CSCD01 Deliverable 4

Team Number: 11
Team Name: Phantom Developers
Date: April 7th, 2020

Table of Contents

User Guide

OpenMRS JIRA Issue: Support formulary status for drugs

Description

This feature adds a “formulary” status to drugs, which are used to put drugs into a list. Each drug can be assigned to more than one formulary. Queries and management of formulary should also be supported. Formularies are lists of approved medication for prescription. They are used by hospitals and insurance companies to determine the eligibility of approving a claim, being prescribed, etc.

Requirements

- Clients should be able to get a list of available formularies.
- Clients should be able to check if a drug is on a particular formulary.
- Clients should be able to get the list of formularies for a particular drug.
- Clients should be able to search for a drug within a specified list of formularies.
- Clients should be able to ask the API for a (paginated) list of all drugs within a given formulary.
- Clients should be able to manage the list of formularies and assignment of drugs to formularies.

User Guide Installation

Database Installation

1. Download and install MySQL 5.6 (MySQL version should below 5.6.7)
2. Use Setup type: Developer Default
3. Remember root password set

Database Startup

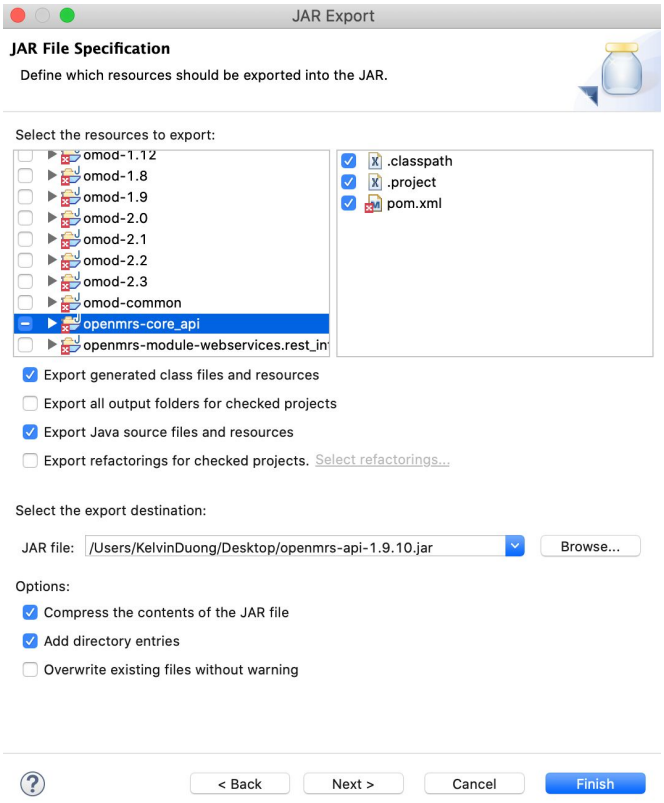
1. Open terminal
2. Go to %ProgramFiles%/MySQL/MySQL Server 5.6/bin
3. Type mysqld to start the database server

