            <div class="col-lg-6">
              <div class="input-group">
                <input type="text" class="form-control" ng-model="item.expDate"
placeholder="Expiration Date"
                  datepicker-popup="dd-MMMM-yyyy" is-open="expDateOpen"
                /> <span class="input-group-addon btn btn-default" ng-
click="$event.preventDefault();$event.stopPropagation();expDateOpen=true;"
                  ng-init="expDateOpen=false"
                > <i class="glyphicon glyphicon-calendar"></i>
                </span>

              </div>
            </div>
          </div>
          <div class="form-group">
```

```
            <div class="col-lg-3">
              <label for="quantity">Quantity<span class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control {required: true}"
name="quantity" ng-model="item.quantity"
                placeholder=""
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-12">
              <button type="submit" class="btn btn-primary">Add</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("physicsAddItemCtrl", function($scope,
$rootScope, $http,$state) {

            var chemicalId = $scope.$stateParams.chemicalId;

            $.extend($scope, {
                  init : function() {
                        $scope.item = {
                              quantity : 1
                        };
                  },
                  autocomplete : function(key) {
                        return $http.post("physics/chemical/search", {
                              name : key
                        }).then(function(response) {
                              return response.data;
                        });
                  },
                  addItem : function() {
                        $http.post("physics/chemical/add-item",
$scope.item).success(function(data) {
                              alert(data.message);
                              $state.go("^.view-definition",{chemicalId :
$scope.item.chemicalId});
                        });
                  },
                  getChemicalName : function() {
                        $http.post("physics/chemical/get", {
                              id : chemicalId
                        }).success(function(data) {
                              $scope.item.chemicalName = data.name;
```

```
                                $scope.item.chemicalId = chemicalId;
                        });
                }

        });

        if (chemicalId) {
                $scope.getChemicalName();
        }

        $scope.init();

    });
})(jQuery);

<div>
  <div ui-view></div>
</div>

<div ng-controller="SearchphysicsChemicalCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Search physics</h4>
      </div>
      <div class="panel-body">
        <form id="searchphysicsForm" class="form-horizontal" role="form" ng-
submit="search()">
          <div class="form-group">
            <div class="col-lg-1">
              <label for="name">Name</label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control" id="name" name="name" ng-
model="physicsForm.name"
                placeholder="Name"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-10">
    <table ng-table="chemicalTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="chemical in $data">
```

```
          <td data-title="'Name'"><a class="clickable"
            ui-sref="physics.chemical.update-definition({chemicalId : chemical.id})"
          >{{chemical.name}}</a></td>
          <td data-title="'Available Quantity'">{{chemical.totalQuantity}}</td>
          <td data-title="'Minimum Quantity'">{{chemical.minimum}}</td>
          <td data-title="'Action'"><a class="clickable"
            ui-sref="physics.chemical.update-definition({chemicalId : chemical.id})"
          >Update</a> | <a class="clickable"
            ui-sref="physics.chemical.add-item({chemicalId : chemical.id})"
          > Add Quantity</a></td>
        </tr>
        <tr ng-if="!$data || !$data.length">
          <td colspan="3"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Page: {{chemicalTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>
</div>

(function($) {
      angular.module("cims").controller("SearchphysicsChemicalCtrl",
                  function($scope, $rootScope, $http, ngTableParams) {
                        $.extend($scope, {
                              physicsForm : {},
                              search : function() {
                                    $scope.chemicalTable.reload();
                              }

                        });

                        $scope.chemicalTable = new ngTableParams({
                              page : 1,
                              count : 10
                        }, {
                              total : 0,
                              getData : function($defer, params) {
                                    $http.post("physics/chemical/search",
$scope.physicsForm).success(function(data) {
                                          params.total(data.length);

      $defer.resolve(data.slice((params.page() - 1) * params.count(), params.page()
                                                      * params.count()));
                                    });
                              }
                        });
```

```
                    });
})(jQuery);

<div ng-controller="UpdatephysicsChemicalCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Update physics Chemical</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="physics.chemical.add-
item({chemicalId : definition.id})"> Add Quantity</a>
        <br>
        <br>
        <br>
        <form id="searchphysicsForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Name">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="molFormula">Molecular Formula</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="molFormula" ng-
model="definition.molFormula"
                placeholder="Molecular Formula"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="color">Color</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="color" ng-
model="definition.color" placeholder="Color">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="texture">Texture</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="texture" ng-
model="definition.texture" placeholder="Texture">
```

```
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="grade">Grade</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="grade" ng-
model="definition.grade" placeholder="Grade">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="hygroscopic">Hygroscopic</label>
            </div>
            <div class="col-lg-8">
              <span class="btn-group"> <label class="btn btn-primary active"
                ng-class="{active:(definition.hygroscopic == true)}"
              > <input type="radio" ng-value="true" ng-model="definition.hygroscopic"
checked="checked"
                  name="hygroscopic" class="ng-pristine ng-valid" value="true"
                > Yes
              </label> <label class="btn btn-primary" ng-
class="{active:(definition.hygroscopic == false)}"> <input
                  type="radio" ng-model="definition.hygroscopic" ng-value="false"
name="hygroscopic"
                  class="ng-pristine ng-valid" value="false"
                > No
              </label>
              </span>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="explosive">Explosive</label>
            </div>
            <div class="col-lg-8">

              <span class="btn-group"> <label class="btn btn-primary active"
                ng-class="{active:(definition.explosive == true)}"
              > <input type="radio" ng-value="true" ng-model="definition.explosive"
checked="checked"
                  name="explosive" class="ng-pristine ng-valid" value="true"
                > Yes
              </label> <label class="btn btn-primary" ng-
class="{active:(definition.explosive == false)}"> <input
                  type="radio" ng-model="definition.explosive" ng-value="false"
name="explosive"
                  class="ng-pristine ng-valid" value="false"
                > No
              </label>
              </span>
            </div>
```

```
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="thresh">Thresh</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="thresh" ng-
model="definition.thresh" placeholder="Thresh">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum" placeholder="Minimum Quantity">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Save</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("UpdatephysicsChemicalCtrl",
function($scope, $rootScope, $http) {

            var chemicalId = $scope.$stateParams.chemicalId;

            $.extend($scope, {
                  definition : {},
                  get : function() {
                        $http.post("physics/chemical/get", {
                              id : chemicalId
                        }).success(function(data) {
                              $scope.definition = data;
                        });
                  },
                  update : function() {
                        $http.post("physics/chemical/update-definition",
$scope.definition).success(function(data) {
                              alert(data.message);
                        });
                  }
            });
```

```
            $scope.get();
        });
})(jQuery);

<div ng-controller="ViewPhysicsChemicalDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Physics Chemical - {{definition.name}}</h4>
      </div>
      <div class="panel-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-lg-2">
              <label>Name</label>
            </div>
            <div class="col-lg-8">{{definition.name}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Available Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.totalQuantity}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Minimum Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.minimum}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Molecular Formula</label>
            </div>
            <div class="col-lg-8">{{definition.molFormula}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Color</label>
            </div>
            <div class="col-lg-8">{{definition.color}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Texture</label>
            </div>
            <div class="col-lg-8">{{definition.texture}}</div>
          </div>
```

```
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Grade</label>
            </div>
            <div class="col-lg-8">{{definition.grade}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Hygroscopic</label>
            </div>
            <div class="col-lg-8">{{definition.hygroscopic}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Explosive</label>
            </div>
            <div class="col-lg-8">{{definition.explosive}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Updated Date</label>
            </div>
            <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("ViewPhysicsChemicalDefCtrl",
                  function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
                        var definitionId = $scope.$stateParams.chemicalId;

                        $.extend($scope, {
                              definition : {},
                              get : function() {
                                    $http.post("physics/chemical/get", {id :
definitionId})
                                      .success(function(data) {
                                        $scope.definition  = data;
                                    });
                              }

                        });

                        $scope.get();
```

```
                    });
})(jQuery);

<div ng-controller="physicsAddLabwareDefCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add physics Labware Definition</h4>
    </div>
    <div class="panel-body">
      <form id="searchphysicsForm" class="form-horizontal" role="form" ui-validate
ui-model="definition"
        ng-submit="add()"
      >
        <div class="form-group">
          <div class="col-lg-2">
            <label for="Name">Name<span class="required">*</span></label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="type">Type</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="type" ng-
model="definition.type"
              placeholder="Type"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="size">Size</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="minimum">Minimum</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
              placeholder="Minimum Quantity"
            >
          </div>
```

```
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="button" class="btn btn-primary" ng-
click="init()">Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("physicsAddLabwareDefCtrl", function($scope,
$rootScope, $http) {
            $.extend($scope, {
                  init : function() {
                        $scope.definition = {
                              minimum : 0,
                              hygroscopic : true,
                              explosive : true
                        };
                  },
                  add : function() {
                        $http.post("physics/labware/add",
$scope.definition).success(function(data) {
                              alert(data.message);
                              if (data.success) {
                                    $scope.init();
                              }
                        });
                  }
            });

            $scope.init();

      });
})(jQuery);

<div ng-controller="physicsAddLabwareItemCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Add physics Labware Item</h4>
      </div>
      <div class="panel-body">
        <form id="searchphysicsForm" class="form-horizontal" role="form" ng-
submit="addItem()" ui-validate
          ui-model="item"
        >
          <div class="form-group">
```

```
            <div class="col-lg-3">
              <label for="chemicalName"> Labware Name <span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="hide" name="labwareId" ng-
model="item.labwareId" /> <input type="text"
                autocomplete="off" class="form-control {required: true}"
id="labwareName" name="labwareName"
                ng-model="item.labwareName" placeholder="Autocomplete Labware Name"
                typeahead="labware as labware.name for labware in
autocomplete($viewValue)  | limitTo:8"
                typeahead-wait-ms="500" typeahead-on-
select="item.labwareId=$model.id;item.labwareName=$model.name"
              >
            </div>
          </div>

          <input type="text" class="hide {required: true}" name="quantity" ng-
model="item.quantity"
            placeholder="Serial Number" ng-value="1"
          >

          <div class="form-group">
            <div class="col-lg-3">
              <label for="serialNumber">Serial Number<span
class="required">*</span></label>
            </div>
            <div class="col-lg-6">
              <input type="text" class="form-control {required: true}"
name="serialNumber" ng-model="item.serialNumber"
                placeholder="Serial Number"
              >
            </div>
          </div>

          <div class="form-group">
            <div class="col-lg-12">
              <button type="submit" class="btn btn-primary">Add</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>


(function($){
       angular.module("cims").

controller("physicsAddLabwareItemCtrl",function($scope,$rootScope,$http,$state){

             var labwareId = $scope.$stateParams.labwareId;
```

```
                $.extend($scope,{
                        init : function(){
                             $scope.item = {
                                     quantity : 1
                               };
                        },
                        autocomplete : function(key){
                             return $http.post("physics/labware/search", {name : key} )
                               .then(function(response) {
                                     return response.data;
                             });
                        },
                        addItem : function(){
                             $http.post("physics/labware/add-item",$scope.item)
                               .success(function(data){
                                    alert(data.message);
                                    $state.go("^.view-
definition",{labwareId:$scope.item.labwareId});
                               });
                        },
                             getLabwareName : function() {
                                    $http.post("physics/labware/get", {
                                           id : labwareId
                                    }).success(function(data) {
                                           $scope.item.labwareName = data.name;
                                           $scope.item.labwareId = labwareId;
                                    });
                             }
                });

                if(labwareId){
                     $scope.getLabwareName();
                }

                $scope.init();

        });
})(jQuery);

<div>
  <div ui-view></div>
</div>

<div>
  <div id="searchLabwareItemModal" class="modal fade">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
```

```
          <h3 class="modal-title">physics Labware Items -
{{selectedLabware.name}}</h3>
        </div>
        <div class="modal-body">
          <table ng-table="labwareItemTable" class="table  table-bordered">
            <tbody>
              <tr ng-repeat="labwareItem in $data">
                <td data-title="'Serial Number'">{{labwareItem.serialNumber}}</td>
                <td data-title="'Created Date'">{{labwareItem.createdDate |
amDateFormat:'dddd, MMMM Do YYYY, h:mm:ss a'}}</td>
                <td data-title="'Action'"><a class="clickable" ng-
click="sendRepairRequest(labwareItem)"> For
                    repair</a> | <a class="clickable" ng-
click="sendCondemnationRequest(labwareItem)"> For condemnation</a></td>
              </tr>
              <tr ng-if="!$data || !$data.length">
                <td colspan="3"><div class="text-center">
                    <strong>No Result Found</strong>
                  </div></td>
              </tr>
            </tbody>
            <tfoot>
              <tr>
                <td>Page: {{labwareItemTable.page()}}</td>
              </tr>
            </tfoot>
          </table>

        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
        </div>
      </div>
    </div>
  </div>
</div>

<div ng-controller="SearchphysicsLabwareCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Search physics Labware {{count}}</h4>
      </div>
      <div class="panel-body">
        <form id="searchphysicsForm" class="form-horizontal" role="form" ng-
submit="search()">
          <div class="form-group">
            <div class="col-lg-1">
              <label for="name">Name</label>
            </div>
            <div class="col-lg-6">
```

