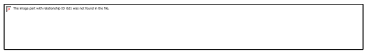
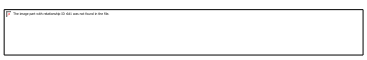


Basic Spring 5.0



Document Revision History

Date	Revision No.	Author	Summary of Changes
Aug-2012	2.0	Mohan C	Lab book exercises are revamped
June-2013	3.0	Mohan C	Lab book exercises are revamped
June-2015	4.0	Rathnajothi.P	Upgraded from spring version 3 to 4
May-2016	5.0	Vinod Satpute	Revamped as per the integrated ELT TOC
June- 2016	6.0	Vinod Satpute Yukti Valecha Tanmaya Acharya	Modified as per Toc for ELTP
Jan-2018	7.0	Bharati Thorat	Lab Book are revamped

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning Basic Spring 4.0. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for Spring Framework

Here is what is expected on your machine in order for the lab assignments to work.

Minimum System Requirements

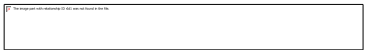
- Intel Pentium IV or higher
- Microsoft Windows (NT 4.0/XP/2K)
- Memory: 256MB of RAM (512 recommended)
- 500MB hard disk space
- JDK version 1.8 + with help, Netscape or IE
- MS-Access/Connectivity to Oracle database
- Wildfly
- Spring Tool Suite
- Spring4.0 API from https://spring.io/docs. Download spring-framework-4.0.3.RELEASE-with-docs.zip, which contains the documentation also and unzip it.

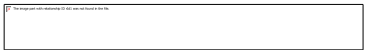
Creating the first Spring application:

- ✓ Ensure that Java 8 is installed and Eclipse Luna is available.
- ✓ You will need Wildfly server to work with.
- ✓ Unzip the spring-framework-4.0.3.RELEASE-with-docs.zip into any folder.
- ✓ Create a new project in eclipse and name it.

NOTE:-Do All the Lab Using Following 3 Approaches.

- Xml Base Configuration Approach
- Annotation Base Approach
- Java Configuration Approach





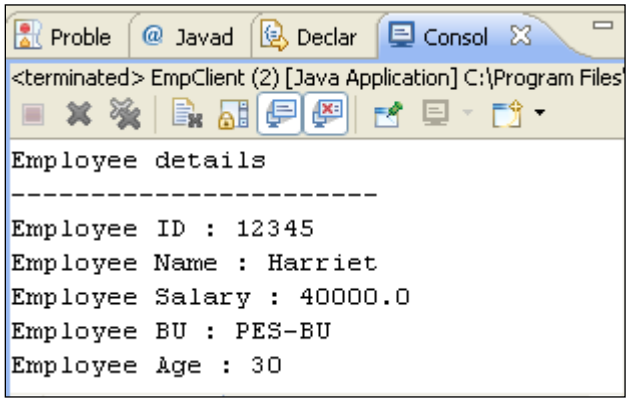
Lab 1. Injecting dependencies into a Spring application

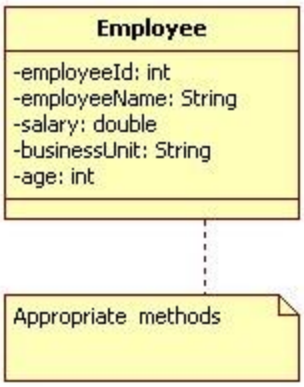
Goals	• Using IoC to integrate disparate systems in a loosely coupled manner.
Time	180 minutes

Problem statement-1.1: Injecting dependencies

Write an Employee bean. Inject values into bean using DI and display all values. Refer the class diagram below

The output would look as shown below:



