**Spring Web MVC**

-> It is one module in Spring Framework to develop web applications.

-> Web MVC module simplified web application development process.

1) Form Binding (form <---> java obj )

2) Flexibility in Form Binding (type conversion)

3) Multiple Presentation Technologies (JSP & Thymeleaf)

4) Form Tag Library (ready-made tags support)

Note: To develop web application using spring-boot we need to add below starter in pom.xml

**### spring-boot-starter-web ###**

-> The above starter provides support for below things

1) MVC based web applications

2) RESTFul Services

3) Embedded Container (Tomcat)

**Spring Web MVC Architecture**

1) DispatcherServlet

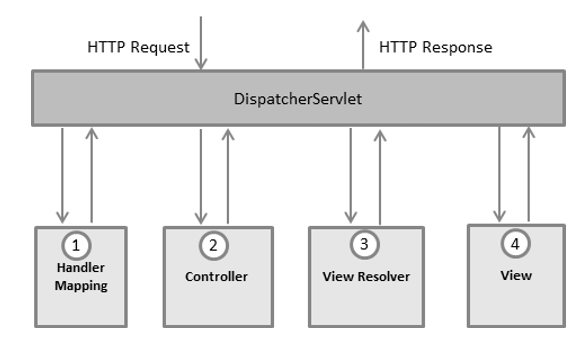
2) Handler Mapper

3) Controller

4) ModelAndView

5) ViewResolver

6) View



=> DispatcherServlet: Framework Servlet / Front Controller.

### Responsible to perform Pre-Processing and Post-Processing of request

=> Handler Mapper : Responsible to identify Request Handler class (controller)

=> Controller: Java class which is responsible to handle request & response

### Controller will return ModelAndView object to DispatcherServlet.

Model: Represents data in key-value format

View: Logical File Name

Note: Controllers are loosely coupled with Presentation technology.

=> ViewResolver: To identify presentation file location and technology

=> View: It is responsible to render Model data in view file.

**Building First Web App with Spring Boot**

1) Create Spring Starter Project with below dependencies

a) spring-boot-starter-web

b) spring-boot-devtools

c) tomcat-embed-jasper (mvnrepository.com)

2) Create controller class with required methods & map controller methods to URL pattern

3) Create View File with presentation logic

4) Configure View Resolver in application.properties file

5) Run the application and test it.

**Observations**

-> devtools dependency is used to restart our server when we make code changes.

-> To represent java class as controller we are using @Controller annotation

-> Controller methods we need to map with HTTP methods using unique URL pattern

GET --> @GetMapping

POST --> @PostMapping

-> Apache Tomcat is coming as default embedded container.

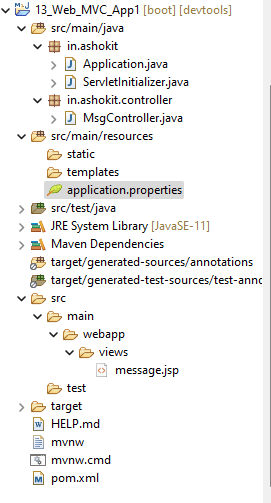
-> Embedded container port number is 8080. We can change that port number using application.properties file

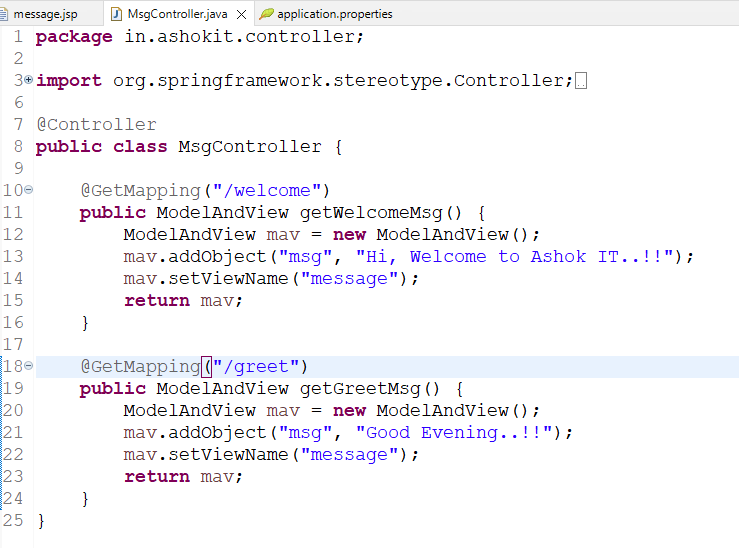
server.port = 9090

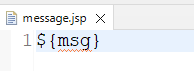
-> Spring Boot web apps will not have context path. We can add context-path using application.properties file.