```
              <input type="text" class="form-control" id="name" name="name" ng-
model="physicsForm.name"
                placeholder="Name"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-7">
              <button type="submit" class="btn btn-primary">Search</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
  <div class="table-responsive col-lg-10">
    <table ng-table="labwareTable" class="table  table-bordered">
      <tbody>
        <tr ng-repeat="labware in $data">
          <td data-title="'Name'"><a class="clickable"
            ui-sref="physics.labware.update-definition({labwareId : labware.id})"
          >{{labware.name}}</a></td>
          <td data-title="'Available Quantity'">{{labware.totalQuantity}}</td>
          <td data-title="'Minimum Quantity'">{{labware.minimum}}</td>
          <td data-title="'Action'"><a class="clickable"
            ui-sref="physics.labware.update-definition({labwareId : labware.id})"
          >Update</a>
          | <a class="clickable"
            ui-sref="physics.labware.add-item({labwareId : labware.id})"
          > Add Quantity</a>
          | <a class="clickable" data-toggle="modal" data-
target="#searchLabwareItemModal"

          > <span ng-click="selectLabware(labware)">View Items</span></a>


          </td>
        </tr>
        <tr ng-if="!$data || !$data.length">
          <td colspan="3"><div class="text-center">
              <strong>No Result Found</strong>
            </div></td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Page: {{labwareTable.page()}}</td>
        </tr>
      </tfoot>
    </table>
  </div>

  <div ng-include src="templates.searchLabwareItemModal">
```

```
  </div>
</div>

(function($) {
	angular.module("cims").controller("SearchphysicsLabwareCtrl", function($scope,
$rootScope, $http, ngTableParams) {
		$.extend($scope, {
			physicsForm : {},
			selectedLabware : {},
			search : function() {
				$scope.labwareTable.reload();
			},
			selectLabware: function(labware){
				$scope.selectedLabware = labware;
				$scope.searchItem();
			},
			searchItem : function(){
	$scope.labwareItemTable.reload();
	},
			templates : $rootScope.extendTemplates({
				searchLabwareItemModal :
"app/physics/labware/physics.labware.item.search.html"
			}),
			sendRepairRequest : function(labwareItem){
				$http.post("physics/labware/item/repair-alert/create", {id
: labwareItem.id})
				   .success(function(data) {
					alert(data.message);
				});
			},
			sendCondemnationRequest: function(labwareItem){
				$http.post("physics/labware/item/condemnation-
alert/create", {id : labwareItem.id})
				   .success(function(data) {
					alert(data.message);
			});
			}

		});

		$scope.labwareTable = new ngTableParams({
			page : 1,
			count : 10
		}, {
			total : 0,
			getData : function($defer, params) {
				$http.post("physics/labware/search",
$scope.physicsForm).success(function(data) {
					params.total(data.length);
					$defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
				});
			}
```

```
		});

		$scope.labwareItemTable = new ngTableParams({
			page : 1,
			count : 10
		}, {
			total : 0,
			getData : function($defer, params) {
				$http.post("physics/labware/item/search", {labwareId :
$scope.selectedLabware.id}).success(function(data) {
					params.total(data.length);
					$defer.resolve(data.slice((params.page() - 1) *
params.count(), params.page() * params.count()));
					});
			}
		});

	});
})(jQuery);

<div ng-controller="UpdatePhysicsLabwareCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Update physics Labware</h4>
      </div>
      <div class="panel-body">
        <a class="btn btn-small btn-primary" ui-sref="physics.labware.add-
item({labwareId : definition.id})">
          Add Quantity</a> <br> <br> <br>
        <form id="searchphysicsForm" class="form-horizontal" role="form" ng-
submit="update()">
          <div class="form-group">
            <div class="col-lg-2">
              <label for="Name">Name<span class="required">*</span></label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="name" ng-
model="definition.name" placeholder="Description">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="type">Type</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="type" ng-
model="definition.type" placeholder="Type">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="size">Size</label>
```

```
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="size" ng-
model="definition.size" placeholder="Size">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="brokenQuantity">Broken Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="brokenQuantity" ng-
model="definition.brokenQuantity"
                placeholder="Broken Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="stainedQuantity">Stained Quantity</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="stainedQuantity" ng-
model="definition.stainedQuantity"
                placeholder="Stained Quantity"
              >
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-2">
              <label for="minimum">Minimum</label>
            </div>
            <div class="col-lg-8">
              <input type="text" class="form-control" name="minimum" ng-
model="definition.minimum"
                placeholder="Minimum Quantity"
              >
            </div>
            <div class="form-group">
              <div class="col-lg-7">
                <button type="submit" class="btn btn-primary">Update</button>
              </div>
            </div>
        </form>
      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller("UpdatePhysicsLabwareCtrl", function($scope,
$rootScope, $http) {
```

```
		var labwareId = $scope.$stateParams.labwareId;

		$.extend($scope, {
			definition : {},
			get : function() {
				$http.post("physics/labware/get", {
					id : labwareId
				}).success(function(data) {
					$scope.definition = data;
				});
			},
			update : function() {
				$http.post("physics/labware/update-definition",
$scope.definition).success(function(data) {
					alert(data.message);
				});
			}
		});

		$scope.get();
	});
})(jQuery);
<div ng-controller="ViewPhysicsLabwareDefCtrl">
  <div class="col-lg-10">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">Physics Labware - {{definition.name}}</h4>
      </div>
      <div class="panel-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-lg-2">
              <label>Name</label>
            </div>
            <div class="col-lg-8">{{definition.name}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Available Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.totalQuantity}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Minimum Quantity</label>
            </div>
            <div class="col-lg-8">{{definition.minimum}}</div>
          </div>
          <br>          <br>
          <div class="row">
            <div class="col-lg-2">
```

```
              <label>Type</label>
            </div>
            <div class="col-lg-8">{{definition.type}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Size</label>
            </div>
            <div class="col-lg-8">{{definition.size}}</div>
          </div>
          <br>
          <div class="row">
            <div class="col-lg-2">
              <label>Updated Date</label>
            </div>
            <div class="col-lg-8">{{definition.updatedDate | date: 'medium'}}</div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

(function($) {
	angular.module("cims").controller("ViewPhysicsLabwareDefCtrl",
			function($scope, $rootScope, $http, ngTableParams,
$state,$stateParams) {
				var definitionId = $scope.$stateParams.labwareId;