OpenMRS Installation

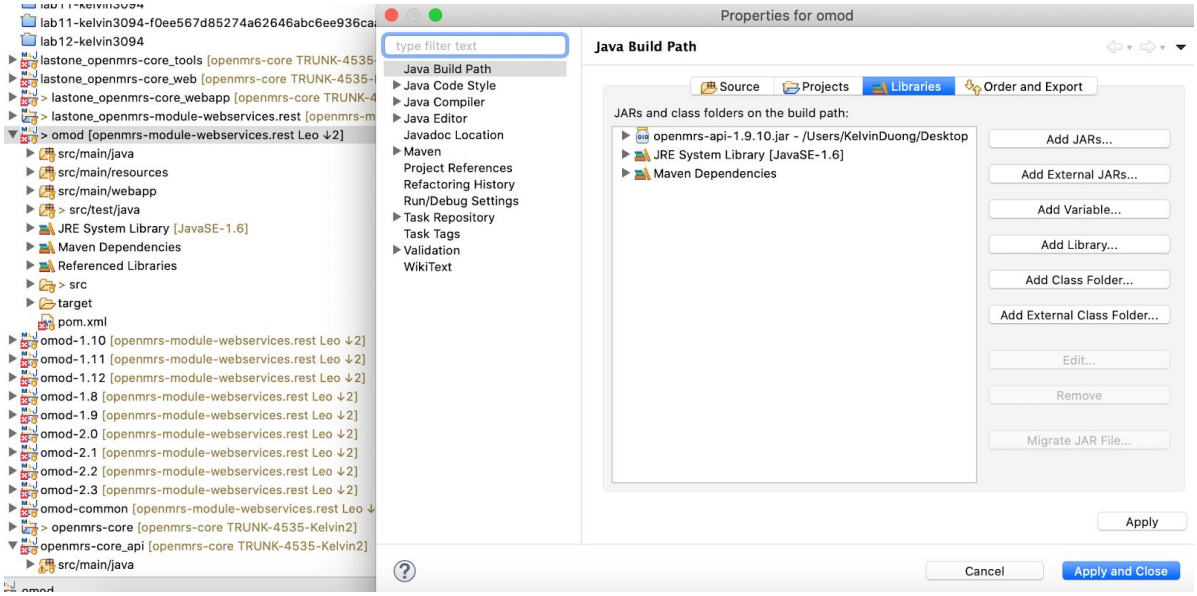
1. Install Java SE Development Kit 8.
2. Clone OpenMRS-Core
3. In terminal, cd into the openmrs-core folder of OpenMRS-Core and run “mvn clean install”
4. Cd into webapp and run mvn jetty:run
5. On your browser, go to “http://localhost:8080/openmrs/” to get the installation prompt, make sure to have the database open
6. After this, cancel the webapp in the terminal
7. Put the given omod files for legacy ui and rest api inside (${HOME}/.OpenMRS/modules) or (Windows + R > %appdata%>OpenMRS>Modules)
8. In terminal, cd into the openmrs-core folder of OpenMRS-Core again and run “mvn clean install”
9. cd into webapp and run mvn jetty:run