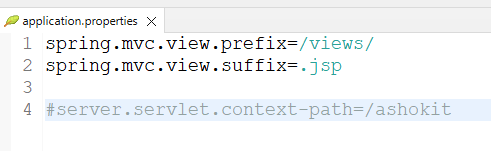
server.servlet.context-path=/ashokit

**Application Code**



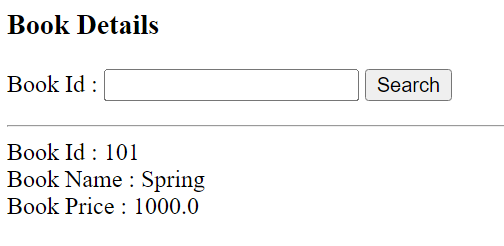


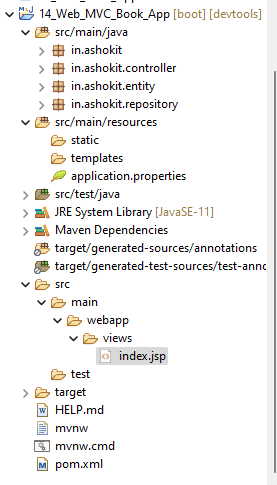




**02-WebApplication Requirement:**

**Retrieve book record based on given id and display in web page like below**

****



1) Create Spring Starter Project with below dependencies

a) web-starter

b) data-jpa

c) mysql-connector-j

d) lombok

e) devtools

f) tomcat-embed-jasper

2) Configure View Resolver & Data Source properties in application.properties file

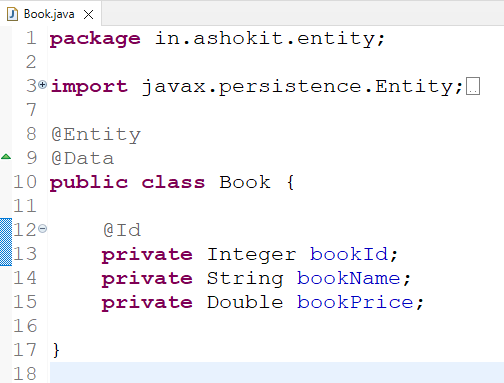
3) Create Entity class (table mapping)

