Basic Spring 5.0

Copyright ©Capgemini Corporation (a part of Capegemini Group). All rights reserved. No part of this publication shall be reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior written permission of IGATE Corporation (a part of Capegemini Group).

IGATE Corporation (a part of Capegemini Group) considers information included in this document to be Confidential and Proprietary.

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

[Table of Contents 3](#_Toc47026532)

[Getting Started 4](#_Toc47026533)

[Overview 4](#_Toc47026534)

[Setup Checklist for Spring Framework 4](#_Toc47026535)

[Minimum System Requirements 4](#_Toc47026536)

[Creating the first Spring application: 4](#_Toc47026537)

[Lab 1. Injecting dependencies into a Spring application 6](#_Toc47026538)

[Lab 2. Spring Rest and Spring Data JPA 10](#_Toc47026539)

[Lab 3 . Spring Rest and Spring Data JPA Movie Application 16](#_Toc47026540)

[Appendix A: Table of Figures 21](#_Toc47026541)

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

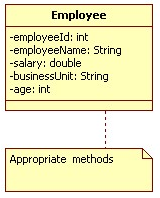
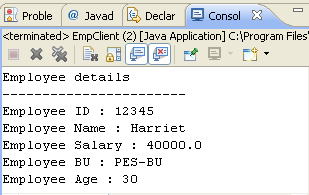
1. Injecting dependencies into a Spring application

|  |  |
| --- | --- |
| **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:

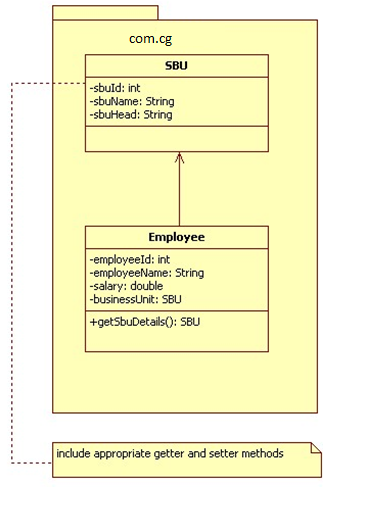


Class Diagram 1: Employee

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

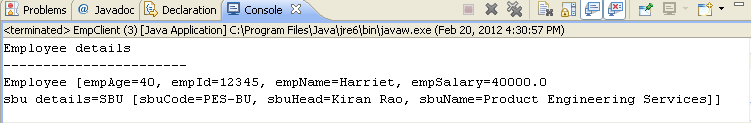
**Problem statement-1.2: Injecting dependencies**

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:



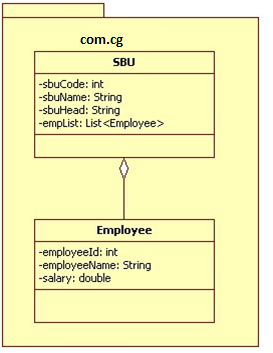
Class Diagram 2: **SBU and Employee**

The output would look as shown below:



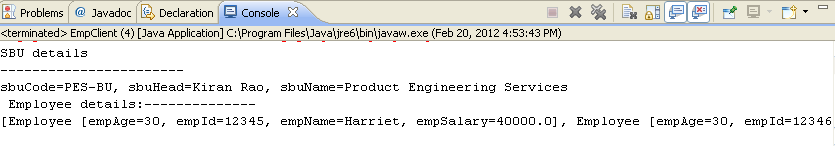
**Problem statement-1.3: Injecting dependencies**

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:

beans.xml

Collection

main()