				$.extend($scope, {
					definition : {},
					get : function() {
						$http.post("physics/labware/get", {id :
definitionId})
						  .success(function(data) {
						    $scope.definition  = data;
						});
					}

				});

				$scope.get();

			});
})(jQuery);

(function($) {
	$.extend(true, window.routeConfig, {
		states : {
			'physics' : {
				url : '/physics',
```

```
                templateUrl : 'app/physics/physics.html'
            },
            'physics.chemical' : {
                url : '/chemical',
                templateUrl : 'app/physics/chemical/physics.chemical.html'
            },
            'physics.chemical.search' : {
                url : '/search',
                templateUrl :
'app/physics/chemical/physics.chemical.search.html'
            },
            'physics.chemical.add-definition' : {
                url : '/add-definition',
                templateUrl : 'app/physics/chemical/physics.chemical.add-
definition.html'
            },
            'physics.chemical.update-definition' : {
                url : '/update-definition?chemicalId',
                templateUrl :
'app/physics/chemical/physics.chemical.update-definition.html'
            },
            'physics.chemical.add-item' : {
                url : '/add-item?chemicalId',
                templateUrl : 'app/physics/chemical/physics.chemical.add-
item.html'
            },
            'physics.chemical.view-definition' : {
                url : '/view-definition?chemicalId',
                templateUrl : 'app/physics/chemical/physics.chemical.view-
definition.html'
            },
            'physics.labware' : {
                url : '/labware',
                templateUrl : 'app/physics/labware/physics.labware.html'
            },
            'physics.labware.add-definition' : {
                url : '/add-definition',
                templateUrl : 'app/physics/labware/physics.labware.add-
definition.html'
            },
            'physics.labware.search' : {
                url : '/search',
                templateUrl :
'app/physics/labware/physics.labware.search.html'
            },
            'physics.labware.update-definition' : {
                url : '/update-definition?labwareId',
                templateUrl : 'app/physics/labware/physics.labware.update-
definition.html'
            },
            'physics.labware.add-item' : {
                url : '/add-item?labwareId',
```

```
                        templateUrl : 'app/physics/labware/physics.labware.add-
item.html'
                  },
                  'physics.labware.view-definition' : {
                        url : '/view-definition?labwareId',
                        templateUrl : 'app/physics/labware/physics.labware.view-
definition.html'
                  },

            }
      });
})(jQuery);

<div>
  <div ui-view></div>
</div>

<div ng-controller="AddRoleCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add Role</h4>
    </div>
    <div class="panel-body">
      <form id="addUserForm" name="userForm" novalidate class="form-horizontal" ng-
submit="addRole()">
        <div ng-include src="templates.roleForm"></div>
        <div class="form-group">
          <div class="col-lg-12">
            <input type="submit" class="btn btn-primary" value="Add" />
          </div>
        </div>
      </form>
    </div>
  </div>
</div>

(function($){
      angular.module("cims").
        controller("AddRoleCtrl",function($scope,$rootScope,$http){
              $.extend($scope,{
              });
        });
})(jQuery);

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
```

```
</body>
</html>

<div ng-controller="RoleCtrl">
  <div ui-view></div>
</div>

(function($){
	angular.module("cims").
	  controller("RoleCtrl",function($scope,$rootScope,$http){
		$.extend($scope,{
			templates: {
				roleForm: "app/role/roleForm.tpl.html"
			},
			getUserTypeList : function(){

	$http.get("http://localhost/CIMS/getUserTypeList").success(function(data,
status, headers, config){
					console.log(data);
					$scope.userTypeList = data;
				}).error(function(data, status, headers, config){
					console.log("error");
				});
			}
		});
		$scope.getUserTypeList();
	  });
})(jQuery);

(function($) {
	angular.module("cims",
			[
				'ngRoute',
				'ngSanitize',
				'ngAnimate',
				'ngCookies',
				'ngTouch',
				'angular-fix',
				'ui.bootstrap',
				'angularLocalStorage',
				'ui.router',
				'services.breadcrumbs',
				'ngTable',
				'loading-notif',
				'chieffancypants.loadingBar',
				'angular-validation',
				'angular-authorization',
				'ngAutoFill',
				'angularMoment',
				'angular-restricted-view',
				'onEnter'
```

```
                  ]).config(function($stateProvider, $routeProvider,
$urlRouterProvider, $locationProvider, $logProvider) {

            $logProvider.debugEnabled(false);

            $locationProvider.html5Mode(false).hashPrefix("!");

            var routes = window.routeConfig;
            $urlRouterProvider.otherwise(routes.otherwise);
            // FIXME possible incompabitibilty with old browsers when iterating
            // object properties must include an Object polyfill
            for ( var key in routes.states) {
                  var routeItem = routes.states[key];
                  $stateProvider.state(key, {
                        url : routeItem.url,
                        templateUrl : routeItem.templateUrl
                  });
            }

      }).run(function($rootScope, ngValidator, $state, $stateParams) {
            $rootScope.$state = $state;
            $rootScope.$stateParams = $stateParams;

      });
})(jQuery);

<div>
<div class="jumbotron">
  <div class="container">
    <h1>Welcome to CIMS</h1>
    <p>This web application is a thesis project intended to automate and
     modernize chemical inventory for UP Manila for more secured, efficient process
and
     centralized accessibility</p>
    <p><a class="btn btn-primary btn-lg">Learn more</a></p>
  </div>
</div>

</div>

<div ng-controller="LoginCtrl">
  <div class="row col-lg-8">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h3 class="panel-title">Login</h3>
      </div>
      <div class="panel-body">

        <form id="loginForm" class="form-horizontal" role="form" ng-
submit="doLogin()" ui-validate ui-model="login">
          <div class="form-group">
            <div class="col-lg-12">
```

```
              <input type="text" class="form-control" name="username" ng-
model="login.username" placeholder="Username">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-12">
              <input type="password" class="form-control" name="password" ng-
model="login.password"
                placeholder="Password" ng-autofill>
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-12">
              <button type="submit" class="btn btn-primary">Login</button>
            </div>
          </div>
        </form>

      </div>
    </div>
  </div>
</div>

(function($) {
      angular.module("cims").controller('LoginCtrl', function($scope, $rootScope,
$storage, $http, $window, $location,$state) {

            $rootScope.loginPage = true;

            $storage.bind($scope, 'login.username');
            $storage.bind($scope, 'login.password');

            $.extend($scope, {
                  doLogin : function() {

                        var login = $scope.login ? $scope.login : {
                              username : ''
                        };
                        $http.post("login", login).success(function(data, status,
headers, config) {
                              if (data.success) {
                                    $rootScope.isLoggedIn = true;
                                    $location.path("/dashboard");
                                    $rootScope.user = data.user;
                                    $scope.getUserRole();
                              } else {
                                    alert(data.message);
                              }
                        }).error(function(data, status, headers, config) {