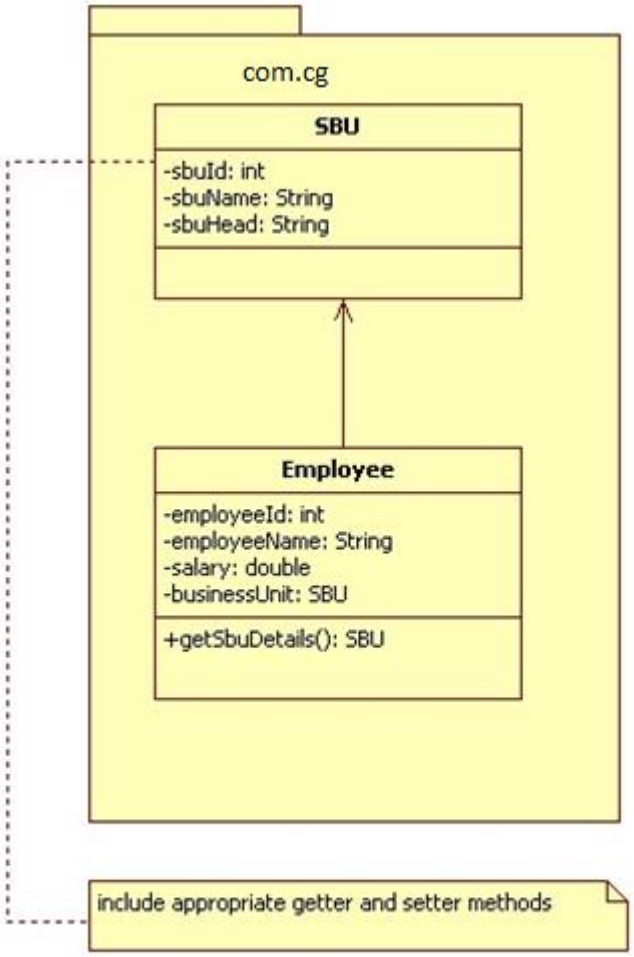


Class Diagram 1: Employee

Keep each of the lab solutions separate, preferably in different packages/source folders

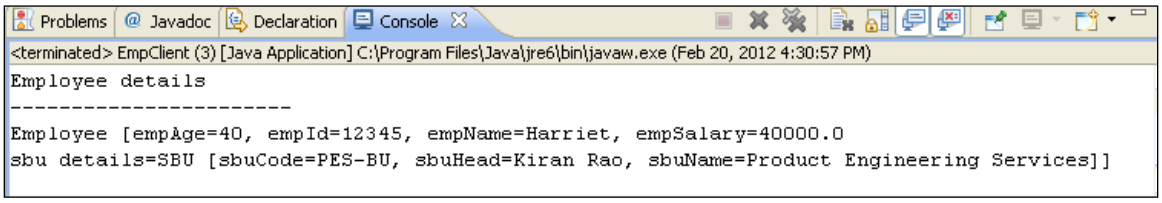
Problem statement-1.2: Injecting dependencies

Code SBU bean. Revisit the Employee bean and provide a method to retrieve SBU details (getSBUDetails()) for the employee. You will need to inject the SBU bean to the Employee bean as shown in the Class diagram below:



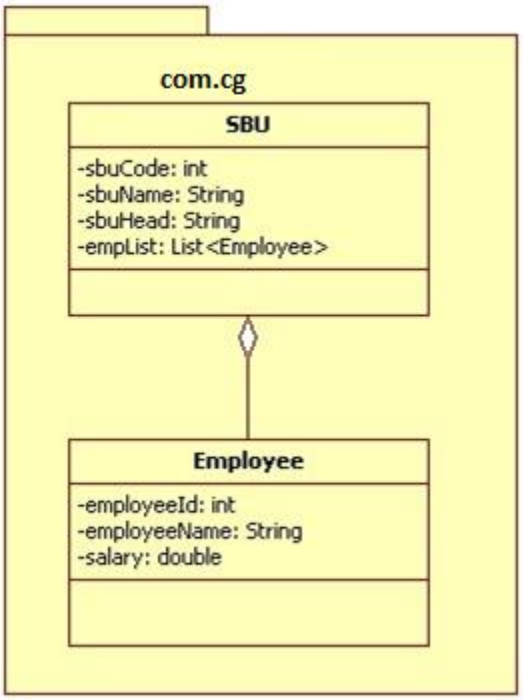
Class Diagram 2**: SBU and Employee**

The output would look as shown below:



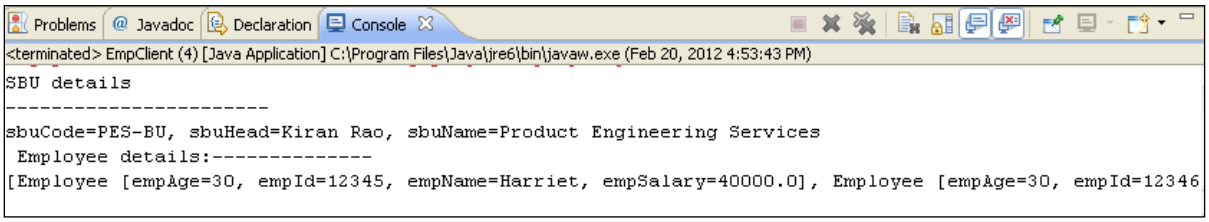
Problem statement-1.3: Injecting dependencies

Revisit the SBU bean. Create a new property called empList which will contain a list of all employees in the PES BU. Display the SBU details, followed by a list of all employees in that BU. To inject employee objects into the SBU bean, use "List" collection. Allocate two employees to PES. Refer Class diagram below



Class Diagram 3**: SBU and Employee (Ver -2)**

The output would look as shown below:



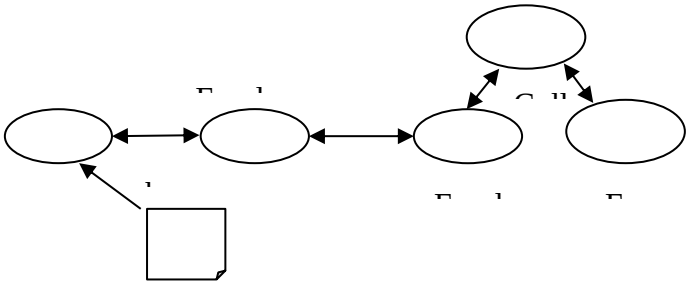
Problem statement-1.4: Injecting dependencies

Develop a console based spring application where main method of client class will retrieve employee information from Employee collection and displays info in the console as shown:

Input:
Employee ID : 100

Output:
Employee Info:
Employee ID :100
Emplpoyee NAME :Rama
Employee SALARY :12345.67

Refer diagram below for implementation details:



Note: implement above application using

- Setter Injection
- Constructor Injection (use index and type attribute with constructor arg tag)
- Use different bean wiring mechanism like..
  - By Name
  - By Type
  - Auto Wiring

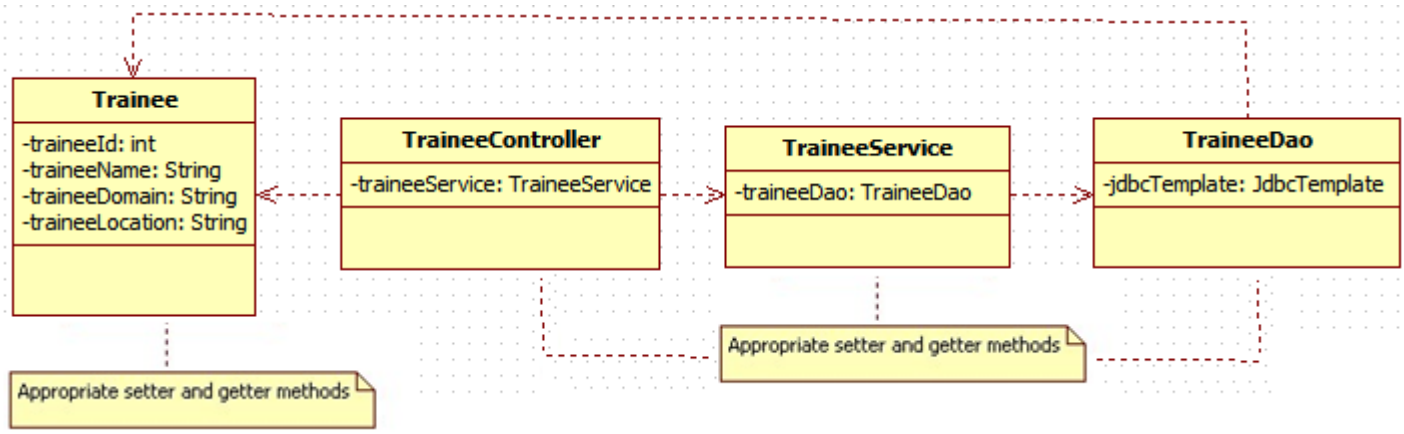
Change Request: Now Change the application so that all components are autowired and components are automatically scanned.Use Spring boot API.