EmployeeService

EmployeeDao

Employee

Flow Diagram - 1

**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.**

**Problem statement-1.5: Injecting dependencies**

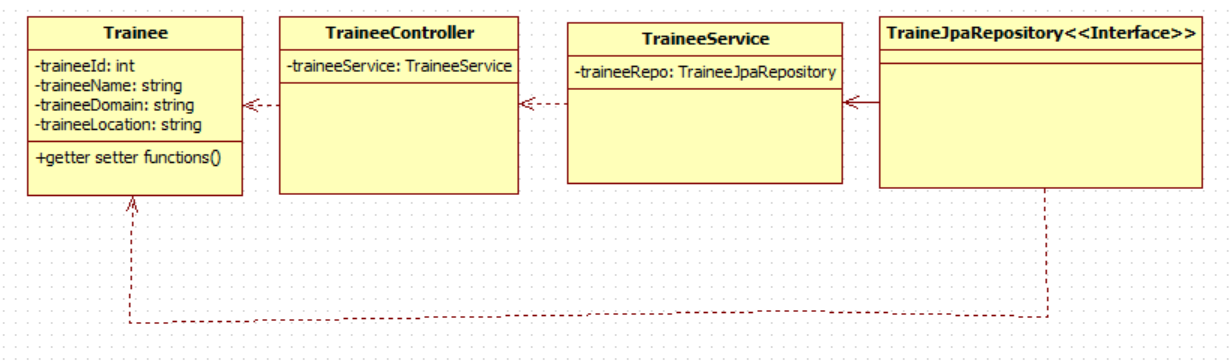
Implement the above lab using Javabase configuration and Spring Boot Features.

1. Spring Rest and Spring Data JPA

|  |  |
| --- | --- |
| **Goals** | * Demonstrate Spring's MVC framework * Integrate Spring and JPA |
| **Time** | 180 minutes |

**Version -1:**

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



Class Diagram 5**: Trainee related Classes**

Write end point function in the controller as follows.

* List All Trainee
* List Trainee by Name
* List Trainee By Id
* Delete Trainee Ny Id
* Update Trainee Information in the table based on different criteria
* Insert Trainee

**Refer below screen shot for reference**

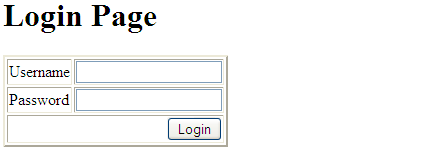
**Note Do not Create UI here.**

**This UI will be created in Module 3**

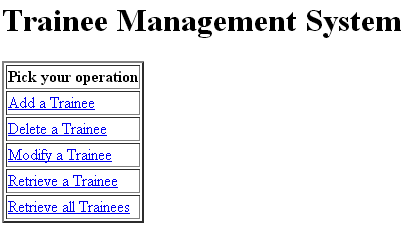
**Test your Spring Resfull web service using Postman**

* Following UI will be created in Angular in Module 3 for the above Trainee Application

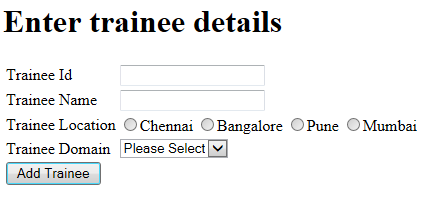
Initially when the application is deployed login page should come up as shown below.



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

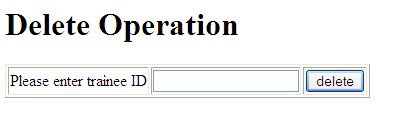


Admin select add hyperlink to add a new trainee. The following page shows up.



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

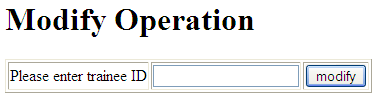
If admin selects delete operation refer the following screen shots to design your application.