                        });
                  }
            });
```

```
	});
})(jQuery);

(function($) {
  window.routeConfig = {
		states : {
			'login': {
				url : '/login',
				templateUrl : 'app/settings/login.html'
			},
			'index': {
				url : '/index',
				templateUrl : 'app/settings/start.html'
			},
			'dashboard': {
				url : '/dashboard',
				templateUrl : 'app/dashboard/dashboard.html'
			}
		},
		otherwise : "/dashboard"
  };
})(jQuery);

(function($) {
	$.extend(true, window.routeConfig, {
		states : {
			'user' : {
				url : '/user',
				templateUrl : 'app/useraccount/user.html'
			},
			'user.add' : {
				url : '/add',
				templateUrl : 'app/useraccount/user.add.html'
			},
			'user.update' : {
				url : '/update?userId',
				templateUrl : 'app/useraccount/user.update.html'
			},
			'user.search' : {
				url : '/search',
				templateUrl : 'app/useraccount/user.search.html'
			},
			'role' : {
				url : '/role',
				templateUrl : 'app/role/role.html'
			},
			'role.add' : {
				url : '/add',
				templateUrl : 'app/role/role.add.html'
			},
			'role.update' : {
```

```
                        url : '/update',
                        templateUrl : 'app/role/role.update.html'
                    }

            }
        });
})(jQuery);

<div ng-controller="AddUserCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Add User</h4>
    </div>
    <div class="panel-body">
      <form id="addUserForm" novalidate="novalidate" class="form-horizontal" ng-
submit="addUser()" ui-validate
        ui-model="userForm"
      >
        <!--
        <div class="form-group">
          <div class="col-lg-2">
            <label for="username">Username <span class="required">*</span>
            </label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="username"
              ng-model="userForm.username" placeholder="Username"
            >
          </div>
        </div>
         -->
        <div class="form-group">
          <div class="col-lg-2">
            <label for="employeeId">Employee Id <span
class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="employeeId"
              ng-model="userForm.employeeId" placeholder="Employee Id"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="firstname">Firstname <span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="firstname"
              ng-model="userForm.firstname" placeholder="Firstname"
            >
```

```
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="lastname">Lastname <span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="lastname"
              ng-model="userForm.lastname" placeholder="Lastname"
            >
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="password">Password <span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="password" class="form-control input-sm {required : true}"
id="password" name="password"
              ng-model="userForm.password" placeholder="Password"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="confirmedPassword">Confirmed Password <span
class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="password" class="form-control input-sm {required : true ,
equalTo: '#password'}"
              name="confirmedPassword" id="confirmedPassword"
name="confirmedPassword"
              ng-model="userForm.confirmedPassword" placeholder="Confirmed Password"
            >
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="departmentId">Unit <span class="required">*</span></label>
          </div>
          <div class="col-lg-4">
            <select class="form-control input-sm {required:true}" ng-
model="userForm.departmentId"
              ng-options="department.id as department.name for department in
departmentList" id="departmentId"
              name="departmentId"
            >
            </select>
          </div>
```

```
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="roleId">Primary Role <span class="required">*</span></label>
          </div>
          <div class="col-lg-4">
            <select class="form-control input-sm {required:true}" ng-
model="userForm.roleId" id="roleId" name="roleId"
            >
            <option ng-repeat="roleItem in roleList"
value="{{roleItem.id}}">{{roleItem.name}}</option>
            </select>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="laboratoryUnitId">Laboratory Unit<span
class="required">*</span></label>
          </div>
          <div class="col-lg-4">
            <select class="form-control input-sm {required:true}" ng-
model="userForm.laboratoryUnitId"
              id="laboratoryUnitId" name="laboratoryUnitId"
              ng-disabled="!userForm.roleId || (userForm.roleId != 3 &&
userForm.roleId != 4)"
            >
              <option ng-repeat="laboratoryUnit in laboratoryUnits"
value="{{laboratoryUnit.id}}">{{laboratoryUnit.name}}</option>
            </select>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Add</button>
            <button type="reset" class="btn btn-primary" ng-click="userForm = {}">
              <i class="glyphicon glyphicon-refresh"></i>Reset
            </button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>

(function($){
       angular.module("cims").
         controller("AddUserCtrl",function($scope,$rootScope,$http,$log,$state){
               $.extend($scope,{
                     userForm : {},

                     addUser : function(){
```

```
                        $log.log("submitting form = " + $scope.userForm.form);

$http.post("addUser",$scope.userForm).success(function(data, status, headers,
config){
                                    alert(data.message);
                                    if(data.success){
                                      $scope.userForm = {};
                                    }
                                    $state.go("user.search");

                                }).error(function(data, status, headers, config){

                                      console.log("error");
                                });
                        },

                        getDepartments: function(){
                            $http.post("department/list",{}).success(function(data,
status, headers, config){
                                    $scope.departmentList = data;
                                });
                        },

                        getLaboratoryUnits: function(){
                            $http.post("laboratory-unit/list",{})
                              .success(function(data,status, headers, config){
                                   $scope.laboratoryUnits = data;
                              });
                        }
                });

                $scope.getDepartments();

                $scope.getLaboratoryUnits();

                $scope.$watch("userForm.roleId",function(){
                    if(!$scope.userForm.roleId || ($scope.userForm.roleId != 3 &&
$scope.userForm.roleId != 4)){
                      $scope.userForm.laboratoryUnitId = null;
                    }
                });

        });
})(jQuery);

<div ng-controller="UserCtrl">
  <div ui-view></div>
</div>

(function($){
       angular.module("cims").
         controller("UserCtrl",function($scope,$rootScope,$http){
```

```
                $.extend($scope,{
                        templates: {
                                userForm: "app/useraccount/userForm.tpl.html"
                        },
                        getRoleList : function(){
                             $http.get("role/list").success(function(data, status,
headers, config){
                                        $scope.roleList = data;
                             }).error(function(data, status, headers, config){
                                        console.log("error");
                             });
                        }
                });
                $scope.getRoleList();
        });
})(jQuery);

<div ng-controller="SearchUserCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Search User</h4>
    </div>
    <div class="panel-body">
      <form id="searchUserForm" class="form-horizontal" role="form" ng-
submit="searchUser()">
        <div class="form-group">
          <div class="col-lg-2">
            <label for="username">Username</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control input-sm" name="username" ng-
model="searchForm.username"
              placeholder="Username"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="employeeId">Employee Id</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control input-sm" name="employeeId" ng-
model="searchForm.employeeId"
              placeholder="Employee Id"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="firstname">Firstname</label>
          </div>
          <div class="col-lg-8">
```