OpenMRS Manual Installation

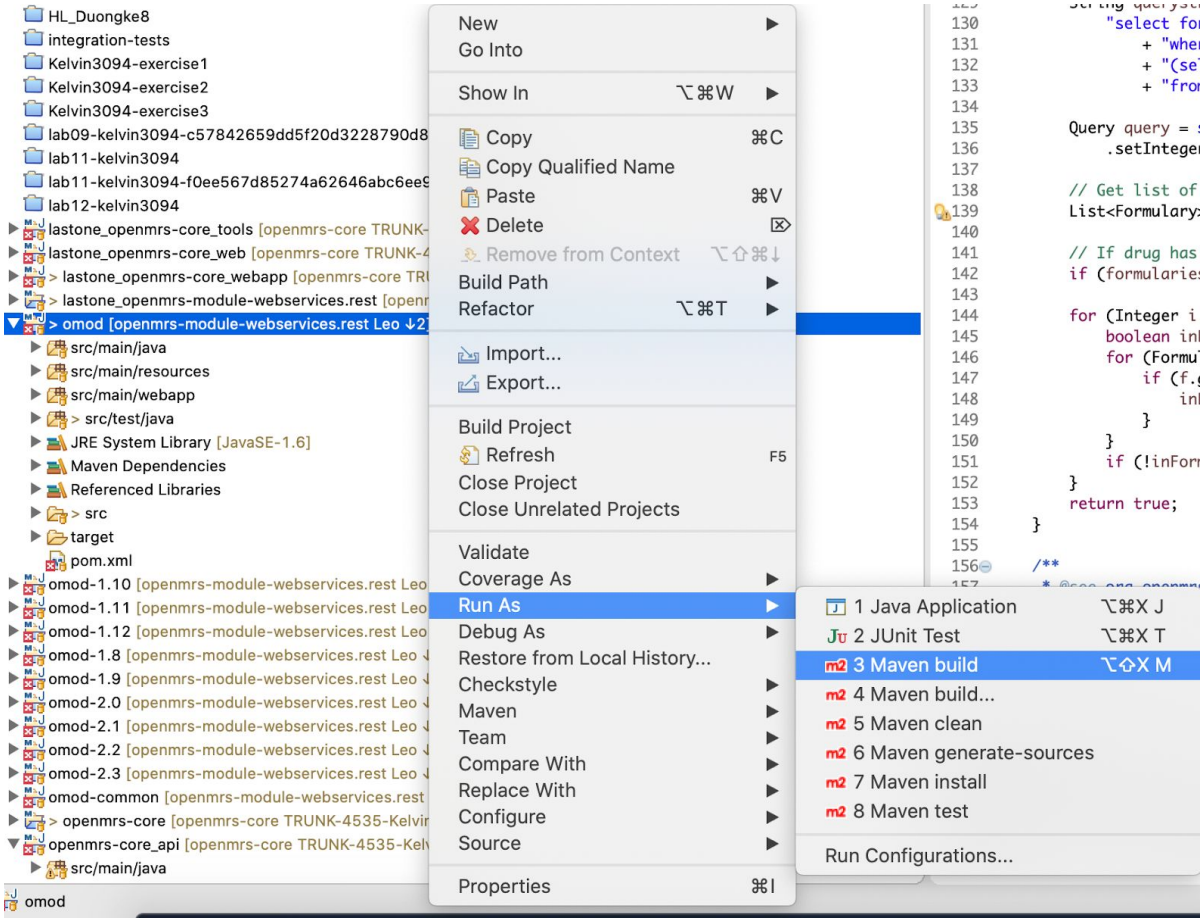
10. Clone OpenMRS Legacy UI, and Rest API and open projects in Eclipse
11. Install the OpenMRS Legacy UI first by following this guide
12. In Eclipse, export the openmrs-core-api package from OpenMRS-Core as a JAR file, with ‘Export generated class files and resources’ and ‘Export Java source files and resource’ checked. Name it “openmrs-api-1.9.10.jar”



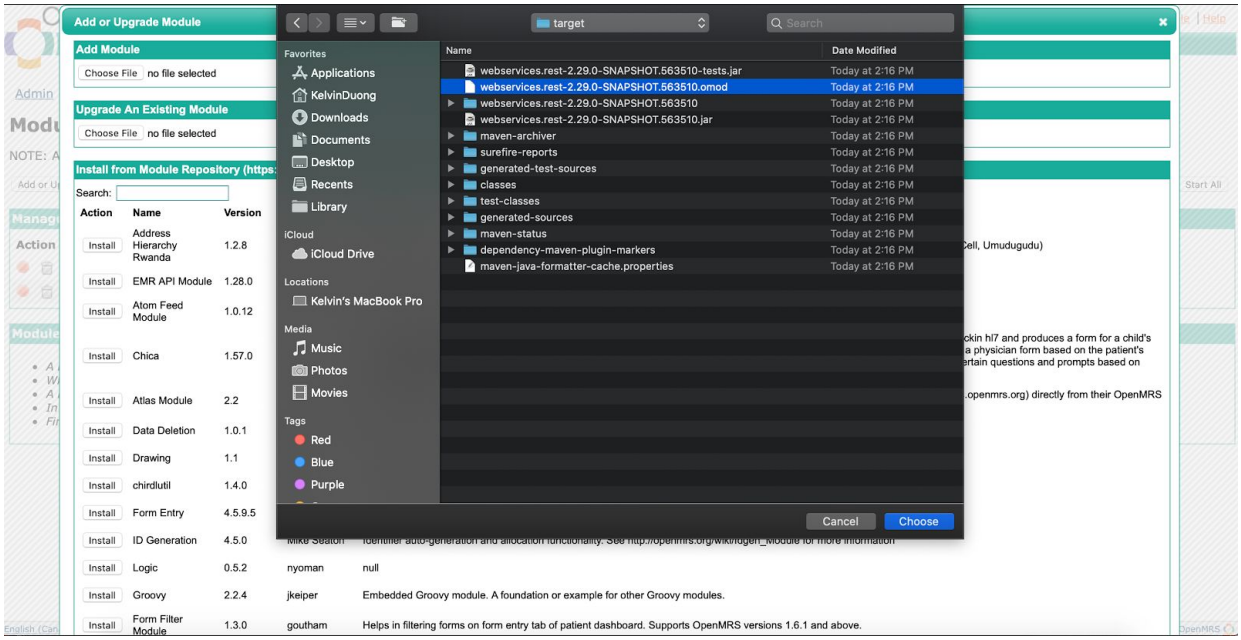
13. Add the “openmrs-api-1.9.10.jar” JAR file to the build path of the omod package