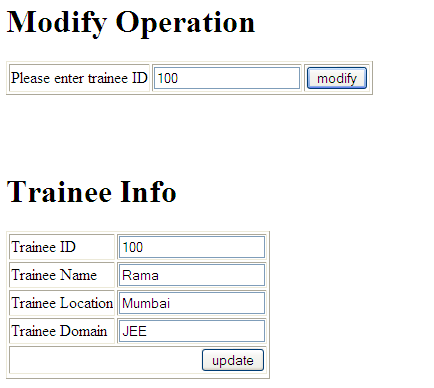
After entering trainee ID in the same page trainee info will be displayed as shown below



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

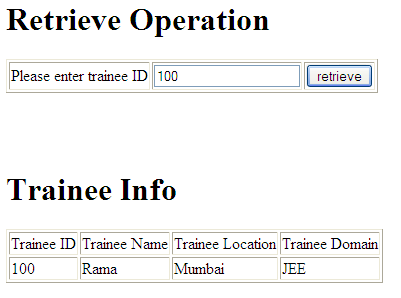


After making appropriate entry admin will click update button to reflect changes in Trainee table.

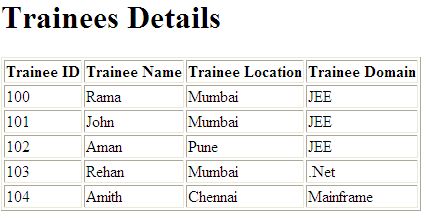
To implement **retrieve** operation on trainee info, refer the following list of screen shots.



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 . Spring Rest and Spring Data JPA Movie Application

5. Create a “Movie App” wherein movies could be maintained according to genres like Fiction, Drama and Satire. There is also a search option allowing to search for movies.

Use <Movies.java>

Create Spring Rest full application to perform following operation on Movie from database

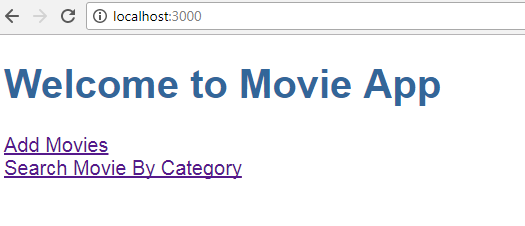
* + - Add Movie
    - Delete Movie
    - Search All Movies
    - Search Movie by Category
    - Update Movie Info

**Refer below screen shot for reference**

**Note Do not Create UI here.**

**This UI will be created in Module 3**

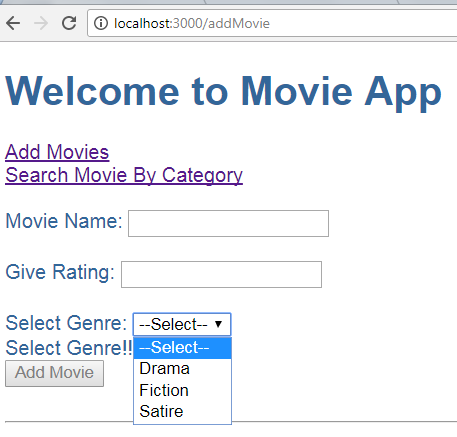
**Test your Spring Resfull web service using Postman**



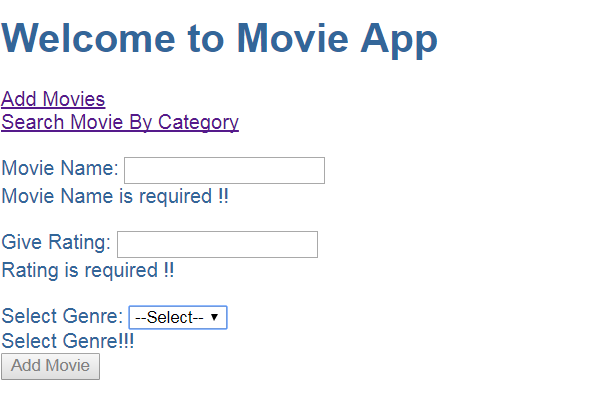
1. When the user clicks on “Add Movies” -> Allows a User to add a movie.

