Restful Web Service for Library Management System (Spark- Java)

Technologies used: Eclipse, Java 8, Spark – Java, Freemaker template engine, html, css, javascript

Lines of code: 700

Summary

Spark Framework is a simple and lightweight Java web framework built for rapid development. Spark's intention is to provide a pure Java alternative for developers that want to (or are required to), develop their web application in Java. Spark is built around Java 8's lambda philosophy, which makes a typical Spark application a lot less verbose than most application written in other Java web frameworks.

We have a simple domain class called Book.java and User.java with a few properties and a DAO class called UserService.java that provides some basic CRUD functionality and interface to access the underlying data structure.

To get started with Spark I added all the necessary Maven dependencies to the pom.xml.

Next I created a controller class called UserController.java that is responsible for handling incoming requests. I implemented the following requests which have the following functionality:

1) createUser - Creates the user if he is not already available in the data store.

2) getAllUsers- Gives the list of all users registered in the library

3) updateUser- Update an existing user

4) getAllBooks- Gives list of all books in the library

5) findBookByName- Given the name of the book, searches for any book that contains the given name

6) addBook- Adds a new book to the inventory

7) checkOutBook - Takes the user id and book id as inputs. A book that is already checked out, cannot be checked out again (send proper error codes back)

Spark has native support for a lot of template engines. I made use of FreeMaker template engine for my MVC model. I created templates for my html content having .ftl extension and stored it in the views folder. I included all fonts, css and javascript in another public folder. The basic idea was to allow me to create a sample page and then replace the sample content with placeholders that were going to be dynamically replaced by the actual content at each request.

Lastly I perfomed Junit testing to test all my request handling functions.