Lab 2. Spring MVC and JPA

Goals	• Demonstrate Spring's MVC framework • Integrate Spring and JPA
Time	180 minutes

Version -1:

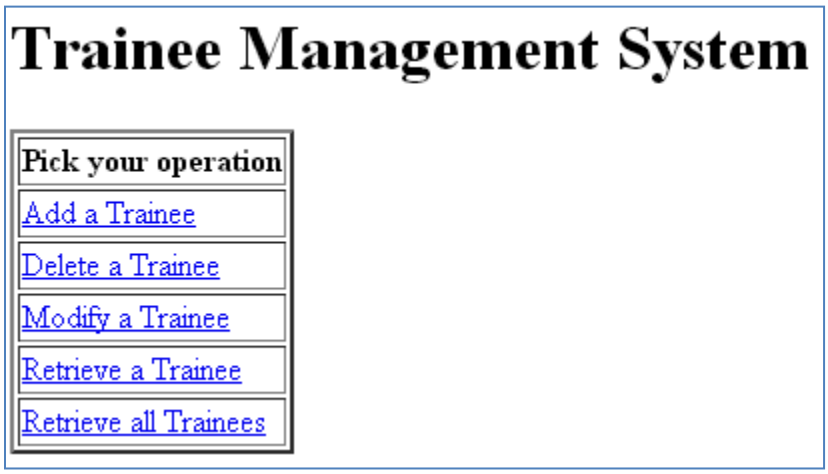
Develop a Spring MVC based application to manage list of trainees by an admin. Refer the class diagram below to develop required classes.



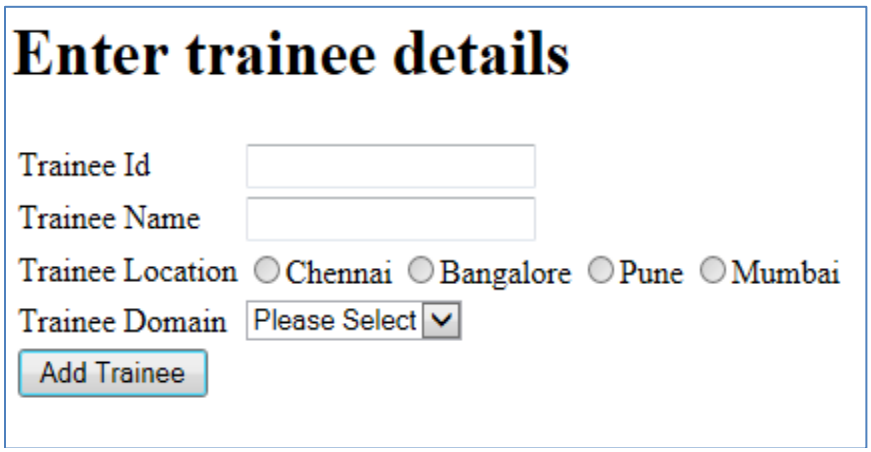
Class Diagram 4**: Trainee related Classes**

Initially when the application is deployed login page should come up as shown below.

Administrator will enter valid credentials to go to the page shown below.

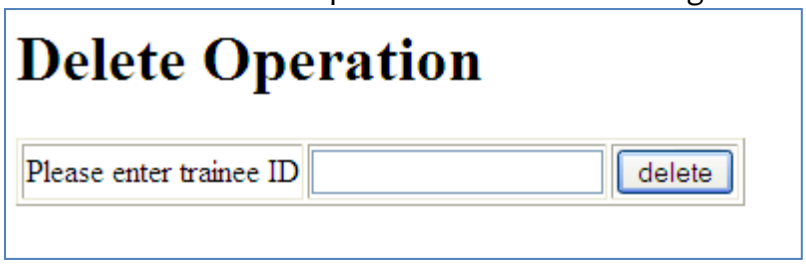


Admin select add hyperlink to add a new trainee. The following page shows up.