14. Build only the omod package with your IDE



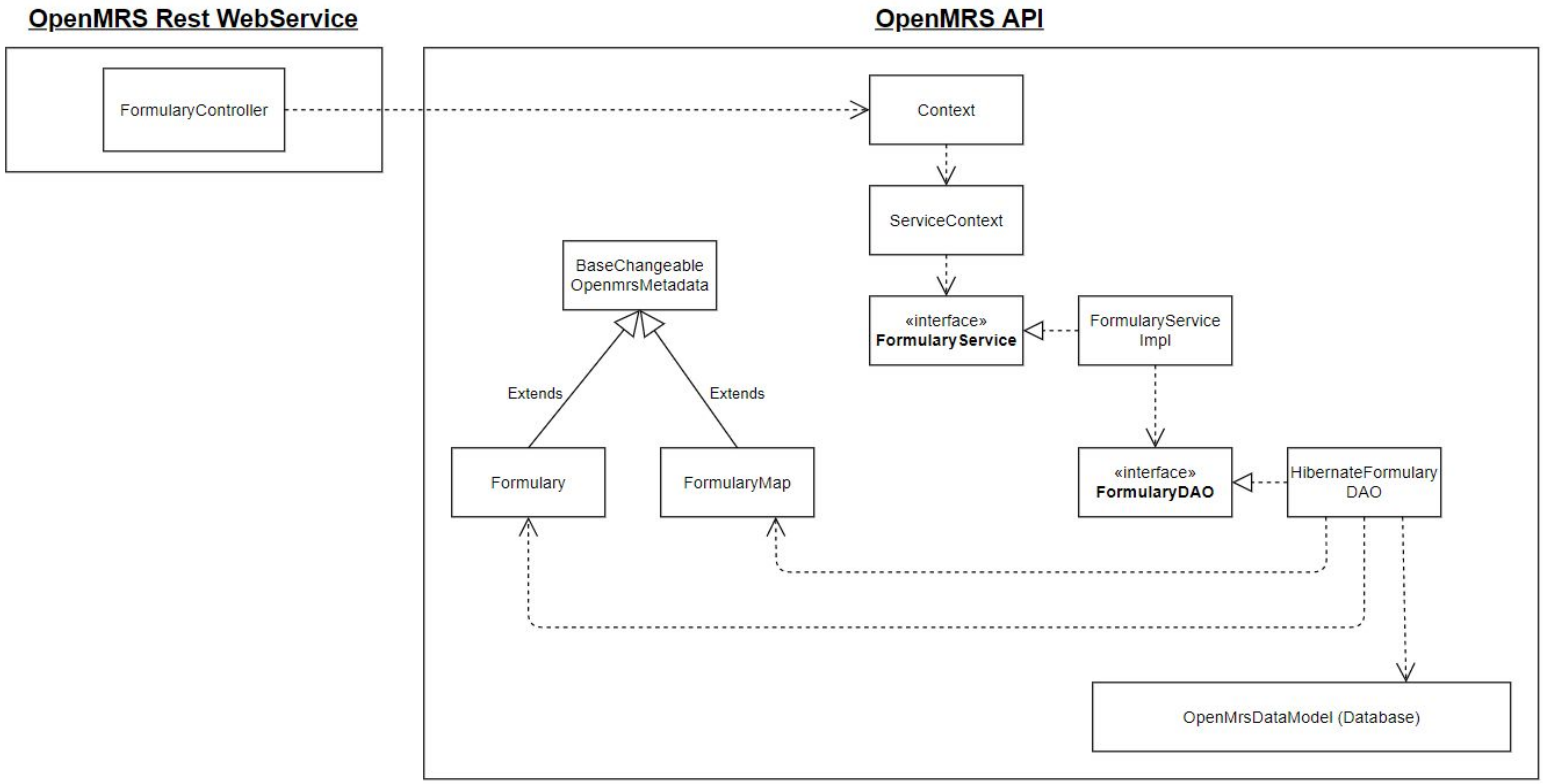
15. In terminal, cd into the openmrs-core folder of OpenMRS-Core and run “mvn clean install”
16. Then cd into the webapp folder and run “mvn jetty:run”
17. On your browser, go to “http://localhost:8080/openmrs/” with username “admin” and password “Admin123” (Note: Must install legacy UI first or nothing will show)
18. If this is your first time installing, you should have a prompt to update the database, click the green arrow button.
19. Stop openmrs using CTRL-C and run it again with “mvn jetty:run”
20. Login to OpenMRS again
21. Click the Administration tab on the top bar
22. Under Modules, click Manage Modules
23. Install the REST Api module. After you have built the project, it should be located in /openmrs-module-webservices.rest/omod/target/webservices.rest-2.29.0-SNAPSHOT.563510.omod (the last 6 digits in the file name may be different).