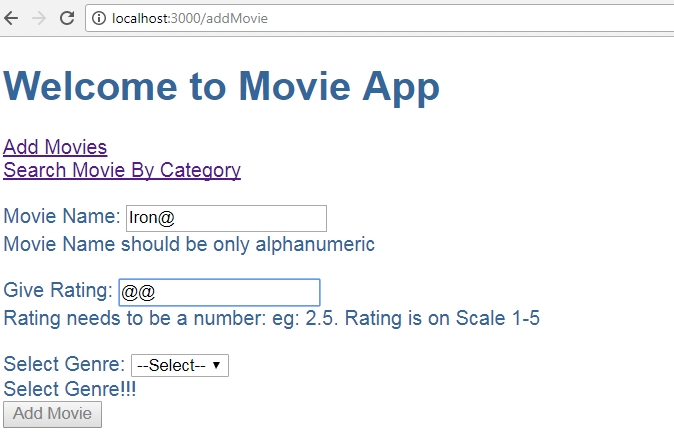
The user can add movie(s) depending on Genre (drop-down list loaded with values [Drama, Fiction, Satire])

The user could add multiple movies. On Adding Movies will be added in Database with the help of Spring Web services (Restful). Use Spring restful with JPA concept Refer below screen shot:



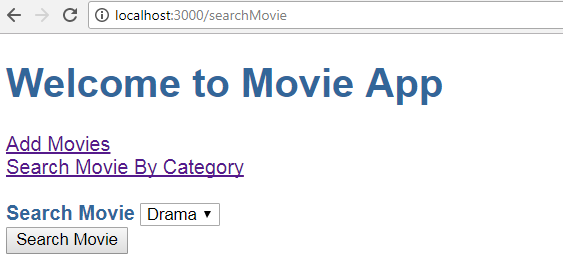
Refer below screens for validations:



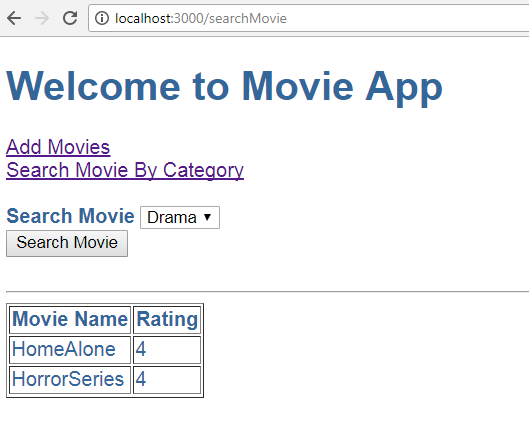


1. When the user clicks on “Search Movie By Category” -> Allows user to search movie(s) by Category. [Depending on Genre (Drama,Fiction,Satire)]

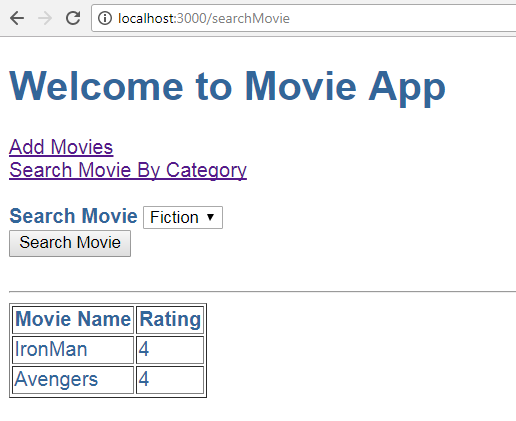
There could be more than one movies depending on same Genre. Refer below screen shots:



Thus if Searching Movie on ‘Drama’ displays below Result:



Thus if Searching Movie on ‘Fiction’ displays below Result:



**Note:**

1. In above assignment movies are added via “Add Movies” and Searched via “Search Movie By Category”. ***These are different components.***
2. Make use of Angular .
3. Make use of services.
4. Make use of Router Module. Which should be created as a separate module and included in the main App Module.
5. Use Spring restful & JPA concept to add movies & search movies.
6. All adding & searching movies done through database

## Appendix A: Table of Figures

No table of figures entries found.