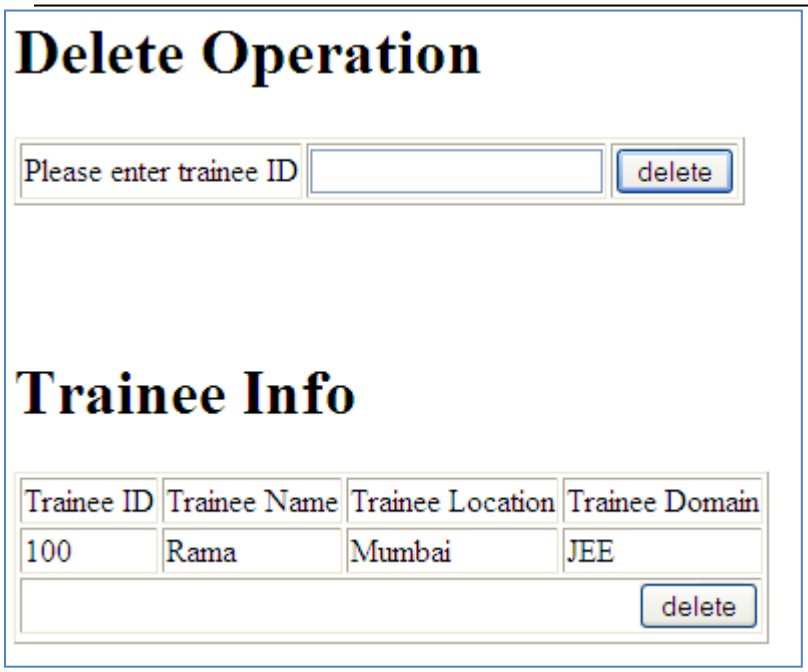


After entering trainee details admin will submit, to insert trainee info into database.

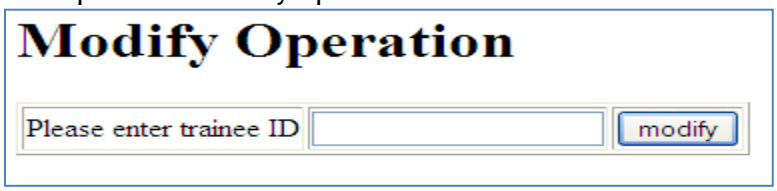
If admin selects delete operation refer the following screen shots to design your application.



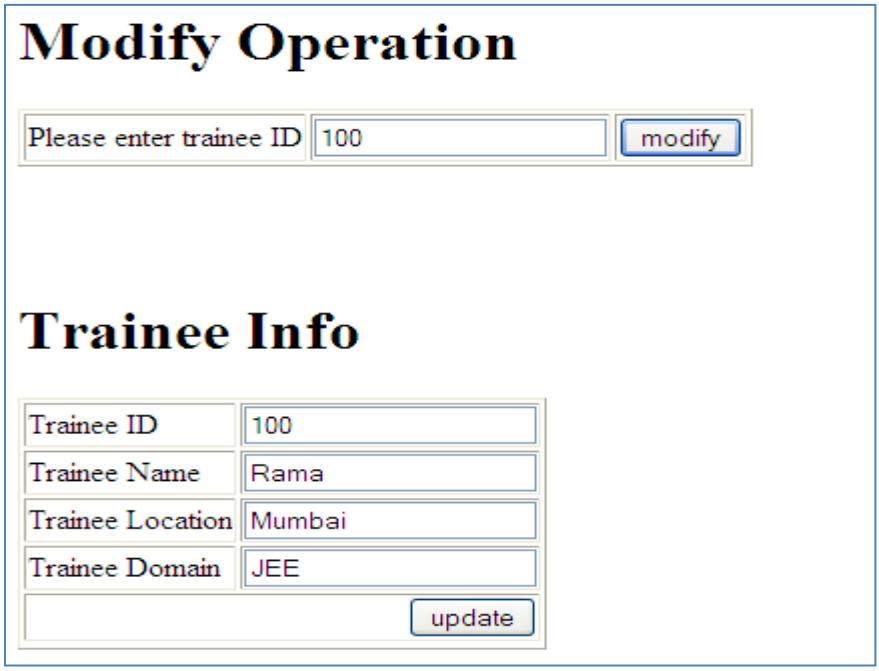
After entering trainee ID in the same page trainee info will be displayed as shown below