24. Click Administration again. Under REST Web Services, click test. This page can be used for the acceptance tests below and should look like this.

Code Documentation

Design



UML Diagram

To support the formularies, we first updated the database in liquibase-update-to-latest.xml to add the new tables drug_formulary and drug_formulary_map.
The HibernateFormularyDAO class allows us to execute queries on the new tables in the database.
We also added the Formulary and FormularyMap classes and the hibernate mapping files for the hibernate mapping which is used in the DAO. This allows for the hibernate database to properly convert rows in the MySQL database with the Java Objects in the source code.
The FormularyService, FormularyDAO is created as a SpringBean Object, that allows Spring to manage them and expose the endpoints to be accessible from the static Context class. Since these were newly created, we updated the Context class and the applicationContex-service xml to support them.
In the REST module web service repository, we then added the FormularyController to handle the requests and give responses to the user by calling the OpenMRS-core API. This was created as we needed a way for the user to interact with the feature we implemented.

Differences

We made some changes to our original design from deliverable 3. First, we added a hibernate mapping file that maps the Formulary class to the drug_formulary table in the database making it easier to insert, update, and get the relations from the database because it allows you to save the object and Hibernate will automatically do the database updates without having to write SQL. We also added a FormularyMap class, because that allows us to use Hibernate to map the FormularyMap class to the drug_formulary_map table, allowing the benefit of not querying the SQL database directly.
We also did not modify liquibase-schema-only.xml and instead modified liquibase-update-to-latest.xml to follow the OpenMRS guidelines on making database changes.
We modified the rest module for OpenMRS and added a rest api to give front facing access for executing the customer acceptance tests.

Code Changes

OpenMRS-core

Implementation

Added Formulary and FormularyMap object classes
Added FormularyService interface
Added FormularyServiceImpl class for extra business logic to create and delete objects
Added FormularyDAO interface
Added HibernateFormularyDAO class for main business logic for interacting with the database
Modified liquibase-update-to-latest.xml to update the database
Added Formulary.hbm.xml and FormularyMap.hbm.xml to add mapping to the new tables
Updated applicationContext-service.xml to add connecting to new classes and service
Updated Context class to get the formulary service

Testing

Added FormularyServiceTest junit test class
Added HibernateFormularyDAOTest junit test class
Updated standardTestDataset.xml to add data to the mock database

OpenMRS RESTful Web Services

Implementation

Added FormularyController class to support user requests

Acceptance Test Suite

In a Terminal, enter the following commands to make a request. The uuid's and id's may not be exactly the same.

