****

**Athlone Institute of Technology**

**Project Thesis – Academic Year 2019/20**

**Book Management System**

**Using**

**Rest API**

**Lecturer: Thiago Braga Rodrigues**

**Chhaya Sharma**

**A00268860**

**Bachelor of Engineering (honors) Software Engineering**

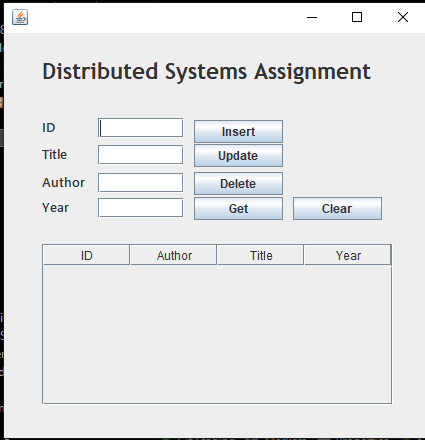
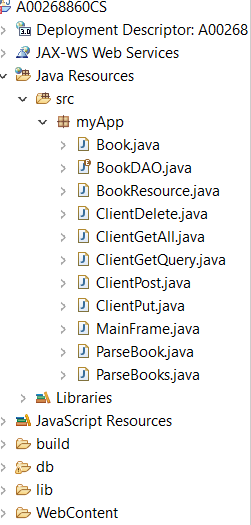
# **Introduction**

With Book Management System holds a database for storing and accessing information about books and runs on a **Tomcat 7.0** server. These records are stored in **HSQLDB** database, which stores id, Title, author, and year. Information is retrieved and processed using **XMLPullParser** to identify it using tags. The server utilizes **JAX:RS API** to access and display the data.

**Functionalities:**

* **Delete:** Enter an id number to delete book details from the system.
* **Get All:** Return the information of all books in database
* **Get:** To get the details of a book by id
* **Put:** Enter the current id of a book and name, title, author that needs to be updated.
* **Post:** Enter the details for a new book and create it on server.

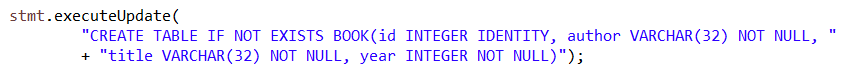
### **Main Window Project Structure**



# **Requirement 1**

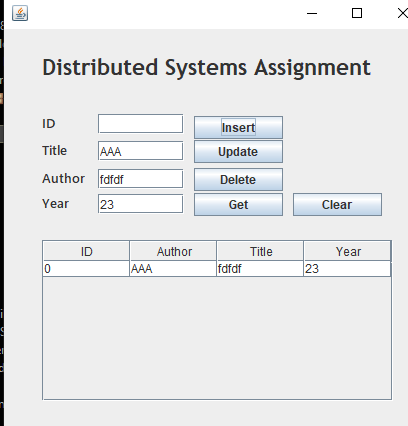
“**Book table should be created at runtime”**

**Creation SQL**



**Steps to be followed for First time user**

1. Run Tomcat Server (here we are using Tomcat 7.0)
2. Run Ant
3. Run CreateTable.java class to create the table
4. Run Mainframe.java as java application

Select, Update, and Delete functions are present in BookDatabase.java class is called from BookDao.java (Data Access Object) files and connect to the HSQLDB server with a direct connection. A data access object acts like an interface between the client and HSQLDB server, which will allow functions to be called and it would not expose too much information about the server. These are some examples for getting books

As this write up should not include too much code, for more information about implementation of SQL commands please refer BookDao.java and BookDatabase.java

# **Requirement 2**

A Jax-Rs/ Jersey client which can send HTTP requests like GET/PUT/POST/DELETE, parse the response XMLPullParser and outputs to the GUI and