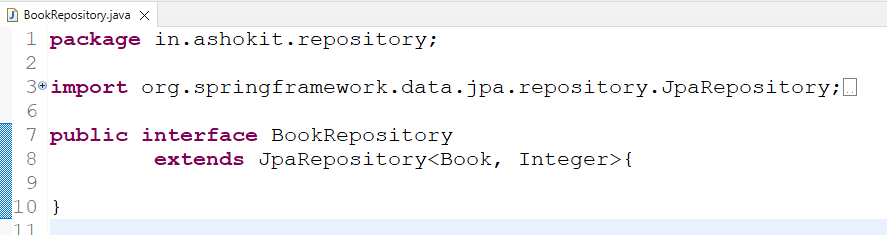
4) Create Jpa Repository interface

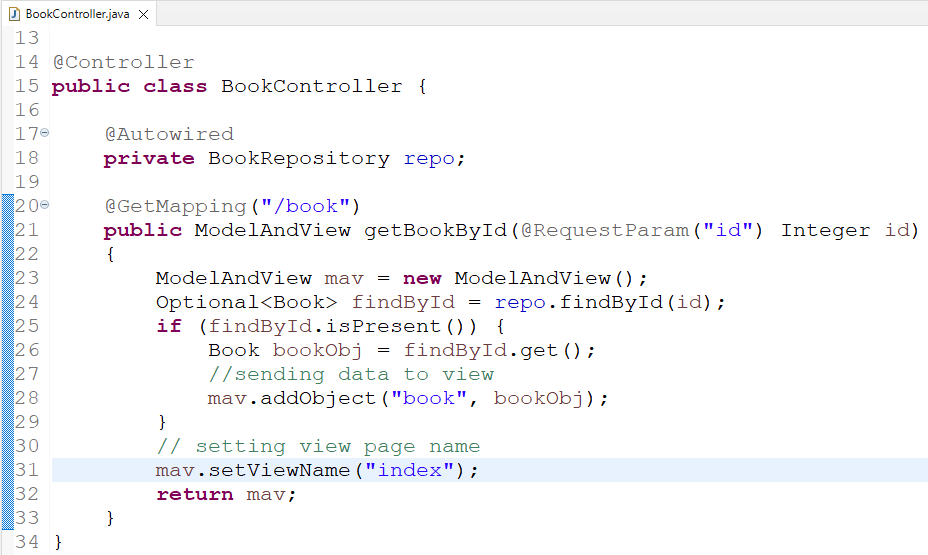
5) Create Controller class with methods to handle request & response