Save Formulary

Description	Clients should be able to create new formulary
Request	curl -i -u Admin:Admin123 -X POST http://localhost:8080/openmrs/ws/rest/v1/formulary/saveFormulary -d '{ "name" : "formulary name", "description" : "formulary desc", "doses" : 10 }'
Expected Result	{ "uuid" : "11f2f381-c93e-4867-9a1d-6eed19bc661c", "name" : "formulary name", "description" : "formulary desc", "retired" : false, "formularyId" : 1, "doses" : 10, "id" : 1 }

Description	Clients should be able to create new formulary and id should be auto incrementing
Request	curl -i -u Admin:Admin123 -X POST http://localhost:8080/openmrs/ws/rest/v1/formulary/saveFormulary -d '{ "name" : "another formulary name", "description" : "new formulary desc", ”clinicalInformation” : ”important information”, "doses" : 200 }'
Expected Result	{ "uuid" : "4a90f649-495c-4f2c-b269-3f65caea4017", "name" : "another formulary name", "description" : "new formulary desc", "retired" : false, "formularyId" : 2, "clinicalInformation" : "important information", "doses" : 200, "id" : 2 }

Clients should be able to get a list of available formularies

Description	Clients should be able to get a list of formularies
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/getAllFormularies

Expected Result	[{
	"uuid" : "11f2f381-c93e-4867-9a1d-6eed19bc661c",
	"name" : "formulary name",
	"description" : "formulary desc",
	"retired" : false,
	"formularyId" : 1,
	"doses" : 10,
	"id" : 1
	}, {
	"uuid" : "4a90f649-495c-4f2c-b269-3f65caea4017",
	"name" : "another formulary name",
	"description" : "new formulary desc",
	"retired" : false,
	"formularyId" : 2,
	"clinicalInformation" : "important information",
	"doses" : 200,
	"id" : 2
	}]

Description	Clients should be able to get a list of formularies with specific page number and limit
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/getAllFormularies?limit=1&page=0
Expected Result	[{ "uuid" : "11f2f381-c93e-4867-9a1d-6eed19bc661c", "name" : "formulary name", "description" : "formulary desc", "retired" : false, "formularyId" : 1, "doses" : 10, "id" : 1 }]

Privileged clients should be able to manage the list of formularies and assignment of drugs to formularies

Description	Clients should be able to insert a drug to a particular formulary
Request	curl -i -u Admin:Admin123 -X POST http://localhost:8080/openmrs/ws/rest/v1/formulary/insertDrugToFormulary?drugId=2&formularyId=1
Expected Result	Success

Description	Clients should be able to insert a drug to a particular formulary
Request	curl -i -u Admin:Admin123 -X POST http://localhost:8080/openmrs/ws/rest/v1/formulary/insertDrugToFormulary?drugId=3&formularyId=1
Expected Result	Success

Clients should be able to check if a drug is on a particular formulary

Description	Clients should be able to check if a drug is in a particular formulary
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/isDrugInFormulary?drugId=2&formularyId=1
Expected Result	true

Description	Clients should be able to check if a drug is in a particular formulary
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/isDrugInFormulary?drugId=2&formularyId=2
Expected Result	false

Clients should be able to get the list of formularies for a particular drug

Description	Clients should be able to get a list of formularies for a particular drug
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/getFormulariesFromDrug?drugId=2
Expected Result	[{ "uuid" : "11f2f381-c93e-4867-9a1d-6eed19bc661c", "name" : "formulary name", "description" : "formulary desc", "retired" : false, "formularyId" : 1, "doses" : 10, "id" : 1 }]

Description	Clients should be getting an empty list if no formularies for a specific drug
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/getFormulariesFromDrug?drugId=4
Expected Result	[ ]

Clients should be able to search for a drug within a specified list of formularies

Description	Clients should be able to search for a drug within a specified list of formularies
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/isDrugInFormularies?formularyId=1&formularyId=2&drugId=2
Expected Result	true