To implement modify operation on trainee info refer the following list of screen shots.

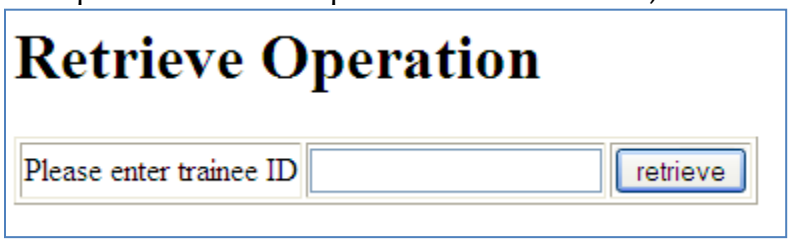


When admin enters trainee id, in the same page trainee info will be retrieved for modification as shown below in the screen shot

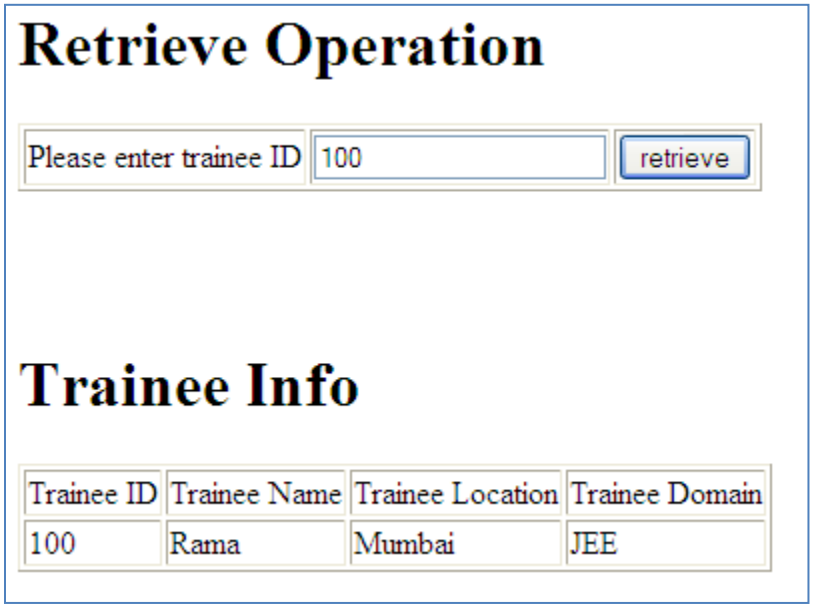


After making appropriate entry admin will click update button to reflect changes in Trainee table.

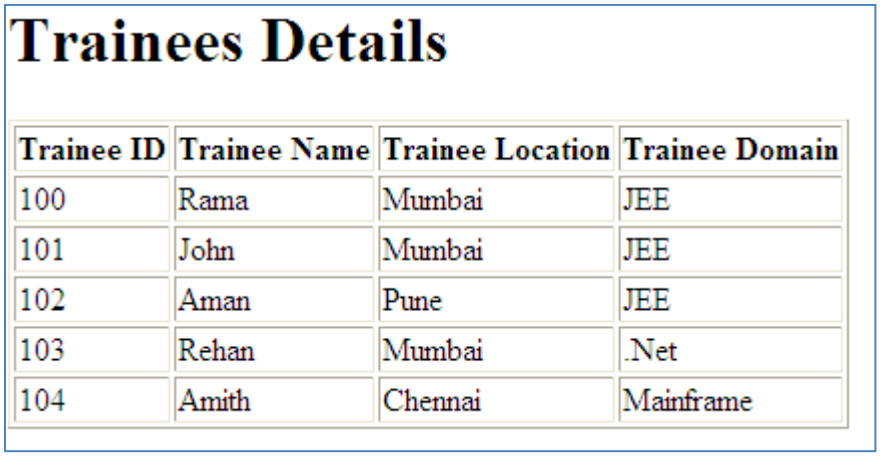
To implement **retrieve** operation on trainee info, refer the following list of screen shots.