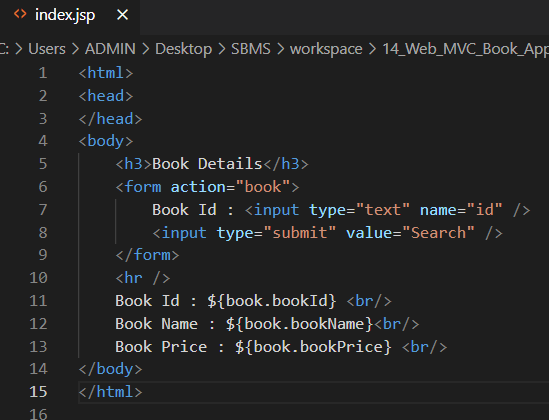
6) Create View Page

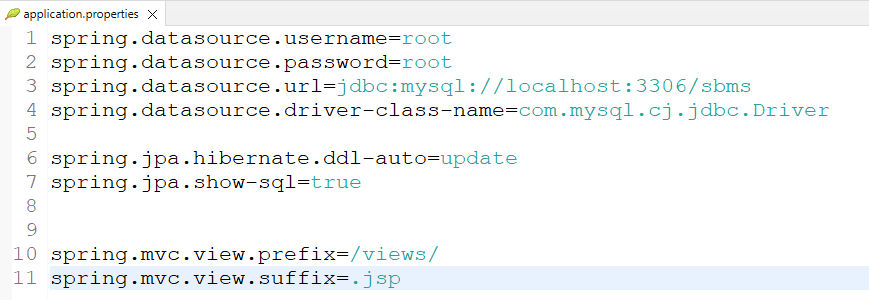
7) Run the application and test it

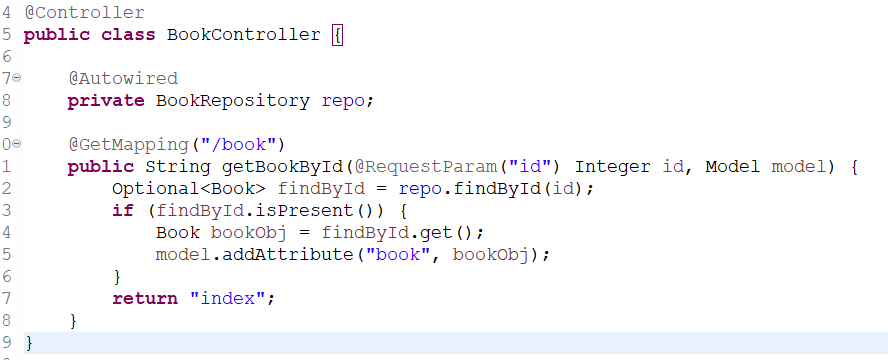




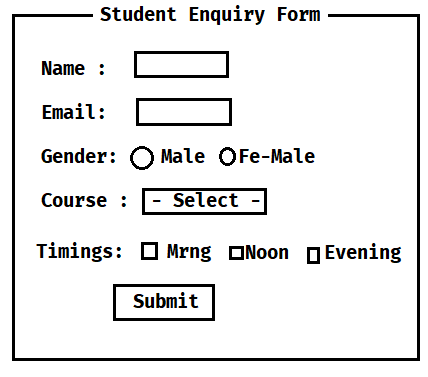








**03 – Web Application Requirement : Develop Student Enquiry Form like below**

****

**1) Course name drop down values should come from database table**

**2) Timings checkboxes options should come from database table**

**Note: When we click on submit button record should inserted into database table (STUDENT\_ENQUIRIES) and display success message on the same page.**