```
            <input type="text" class="form-control input-sm" name="firstname" ng-
model="searchForm.firstname"
              placeholder="Firstname"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="lastname">Lastname</label>
          </div>
          <div class="col-lg-8">
            <input type="text" class="form-control input-sm" name="lastname" ng-
model="searchForm.lastname"
              placeholder="Lastname"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="roleId">User Type</label>
          </div>
          <div class="col-lg-8">
            <select ng-model="searchForm.roleId" name="roleId" ng-options="role.id as
role.name for role in roleList">
            </select>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Search</button>
            <button type="button" class="btn btn-primary" ng-click="searchForm =
{}"><i class="glyphicon glyphicon-refresh"></i>Reset</button>
          </div>
        </div>
      </form>
    </div>
  </div>

  <div>
    <a ui-sref="user.add" class="btn btn-primary btn-sm pull-right"> <span
class="glyphicon glyphicon-plus-sign"> </span>
      Add User
    </a>
  </div>
  <div class="clearfix"></div>
  <br>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Users</h4>
    </div>
    <div class="panel-body table-responsive">
      <table ng-table="userTable" class="table  table-bordered">
        <tbody>
```

```
          <tr ng-repeat="user in $data">
            <td data-title="'Name'"><a class="clickable" ui-
sref="user.update({userId:user.id})">{{user.firstname}} {{user.lastname}}</a></td>
            <td data-title="'Role'">{{user.role.name}}</td>
            <td data-title="'Action'"><a class="clickable" ui-
sref="user.update({userId:user.id})">Update</a></td>
          </tr>
          <tr ng-if="!$data || !$data.length">
            <td colspan="3"><div class="text-center"><strong>No Result
Found</strong></div></td>
          </tr>
        </tbody>
        <tfoot>
          <tr>
            <td>Page: {{userTable.page()}}</td>
          </tr>
        </tfoot>
      </table>
    </div>
  </div>
</div>

(function($){
       angular.module("cims").
         controller("SearchUserCtrl",function($scope,$rootScope,$http,ngTableParams){
               $.extend($scope,{
                    users : [],
                      searchForm : {},
                      searchUser : function(){

$http.post("user/list",$scope.searchForm).success(function(data, status, headers,
config){
                                  $scope.users = data;
                                  $scope.userTable.reload();
                                  }).error(function(data, status, headers, config){
                                        console.log("error");
                                  });
                      },
                      createUserTable : function(){
                           $scope.userTable =  new ngTableParams({
            page: 1,
            count: 10
        }, {
            total: 0,
            getData: function($defer, params) {
              params.total($scope.users.length);
               $defer.resolve($scope.users.slice((params.page() - 1) * params.count(),
params.page() * params.count()));
            }
        });
                      }
               });
               $scope.createUserTable();
```

```
            $scope.searchUser();
        });
})(jQuery);

<div ng-controller="UpdateUserCtrl">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">Update User</h4>
    </div>
    <div class="panel-body">
      <form class="form-horizontal" role="form" ng-submit="updateUser()">
        <input type="text" class="hide" name="id" ng-model="user.id" />
        <div class="form-group">
          <div class="col-lg-2">
            <label for="username">Username </label>
          </div>
          <div class="col-lg-6">{{user.username}}</div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="employeeId">Employee Id <span
class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="employeeId"
              ng-model="user.employeeId" placeholder="Employee Id"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="firstname">Firstname <span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="firstname"
              ng-model="user.firstname" placeholder="Firstname"
            >
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="lastname">Lastname <span class="required">*</span></label>
          </div>
          <div class="col-lg-6">
            <input type="text" class="form-control input-sm {required : true}"
name="lastname" ng-model="user.lastname"
              placeholder="Lastname"
            >
          </div>
        </div>
        <div class="form-group">
```

```
          <div class="col-lg-2">
            <label for="departmentId">Unit <span class="required">*</span></label>
          </div>
          <div class="col-lg-4">
            <select class="form-control input-sm {required:true}" ng-
model="user.departmentId"
              ng-options="department.id as department.name for department in
departmentList" id="departmentId"
              name="departmentId"
            >
            </select>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-2">
            <label for="roleId">Primary Role <span class="required">*</span></label>
          </div>
          <div class="col-lg-4">
            <select class="form-control input-sm {required:true}" ng-
model="user.roleId" id="roleId" name="roleId">
              <option ng-repeat="roleItem in roleList"
value="{{roleItem.id}}">{{roleItem.name}}</option>
            </select>
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-2">
            <label for="laboratoryUnitId">Laboratory Unit<span
class="required">*</span></label>
          </div>
          <div class="col-lg-4">
            <select class="form-control input-sm {required:true}" ng-
model="user.laboratoryUnitId"
              id="laboratoryUnitId" name="laboratoryUnitId"
              ng-disabled="!user.roleId || (user.roleId != 3 && user.roleId != 4)"
            >
              <option ng-repeat="laboratoryUnit in laboratoryUnits"
value="{{laboratoryUnit.id}}">{{laboratoryUnit.name}}</option>
            </select>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-12">
            <button type="submit" class="btn btn-primary">Update</button>
          </div>
        </div>

      </form>

    </div>
  </div>
```

```
</div>
(function($) {
	angular.module("cims").controller("UpdateUserCtrl", function($scope,
$rootScope, $http, $location) {

		var userId = $scope.$stateParams.userId;

		$.extend($scope, {
			user : {},
			getUserDetails : function() {
				$http.post("user/details", {
					id : userId
				}).success(function(data) {
					$scope.user = data;
				});
			},
			updateUser : function() {
				$http.post("user/admin/update",
$scope.user).success(function(data) {
					alert(data.message);
				});
			},
			getDepartments : function() {
				$http.post("department/list", {})
				  .success(function(data, status, headers, config) {
					$scope.departmentList = data;
				});
			},
		  getLaboratoryUnits: function(){
			$http.post("laboratory-unit/list",{})
			  .success(function(data,status, headers, config){
				$scope.laboratoryUnits = data;
			  });
		  }
		});

		$scope.getDepartments();

		$scope.getLaboratoryUnits();

		$scope.getUserDetails();

		$scope.$watch("user.roleId",function(){
			if(!$scope.user.roleId || ($scope.user.roleId != 3 &&
$scope.user.roleId != 4)){
			  $scope.user.laboratoryUnitId = null;
			}
		});


