

## Spring Assignments

### Assignment 1:

NOTE : Configure Bean with XML file.

Create a Bean class with properties of

- Student ID
- Student Name
- Map of Subjects and marks.

#### TASK 1:

- Create one Bean Configuration with Setter Injection of above properties.
- Create another Bean Configuration with Constructor Injection of above properties.
- Get Both Bean configuration Objects and print Details

### Assignment 2:

Create a Bean of Address with Properties of.

- City
- State
- Pincode

Create a Bean of Organization with Properties of.

- Organization name
- Address of Organization <Bean of Above Address>

Create a Bean of Employee with Properties of.

- Employee Name
- Employee Id
- Employee Salary
- Address of Employee <Bean of Above Address>
- Organization of Employee <Bean of Above Organization >

NOTE : Configure Beans with XML file.

**TASK 1 :** Now Create an application with Setter injection of all above dependencies

**TASK 2 :** Now Create another application with constructor injection of all above dependencies

### **Assignment 3:**

Create a Bean of Address with Properties of.

- City
- State
- Pincode

Create a Bean of Organization with Properties of.

- Organization name
- Address of Organization **<Bean of Above Address>**

Create a Bean of Employee with Properties of.

- Employee Name
- Employee Id
- Employee Salary
- Address of Employee **<Bean of Above Address>**
- Organization of Employee **<Bean of Above Organization >**

NOTE : Configure Beans with XML file.

#### **TASK 1:**

Now Configure above three beans with default values as followed below.

- Setter injection of all above dependencies
- Please enable autowire=no

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 2:**

Now Configure above three beans with default values as followed below.

- Setter injection of all above dependencies
- Please enable autowire=byName

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 3:**

Now Configure above three beans with default values as followed below.

- Setter injection of all above dependencies
- Please enable autowire=byType

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

**TASK 4:** Now Configure above three beans with default values as followed below.

- constructor injection of all above dependencies
- Please enable autowire=constructor

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **Assignment 4:**

**NOTE :** Configure Bean with Annotations. NO XML configurations.

Create a Bean class with properties of

- Student ID
- Student Name
- Map of Subjects and marks.

#### **TASK 1:**

- Create one Bean Configuration with Configuration class and Bean method.
- Create another Bean Configuration with component and component scans.
- Get Both Bean configuration Objects and print Details

#### **Assignment 5:**

Create a Bean of Address with Properties of.

- City
- State
- Pincode

Create a Bean of Organization with Properties of.

- Organization name
- Address of Organization **<Bean of Above Address>**

Create a Bean of Employee with Properties of.

- Employee Name
- Employee Id
- Employee Salary
- Address of Employee **<Bean of Above Address>**
- Organization of Employee **<Bean of Above Organization >**

**NOTE :** Configure Bean with Annotations. NO XML configurations.

**TASK 1 :** Now Create an application with Setter injection of all above dependencies

**TASK 2 :** Now Create another application with constructor injection of all above dependencies

### **Assignment 6:**

Create a Bean of Address with Properties of.

- City
- State
- Pincode

Create a Bean of Organization with Properties of.

- Organization name
- Address of Organization **<Bean of Above Address>**

Create a Bean of Employee with Properties of.

- Employee Name
- Employee Id
- Employee Salary
- Address of Employee **<Bean of Above Address>**
- Organization of Employee **<Bean of Above Organization >**

- NOTE : Configure Bean with Annotations. NO XML configurations.
- NOTE : Use only Configuration classes and Bean Methods
- NOTE : Don't Use Component

#### **TASK 1:**

Now Configure above three beans with default values as followed below.

- Use Setter injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 2:**

Now Configure above three beans with default values as followed below.

- Field injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 3:**

Now Configure above three beans with default values as followed below.

- constructor injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

### **Assignment 7:**

Create a Bean of Address with Properties of.

- City
- State
- Pincode

Create a Bean of Organization with Properties of.

- Organization name
- Address of Organization **<Bean of Above Address>**

Create a Bean of Employee with Properties of.

- Employee Name
- Employee Id
- Employee Salary
- Address of Employee **<Bean of Above Address>**
- Organization of Employee **<Bean of Above Organization >**

- NOTE : Configure Bean with Annotations. NO XML configurations.
- NOTE : Use only Component classes
- NOTE : Don't Use Configuration Classes and Bean Methods

#### **TASK 1:**

Now Configure above three beans with default values as followed below.

- Use Setter injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 2:**

Now Configure above three beans with default values as followed below.

- Field injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 3:**

Now Configure above three beans with default values as followed below.

- constructor injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

### **Assignment 8:**

Create a Bean of Address with Properties of.

- City
- State
- Pincode

Create a Bean of Organization with Properties of.

- Organization name
- Address of Organization **<Bean of Above Address>**

Create a Bean of Employee with Properties of.