**Spring MVC Form Tag Library**

-> Predefined tags provided to simplify Forms development

-> To use Spring MVC form tag library in jsp we have to add below taglib url

<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

-> By using prefix we can access tags like below

1) <form:form>

2) <form:input>

3) <form:password>

4) <form:radioButton> & <form:radiobuttons>

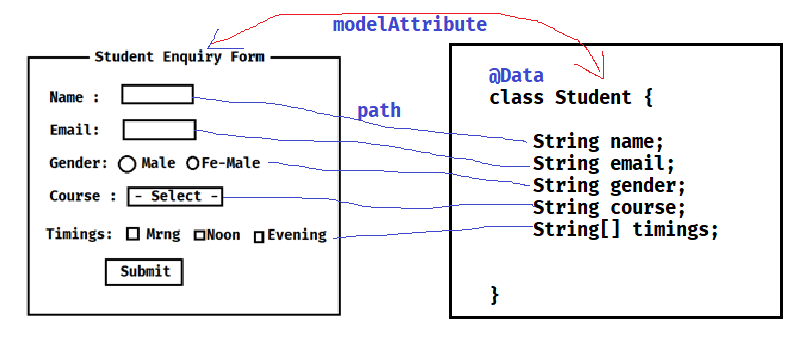
5) <form:select>

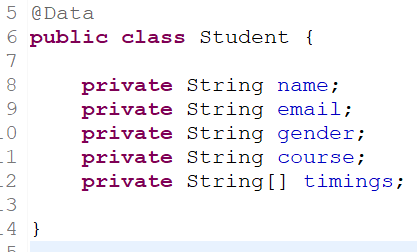
6) <form:option> & <form:options>

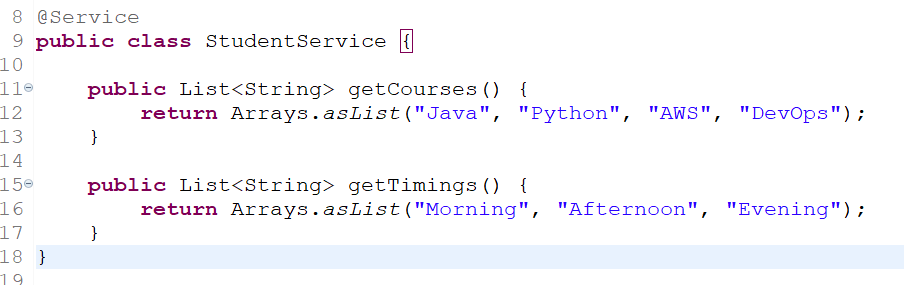
7) <form:checkbox> & <form:checkboxes>

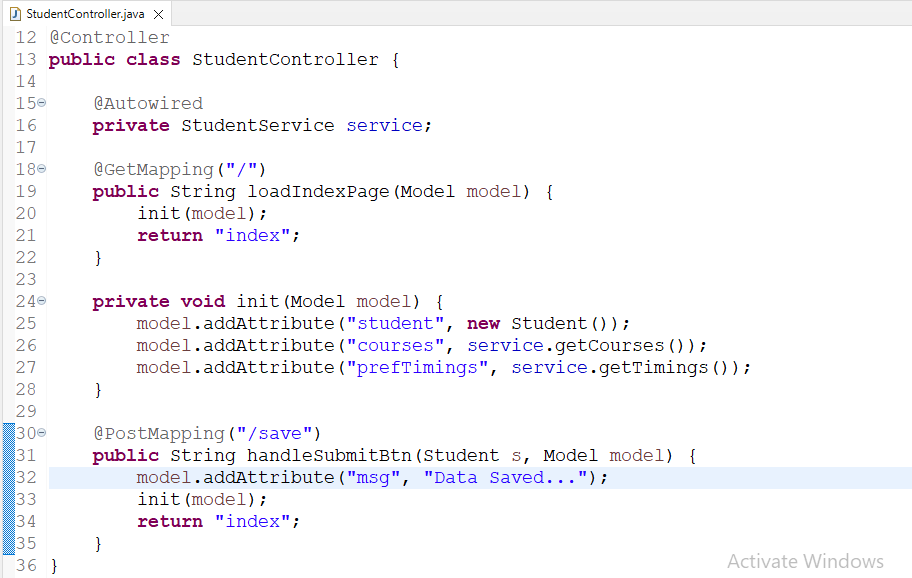
8) <form:hidden>

9) <form:textarea>









<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ taglib uri=*"http://www.springframework.org/tags/form"* prefix=*"form"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

</head>

<body>

<h2>Student Enquiry Form</h2>

<p>

<font color=*'green'*>${msg }</font>

</p>

<form:form action=*"save"* modelAttribute=*"student"* method=*"POST"*>

<table>

<tr>

<td>Name:</td>

<td><form:input path=*"name"* /></td>

</tr>

<tr>

<td>Email:</td>

<td><form:input path=*"email"* /></td>

</tr>

<tr>

<td>Gender:</td>

<td><form:radiobutton path=*"gender"* value=*"M"* />Male

<form:radiobutton path=*"gender"* value=*"F"* />Fe-Male

</td>

</tr>

<tr>

<td>Course</td>

<td><form:select path=*"course"*>

<form:option value=*""*>-Select-</form:option>

<form:options items=*"*${courses}*"* />

</form:select></td>

</tr>

<tr>

<td>Timings</td>

<td><form:checkboxes items=*"*${prefTimings}*"* path=*"timings"* /></td>

</tr>

<tr>

<td></td>

<td><input type=*"submit"* value=*"Save"* /></td>

</tr>

</table>

</form:form>

</body>

</html>

**Embedded Database**

**-> Embedded Database is called as In-Memory Database / Temporary database**

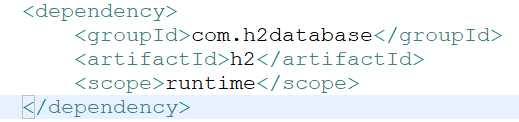
**Ex: H2 DB**

**-> When application starts H2 DB will start. When application stopped database will be stopped.**

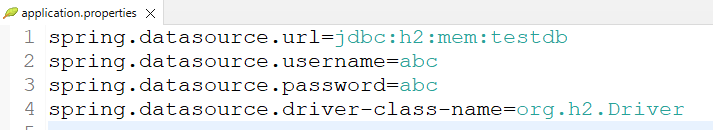
**Note: Embedded Databases are used for Proof Of Concept (POC) development.**

**How to use Embedded DB in Spring Boot**

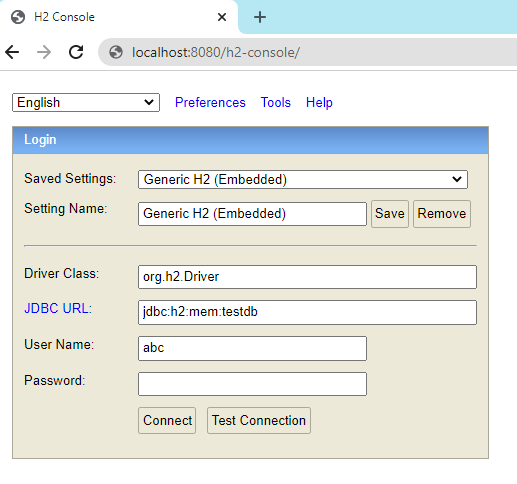
**-) Add H2 dependency in pom.xml file**