	});
})(jQuery);
```

```
(function($) {
	angular.module("cims").controller('IndexCtrl',
			function($scope, $rootScope, $storage, $window, breadcrumbs,
$location, $http, restrictView,$state) {
				$storage.bind($rootScope, 'isLoggedIn', {
					defaultValue : false
				});

				$storage.bind($rootScope, 'user', {});

				$.extend($rootScope, {
					rootUrl : $window.location.protocol +
$window.location.host + $window.location.port,
					urlPrefix : "#!",
					breadcrumbs : breadcrumbs,
					loginPage : function() {

					},
					doLogout : function() {
						console.log('logout');
						restrictView.clearRoles();
						$rootScope.isLoggedIn = false;
						$rootScope.user = {};
						$http.post('logout').success(function(data) {
						});
					},
					heartbeat : function() {
						$http.post("heartbeat", {})
						  .success(function(data) {
							$rootScope.isLoggedIn = data.success;
							if (data.success) {

								if ($location.path() ==
"/login") {

	$location.path("/dashboard");
									return;
								}
								$scope.getUserRole();
								$scope.getUserDetails();
							}else{
							  $rootScope.user = {};
							}
						}).error(function(){
							restrictView.clearRoles();
						});
					},
					getUserDetails : function(){

	$http.post('user/details',{}).success(function(data, status, headers, config)
{
								$rootScope.user = data;
						});
```

```
                                    },
                                    getUserRole : function() {

      $http.post('user/role').success(function(data, status, headers, config) {
                                          restrictView.clearRoles();
                                          var roles = [];
                                          roles.push(data.role);
                                          roles.push(data.laboratoryUnit + " " +
data.role);
                                          restrictView.addRoles(roles);
                                          if(data.role.toLowerCase().split('
').join('_') == 'system_admin' && $location.path() == "/dashboard"){
                                                $state.go("user.search");
                                          }
                                    });
                                    },
                                    extendTemplates : function(templates) {
                                          return $.extend({}, $rootScope.templates,
templates);
                                    },
                                    templates : {
                                          header : 'app/core/layout/header.html',
                                          footer : 'app/core/layout/footer.html',
                                          sidebar : 'app/core/layout/sidebar.html',
                                          breadcrumb :
'app/core/layout/breadcrumb.html'
                                    }
                              });

                              $scope.heartbeat();
                        });
})(jQuery);

.autocomplete-w1 {
      background: #FFF;
      position: absolute;
      top: 0px;
      left: 0px;
      margin: 6px 0 0 6px; /* IE6 fix: */
      _background: none;
      _margin: 1px 0 0 0;
}

.autocomplete-suggestions {
      border: 1px solid #9BA0AF;
      background: #F8F8F8;
      cursor: pointer;
      text-align: left;
      max-height: 350px;
      overflow: auto;
      -webkit-border-radius: 3px;
      -moz-border-radius: 3px;
      border-radius: 3px;
```

```
	-webkit-box-shadow: 0 1px 0 #fff;
	-moz-box-shadow: 0 1px 0 #fff;
	box-shadow: 0 1px 0 #fff;
	/* IE6 specific: */
	_height: 350px;
	_margin: 0;
	_overflow-x: hidden;
	-moz-border-radius: 5px;
	border-radius: 5px;
	-webkit-box-shadow: 0 1px 0 #fff;
	-moz-box-shadow: 0 1px 0 #fff;
	box-shadow: 0 1px 0 #fff;
}

.autocomplete-suggestion {
	padding: 5px 0px;
	text-indent: 5px;
}

.autocomplete-suggestion:nth-child(2n) {
	background-color: #DDD;
}

.autocomplete-suggestion.autocomplete-selected {
	background-color: black;
	color: white;
}

.autocomplete div {
	padding: 2px 5px;
	white-space: nowrap;
	overflow: hidden;
}

.autocomplete strong {
	font-weight: normal;
	color: #3399FF;
}

@CHARSET "UTF-8";

.required{
	color: red;
	font-weight: bold;
	font-size: medium;
	margin-left: 5px;
}

html,div,map,dt,isindex,form,header,aside,section,section,footer {
	display: block;
}
```

```
html,body {
	height: 100%;
	margin: 0;
	padding: 0;
	font-family: "Helvetica Neue", Helvetica, Arial, Verdana, sans-serif;
	background: #F8F8F8;
	font-size: 12px;
}
#header{
	color:white;
}

html body #sidebar a{
  display:block;
  cursor:pointer;
}

.main-view{

}

.centered {
  display: inline-block;
  vertical-align: middle;
}

.footer {
	position: relative;
	display: block;
	min-height: 100px;
	height: auto !important;
	height: 100px;
	width: 100%;
	margin-left: auto;
	margin-right: auto;
	background: #000000 url(../../images/footer_bg.png) repeat-x;
}

.clickable {
	cursor: pointer;
}

.page-link{
	text-decoration: underline;
}

table.ng-table th{
	text-align: left;
}

<!DOCTYPE html>
<html ng-app="cims">
```

```
<head>
<meta charset="UTF-8">
<meta name="viewport"
  content="width=device-width,height=device-height, initial-scale=1.0, maximum-
scale=1.0, user-scalable=no"
>
<title>LIMS</title>
<link href="components/bootstrap/3.0.0/css/bootstrap.min.css" rel="stylesheet" />
<link href="components/bootstrap/3.0.0/css/bootstrap-theme.min.css" rel="stylesheet"
/>
<link href="assets/css/common/page.css" rel="stylesheet" />
<link href="components/angular/ngTable/0.3.2/ng-table.css" rel="stylesheet" />
<link href="assets/css/common/table.css" rel="stylesheet" />
<link href="assets/css/common/form.css" rel="stylesheet" />
<link href="components/angular/loading-notif/loading-notif.css" rel="stylesheet" />
<link href="components/angular/loading-bar/loading-bar.css" rel="stylesheet" />
<link href="components/angular/validation/ui-validation.css" rel="stylesheet" />

</head>
<body ng-controller="IndexCtrl">
  <div ng-include src="templates.header"></div>
  <div ui-loading-notif></div>
  <div class="row">
    <div class="col-md-2" style="padding-right:0" ng-include
src="templates.sidebar"></div>
    <div class="col-md-10">
      <div ng-include src="templates.breadcrumb"></div>
      <div ui-view class="main-view" class="container"></div>
    </div>
  </div>

  <div ng-include src="templates.footer"></div>

</body>
<!-- jquery -->
<script src="components/jquery/1.10.2/jquery-1.10.2.min.js"></script>
<script src="components/jquery/jquery-validate/jquery.validate.min.js"></script>
<script src="components/jquery/jquery-metadata/jquery.metadata.min.js"></script>