When admin enters trainee id, trainee info will be displayed in the same page as shown below in the screen shot.



Admin selects '*retrieve all*' link to retrieve all trainee info as shown below in the screen shot



Note:

1. Perform appropriate validations on each field for all admin operations including login page
2. If any operation fails admin should be redirected to appropriate error page

Lab 3. Lab 3. Web Services (JAX-RS) and Spring REST

Goals	Understand the process of creating and Consuming a RESTful Java Web Service
Time	120 minutes

1. Refer below Java files:
   Product.java
   ProductDB.java

   Create a Product RESTful web service that will display all products to the Web service consumer.

   Note: Make use of the static DB given in ProductDB.java

   Refer below screen shot:

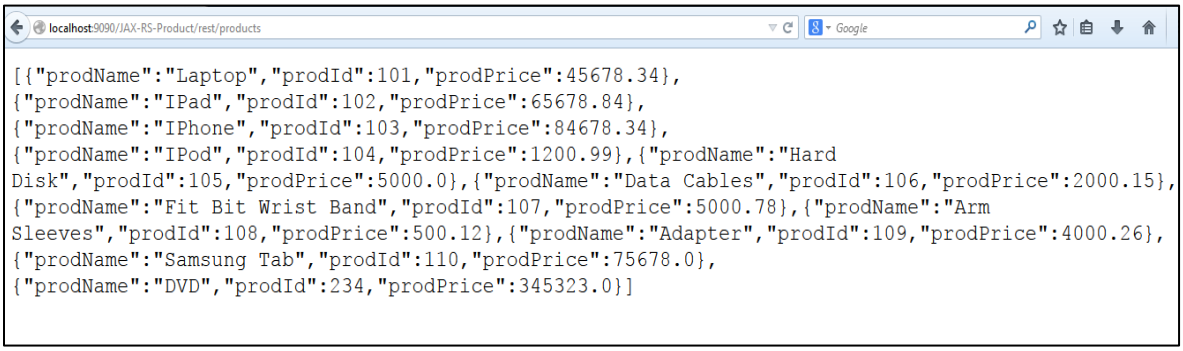


Figure 1 : Screenshot

The consumer should also be able to add a product to the existing list of products.

Refer below screen shots:

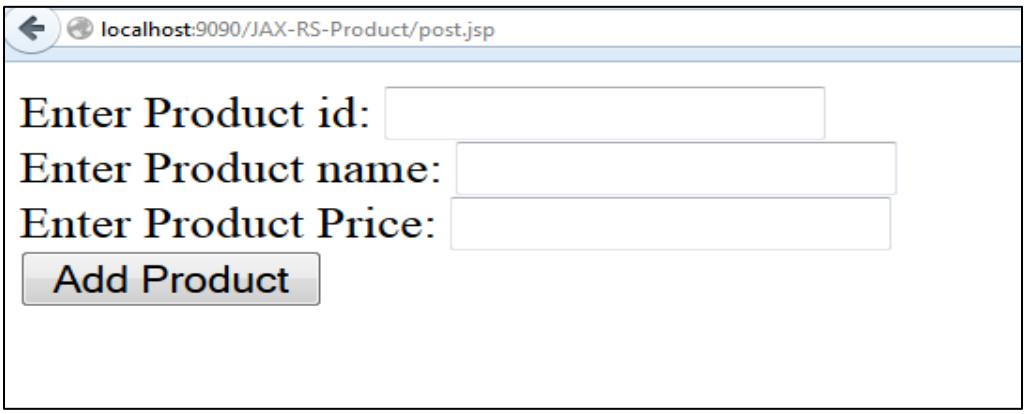


Figure 2 : Input Screen

After entering product details, refer below screen shot

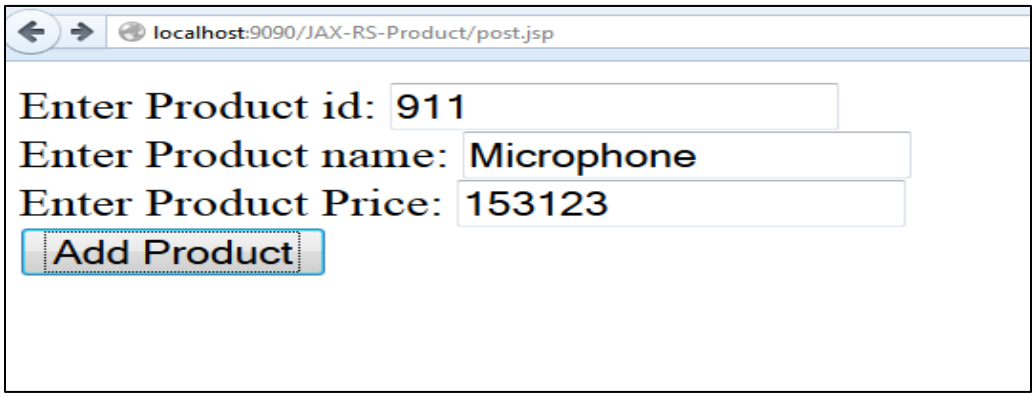


Figure 3 : Input Screen with data

Product is added to existing list of products

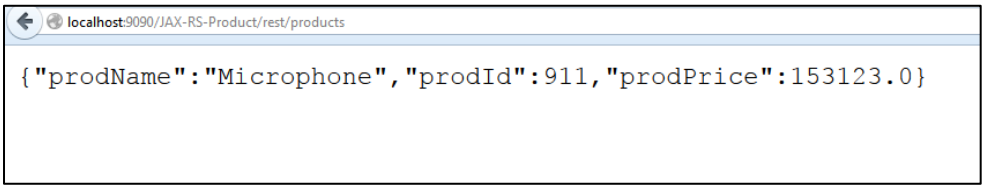


Figure 4 : Screen shot

Now, new product is added to existing list of products

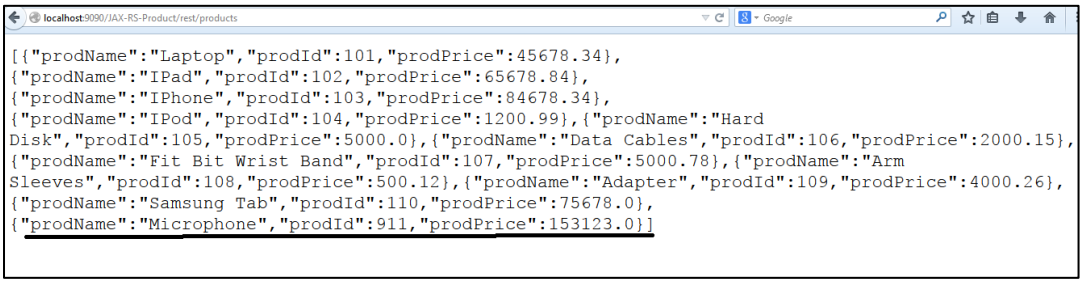
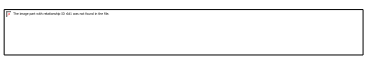


Figure 5 : Screen shot

2. Refer to the above assignment and create with Spring REST.



Lab 4. Spring REST: Exception Handling, Versioning and Pagination

Goals	Understand the process of creating and Consuming a RESTful Java Web Service
Time	90 Minutes

1. Refer to the Spring REST application created in Lab 2.2. Apply the Exception handling.
2. In Previous lab add the category property in the product class and add the versioning features as new product data will be available with V1 link.
   [Hint: Use URI Versioning]