****

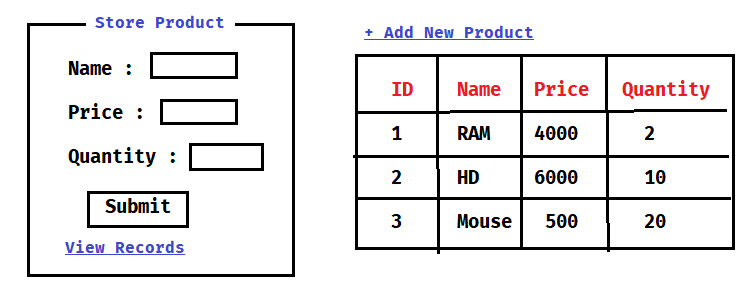
**2) Configure H2 DB data source properties in application.properties file**

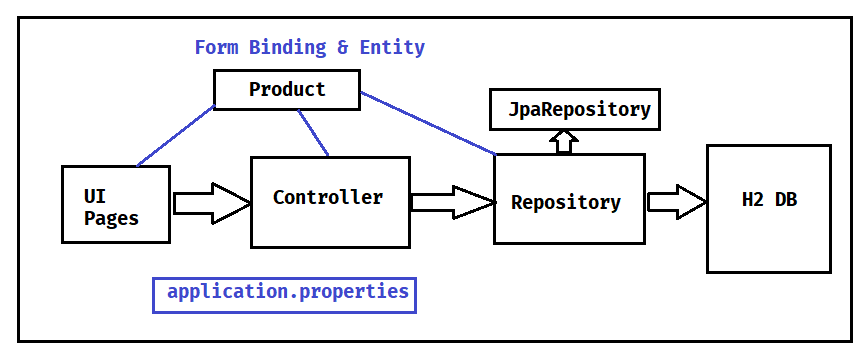
****

**3) Once application started we can access H2 DB console using below URL**

****

**Task**

****

****

**1) Create Spring Starter Project with below dependencies**

**a) web-starter**

**b) jpa-starter**

**c) h2 db**

**d) lombok**

**e) tomcat-embed-jasper**

**f) jstl**

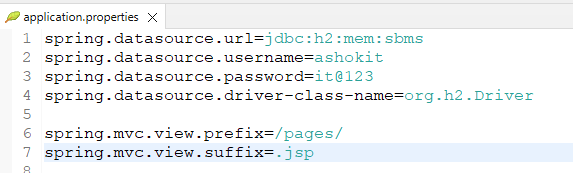
**2) Configure Data Source Properties and View Resolver Properties in application.properties file**

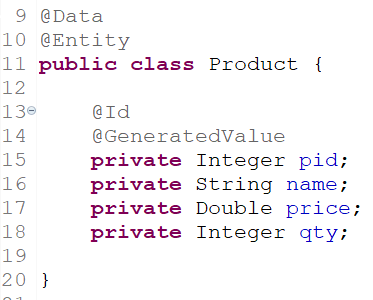
**3) Create Entity Class & Repository interface**

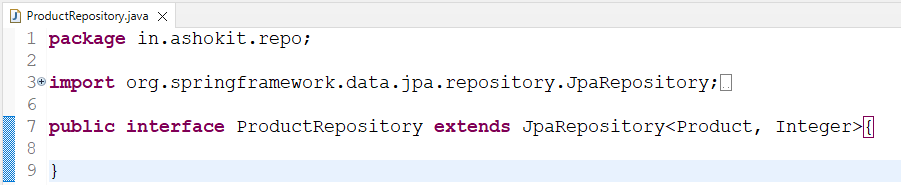
**4) Create Controller class with Required Methods**