<script src="components/moment/moment.min.js"></script>

<!-- angular/bootstrap -->
<script src="components/bootstrap/3.0.0/js/bootstrap.min.js"></script>
<script src="components/angular/1.2.9/angular.min.js"></script>
<script src="components/angular/1.2.9/angular-animate.min.js"></script>
<script src="components/angular/1.2.9/angular-cookies.min.js"></script>
<script src="components/angular/1.2.9/angular-route.min.js"></script>
<script src="components/angular/1.2.9/angular-resource.min.js"></script>
<script src="components/angular/1.2.9/angular-sanitize.min.js"></script>
<script src="components/angular/1.2.9/angular-touch.min.js"></script>
<script src="components/angular/fix-route-change/route-change.js"></script>
<script src="components/angular/angular-bootstrap/ui-bootstrap-tpls-
0.10.0.min.js"></script>
```

```
<script src="components/angular/ui-router/angular-ui-router.js"></script>
<script src="components/angular/breadcrumbs/angular-breadcrumbs.js"></script>
<script src="components/angular/localStorage/angularLocalStorage.js"></script>
<script src="components/angular/ngTable/0.3.2/ng-table.js"></script>
<script src="components/angular/loading-notif/loading-notif.js"></script>
<script src="components/angular/validation/http-validation-interceptor.js"></script>
<script src="components/angular/validation/ui-validation.js"></script>
<script src="components/angular/loading-bar/loading-bar.js"></script>
<script src="components/angular/authorization/http-auth-interceptor.js"></script>
<script src="components/angular/autofill/ng-autofill.js"></script>
<script src="components/angular/angular-moment/angular-moment.min.js"></script>
<script src="components/angular/ng-restricted-view/ng-restricted-view.js"></script>
<script src="components/angular/ng-enter/ng-enter.js"></script>


<!-- CIMS -->
<script src="app/settings/root-route-config.js"></script>
<script src="app/useraccount/user-route-config.js"></script>
<script src="app/chemistry/chemistry-route-config.js"></script>
<script src="app/physics/physics-route-config.js"></script>
<script src="app/biology/biology-route-config.js"></script>
<script src="app/mathcs/mathcs-route-config.js"></script>
<script src="app/settings/config.js"></script>
<script src="app/index.js"></script>
<script src="app/core/layout/sidebar.js"></script>
<script src="app/settings/login.js"></script>
<script src="app/dashboard/dashboard.js"></script>
<script src="app/useraccount/user.js"></script>
<script src="app/useraccount/user.search.js"></script>
<script src="app/useraccount/user.add.js"></script>
<script src="app/useraccount/user.update.js"></script>
<script src="app/role/role.js"></script>
<script src="app/role/role.add.js"></script>
<script src="app/accountability/accountability.new.js"></script>
<script src="app/accountability/accountability.search.js"></script>
<script src="app/accountability/accountability.report.js"></script>
<script src="app/accountability/accountability.today.js"></script>
<script src="app/maintenancerequest/alert.js"></script>
<script src="app/maintenancerequest/pending-request.js"></script>
<script src="app/maintenancerequest/approved-request.js"></script>
<script src="app/maintenancerequest/rejected-request.js"></script>

<script src="app/chemistry/chemical/chemistry.chemical.search.js"></script>
<script src="app/chemistry/chemical/chemistry.chemical.add-definition.js"></script>
<script src="app/chemistry/chemical/chemistry.chemical.update-
definition.js"></script>
<script src="app/chemistry/chemical/chemistry.chemical.view-definition.js"></script>
<script src="app/chemistry/chemical/chemistry.chemical.add-item.js"></script>
<script src="app/chemistry/labware/chemistry.labware.search.js"></script>
<script src="app/chemistry/labware/chemistry.labware.add-definition.js"></script>
<script src="app/chemistry/labware/chemistry.labware.update-definition.js"></script>
<script src="app/chemistry/labware/chemistry.labware.add-item.js"></script>
<script src="app/chemistry/labware/chemistry.labware.view-definition.js"></script>
```

```
<script src="app/physics/chemical/physics.chemical.search.js"></script>
<script src="app/physics/chemical/physics.chemical.add-definition.js"></script>
<script src="app/physics/chemical/physics.chemical.update-definition.js"></script>
<script src="app/physics/chemical/physics.chemical.add-item.js"></script>
<script src="app/physics/chemical/physics.chemical.view-definition.js"></script>
<script src="app/physics/labware/physics.labware.search.js"></script>
<script src="app/physics/labware/physics.labware.add-definition.js"></script>
<script src="app/physics/labware/physics.labware.update-definition.js"></script>
<script src="app/physics/labware/physics.labware.add-item.js"></script>
<script src="app/physics/labware/physics.labware.view-definition.js"></script>

<script src="app/biology/labware/biology.labware.add-item.js"></script>
<script src="app/biology/chemical/biology.chemical.search.js"></script>
<script src="app/biology/chemical/biology.chemical.add-definition.js"></script>
<script src="app/biology/chemical/biology.chemical.update-definition.js"></script>
<script src="app/biology/chemical/biology.chemical.add-item.js"></script>
<script src="app/biology/chemical/biology.chemical.view-definition.js"></script>
<script src="app/biology/labware/biology.labware.search.js"></script>
<script src="app/biology/labware/biology.labware.add-definition.js"></script>
<script src="app/biology/labware/biology.labware.update-definition.js"></script>
<script src="app/biology/labware/biology.labware.add-item.js"></script>
<script src="app/biology/labware/biology.labware.view-definition.js"></script>

<script src="app/mathcs/labware/mathcs.labware.search.js"></script>
<script src="app/mathcs/labware/mathcs.labware.add-definition.js"></script>
<script src="app/mathcs/labware/mathcs.labware.update-definition.js"></script>
<script src="app/mathcs/labware/mathcs.labware.add-item.js"></script>
<script src="app/mathcs/labware/mathcs.labware.view-definition.js"></script>
</html>
```

XII. Acknowledgement

Foremost, I would like to express my sincere gratitude to my adviser Prof. Avegail D. Carpio, M.Sc.for the continuous support of my study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better adviser and mentor for my study.

Besides my adviser, I would like to thank my bestfriend John Michael Vincent Rustia for his encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Ishmael Viñas, Jose Louie Mark Año, Bona Rae Villarta and John Simon Tayson for offering me a copy of their thesis as my reference of all the formatting.

Last but not the least, I would like to thank my family: my parents Romeo San Miguel and Ma. Teresa San Miguel, for giving birth to me at the first place and supporting me spiritually throughout my life.