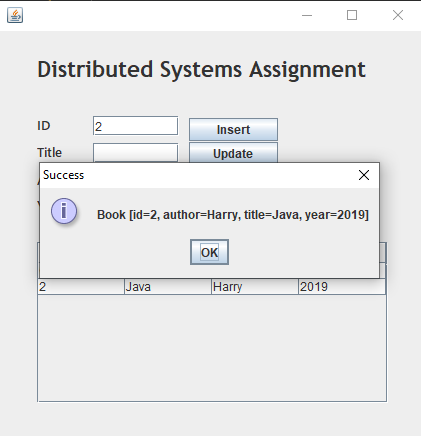
A tomcat server that responds to all the HTTP requests GET/PUT/POST/DELETE

## **Completion**

## **Get:**

Two different @GET JAX:RS commands implemented,

1. It will return all the books stored on the server
2. For returning a specific book by id. The single set will work by passing id as a part of the path.
3. XMLPullParser is used to separate the information returned by its tags which can be seen in ParseBook.java class. When the information is returned it is then pull apart using set of if statement to find which tag of XML the parser is currently in.

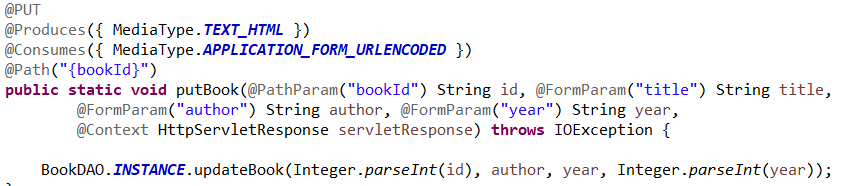


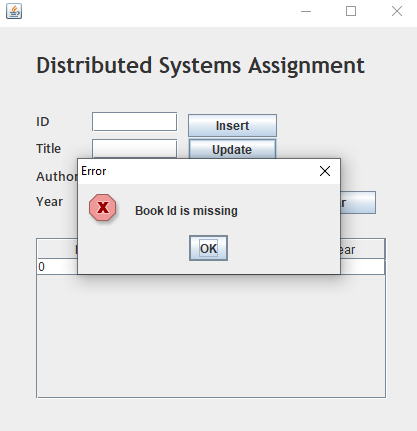
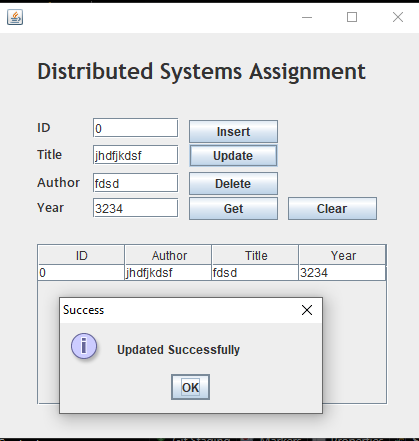
# **Requirement 3**

GUI that handles GET/PUT/POST/DELETE

## **PUT**

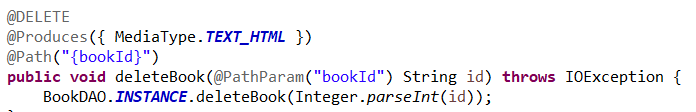
@PUT command is used to update book details as title, author and year. For this is passed as parameter and the record to which it belongs will be updated. These are parsed by adding them as BasicNameValuePairs to a nameValuePairs list and set as the URLEncodedFormEntity for the httpPut request.

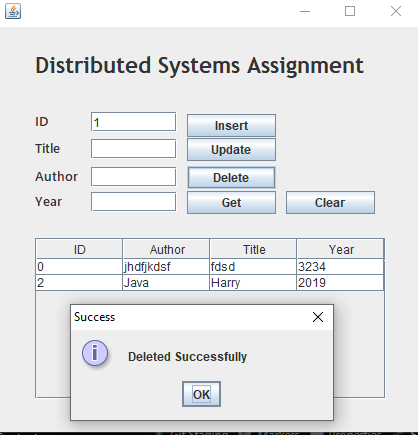




## **DELETE**

@Delete takes one parameter as id and delete the corresponding record. DELETE works similar to GET command except that it calls a command in hsqldb based on name provided





## **POST**

@POST Takes in 4 parameters from the user for a new book and then passes to the BookResource object in a similar way to the PUT (I.e. with parameters) and then on the BookDao. to create the object in SQL.