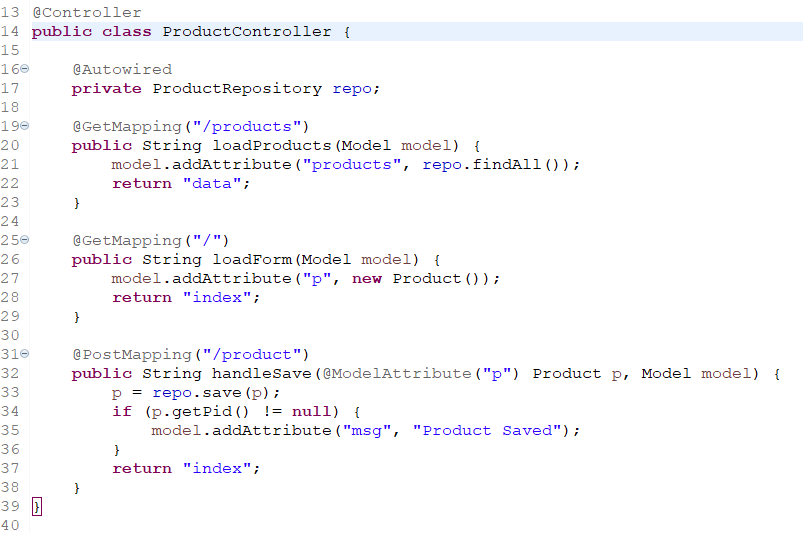
**5) Create View Pages**

**6) Run the application**

****





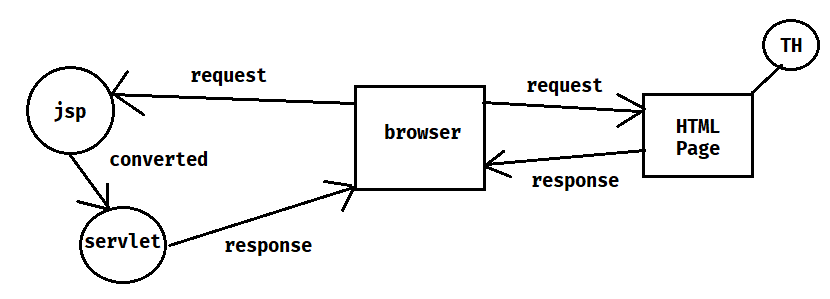


**Thymeleaf**

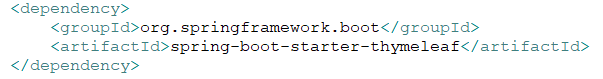
* **It is a template engine**
* **In spring web mvc we can use Thymeleaf as presentation technology**
* **We can use Thymeleaf as replacement for JSP**

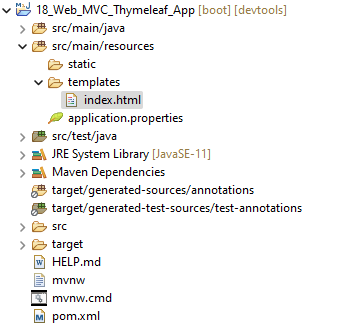
**Note: Every JSP page should be converted to Servlet to send response to browser hence performance wise JSP is slow.**

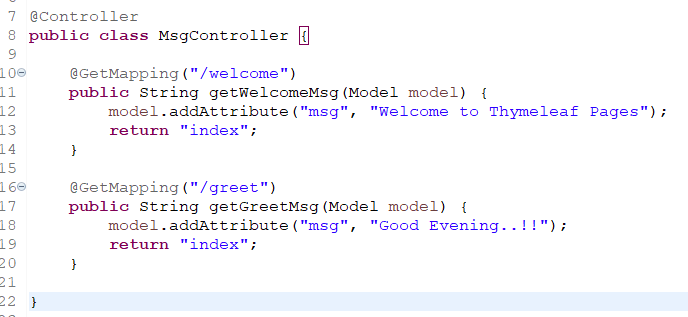
* **To overcome JSP problems we are using Thymeleaf for presentation layer development.**

****

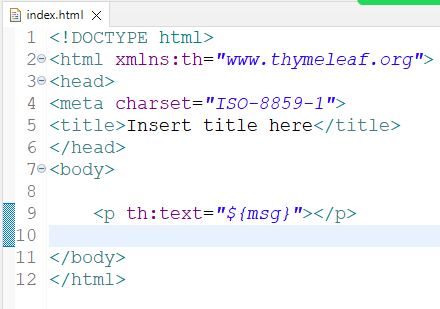
* **To work with Thymeleaf we need to add below starter in pom.xml file**