- Employee Name
- Employee Id
- Employee Salary
- Address of Employee **<Bean of Above Address>**
- Organization of Employee **<Bean of Above Organization >**

- NOTE : Configure Bean with Annotations. NO XML configurations.
- NOTE : Use Component classes and minimum one Bean Method Configuration for every Bean class i.e. More than one Bean Configurations of every Bean class, so more than one Bean Objects of Bean class.

#### **TASK 1:**

Now Configure above three beans with default values as followed below.

- Use Setter injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 2:**

Now Configure above three beans with default values as followed below.

- Field injection of all above dependencies

Now get Employee Object and from that get all injected dependency Objects and print all details individually.

#### **TASK 3:**

Now Configure above three beans with default values as followed below.

- constructor injection of all above dependencies

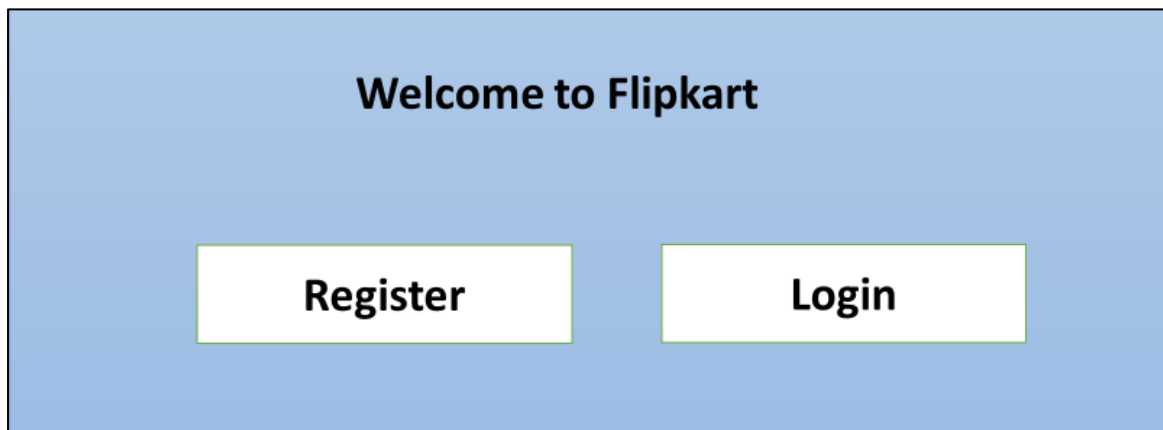
Now get Employee Object and from that get all injected dependency Objects and print all details individually.

# Web/MVC and JPA Module Assignments :

## Assignment 1 :

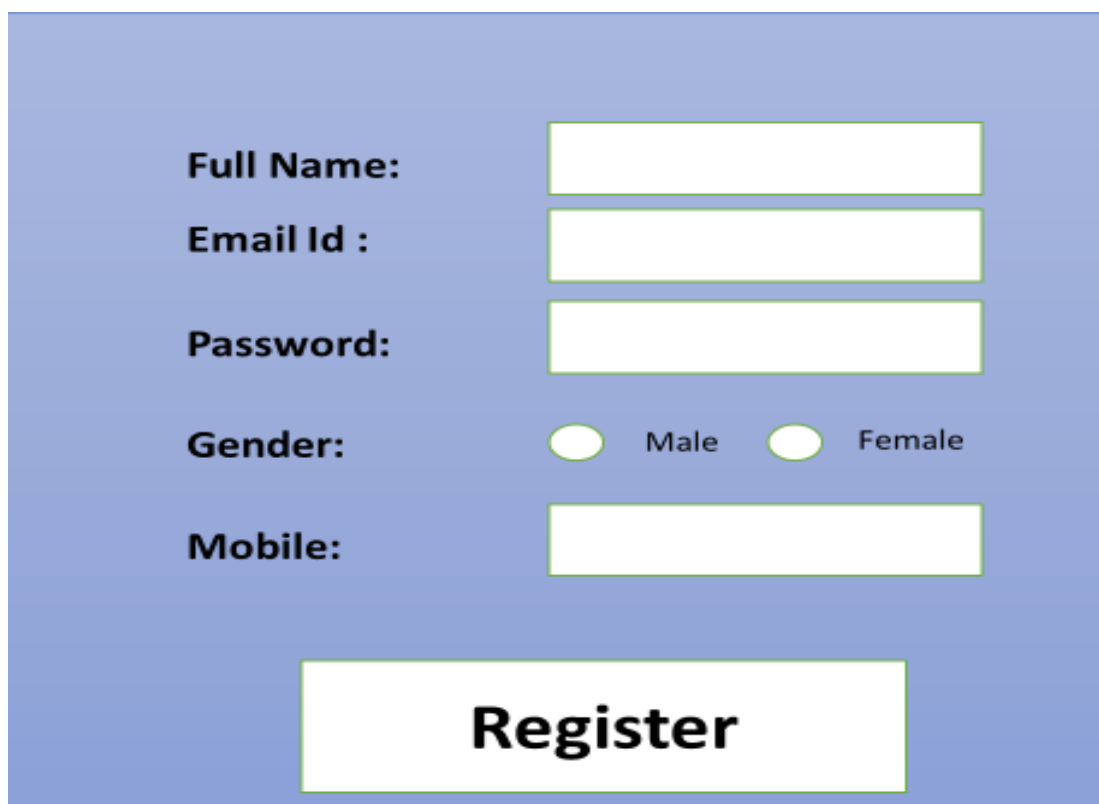
Create a web application with below functionalities.