Description	Should be able to return false if a drug is not within a specified list of formularies
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/isDrugInFormularies?formularyId=1&formularyId=2&formularyId=3&drugId=5
Expected Result	false

Clients should be able to ask the API for a list of all drugs within a given formulary

Description	Clients should be able to get a list of drugs for a specific formulary
Request	Request: curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/getDrugsFromFormulary?formularyId=1

Expected Result	[{
	"uuid" : "a5d200e4-744b-11ea-9de3-00ff3772e317",
	"name": "Triomune-30",
	"combination" : "1",
	"drugId" : "2",
	"conceptId" : "792"
	"id" : "2"
	},
	{
	"uuid" : "a5d201f5-744b-11ea-9de3-00ff3772e317",
	"name": "Triomune-40",
	"combination" : "1",
	"drugId" : "3",
	"conceptId" : "792"
	"id" : "3"
	}]

Description	Clients should get an empty list if no drugs for a specific formulary
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/getDrugsFromFormulary?formularyId=3
Expected Result	[ ]

Clients should be able to delete given formulary

Description	Clients should delete a specific formulary and all the formulary drug mapping of the formulary
Request	curl -i -u Admin:Admin123 -X DELETE http://localhost:8080/openmrs/ws/rest/v1/formulary/deleteFormulary?formularyId=1
Expected Result	Success

Description	Clients should be able to remove a specific drug for the given formulary
Request	curl -i -u Admin:Admin123 -X DELETE http://localhost:8080/openmrs/ws/rest/v1/formulary/removeDrugFromFormulary?formularyId=2&drugId=2
Expected Result	Success

Clients should get an error when putting an invalid drugId or formularyId

Description	Clients get an error when the drugId is not in the database
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/isDrugInFormularies?formularyId=1&formularyId=2&drugId=99
Expected Result	Status Code: 400. Invalid drug id

Description	Clients get an error when the formularyId is not in the database
Request	curl -i -u Admin:Admin123 http://localhost:8080/openmrs/ws/rest/v1/formulary/isDrugInFormularies?formularyId=1&formularyId=2&formularyId=99&drugId=5
Expected Result	Status Code: 400. Invalid formularyId

Unit Test Suite

The name of the tests should be self explanatory

HibernateFormularyDAOTest

Tests to verify methods in the DAO

saveFormulary_shouldSaveFormulary
getFormulary_shouldGetFormulary
getFormulary_shouldReturnNullIfFormularyDoesNotExists
getAllFormularies_shouldGetAllFormularies
getAllFormularies_shouldGetFormularyGivenPageAndLimit
getDrugsFromFormulary_shouldGetDrugs
getDrugsFromFormulary_shouldThrowExceptionWhenInvalidFormularyId
isDrugInFormulary_shouldDrugBeInFormulary
isDrugInFormulary_shouldDrugNotBeInFormulary
isDrugInFormulary_shouldDrugInFormularyFormularyNotExist
isDrugInFormularies_shouldBeTrueIfDrugIsInFormularies
isDrugInFormularies_shouldBeFalseIfDrugIsInNoFormularies
isDrugInFormularies_shouldBeTrueIfDrugIsInSomeFormularies
isDrugInFormularies_shouldThrowExceptionForInvalidDrugId
isDrugInFormularies_shouldThrowExceptionForInvalidFormularyId
getFormulariesFromDrug_shouldgetFormularies
insertDrugToFormulary_shouldinsertdrug
deleteFormulary_shouldDeleteFormularyFromDatabase
deleteFormulary_shouldCascadeDelete
deleteFormulary_shouldGiveErrorIfInvalidFormularyId
removeDrugFromFormulary_shouldRemoveDrugFromFormulary
removeDrugFromFormulary_shouldGiveErrorIfInvalidDrugId

FormularyServiceTest

Tests to verify methods in the DAO

shouldSaveGetDeleteFormulary
getAllFormularies_shouldGetAllFormularies
shouldSaveGetDeleteFormularyMap