- Create UI Pages as followed.



A light blue rectangular box representing a web page. At the top center, the text "Welcome to Flipkart" is displayed in a bold, black, sans-serif font. Below this text, there are two white rectangular buttons with black borders. The left button is labeled "Register" and the right button is labeled "Login", both in bold, black, sans-serif font.

- If User Clicks on Register, Display Below Form and ask User to Enter Details.



A light blue rectangular box representing a registration form. It contains several labels and input fields. On the left side, the labels "Full Name:", "Email Id :", "Password:", "Gender:", and "Mobile:" are listed vertically in bold, black, sans-serif font. To the right of each label is an input field: a white rectangle for "Full Name:", a white rectangle for "Email Id :", a white rectangle for "Password:", two radio buttons for "Gender:" (one labeled "Male" and one labeled "Female"), and a white rectangle for "Mobile:". At the bottom center of the box is a large white rectangular button with a black border, labeled "Register" in bold, black, sans-serif font.

- If User Clicks on Login, Display Below Form and ask User to Enter Login Details.

A login form with a light blue background. It contains two input fields: one for 'User Name' and one for 'Password'. Below these fields is a 'Login' button. The labels 'User Name:' and 'Password :' are in bold black text.

**Requirements:**

1. Create Application with Web and JPA Modules
2. Persist User Details in Database
3. Email Id is User Name.
4. Verify User is already available or not, If available Send Response as User Name already existed. If Not available, then proceed with User Registration.
5. When User Trying to login, if user Details exists then display home page with welcome message. If user details not available or matching, then display Invalid details message.

**Assignment 2:**

**NOTE: Create application with JPA module for database operations.**

**Define REST services for below Requirements:**

1. Create Single Order with below information. Please follow below JSON syntax

{

Order Id : Integer

Amount : Double

No Of Items : Integer

Items : String array []

Order status : String

Payment status :

User : {



```

        Email id:
        Name :
        Contact:
        City :
        Area :
    }
}

```

#### Requirements :

- Validate Request Body, like all properties are available in API payload and validate mandatory fields as well. – **Not required**
- If Order Id is available, then Response should be with valid Status code and Meaningful message.
- If Order Id is not available, then Order should be created and Response should be with valid Status code and Meaningful message.

## 2. Update Order with below information. Please follow below JSON syntax

```

{
    Amount : Double
    No Of Items : Integer
    Items : String array []
    Order status : Success/Failure
    Payment status : Success/Failure
    User : {
        Email id:
        Name :
        Contact:
        City :
        Area :
    }
}

```

}

**Requirements :**

- If Order Id is available, After data updating, then Response should be with valid Status code and Meaningful message.
- If Order Id is not available, Response should be with valid Status code and Meaningful message.

**3. Get Orders information with different scenarios.**

**Requirements :**

**NOTE:** Please use Response Status codes respectively.

- Get All Orders with user Id
- Get All Orders with user Id and order status
- Get All Orders with whose amount is > provided value
- Get All Orders from a city and area
  - i. city & area are required data
- Get All Orders from a city and area

**Note:** city & area are optional data

**Note :** Add Sorting for below Service.

- i. if city is passed and area not passed, orders should be loaded only for city value.
- ii. if city not passed and area passed, orders should be loaded only for area value.
- iii. if city and area both are passed, orders should be loaded only for city and area values.
- iv. If both are not passed, then all Orders should load irrespective of city and area values

**4. Delete Orders information with different scenarios.**

**Requirements :**

**NOTE:** Please use Response Status codes respectively.

- Delete Order with Order Id
- Delete All Orders with user Id
- Delete All Orders from a city and area

i. city & area are required data

- Delete All Orders from a city and area

**Note :** city & area are optional data

- i. if city is passed and area not passed, orders should be deleted only for city value.
- ii. if city not passed and area passed, orders should be deleted only for area value.
- iii. if city and area both are passed, orders should be deleted only for city and area values.
- iv. If both are not passed, then return Repone as either City or Area or both is required.