****

****

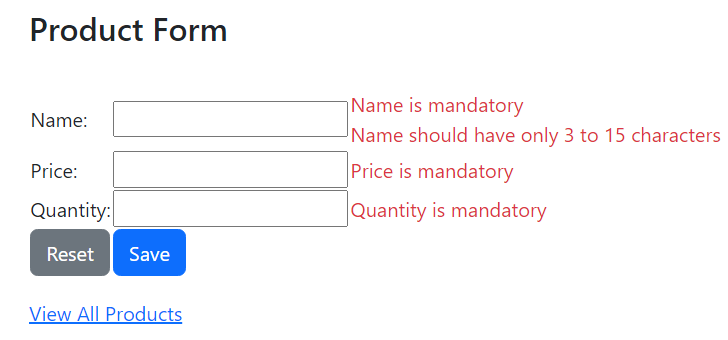
****

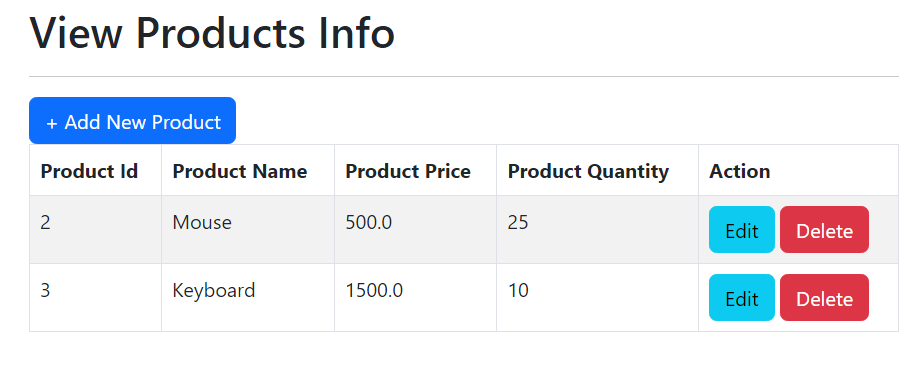
* **Create below HTML page under src/main/resources/templates folder.**

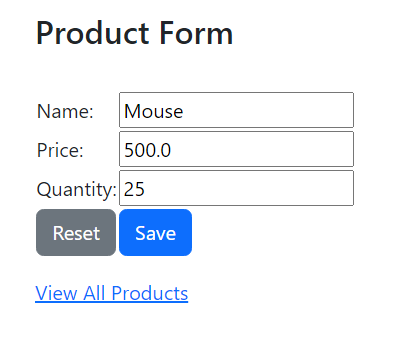
****

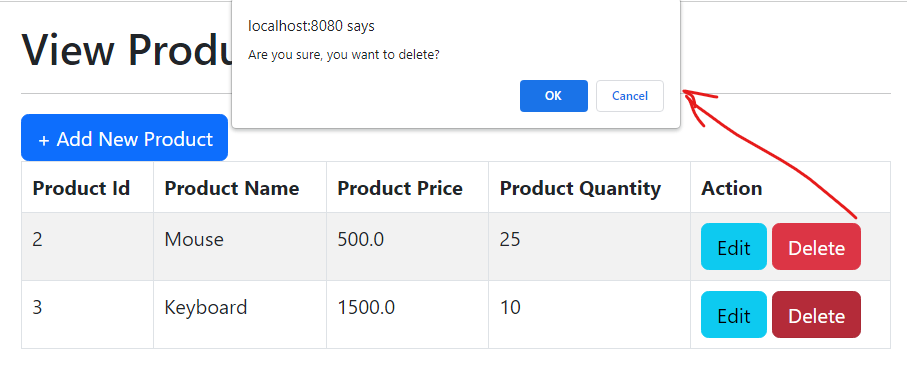
**Assignment: Develop Product application using Thymeleaf**

1. **Save Product**
2. **View Products**
3. **Edit and Update Product**
4. **Delete Product**

****

****

****

****

**1) Create Spring Starter Project with below dependencies**

**a) web-starter**

**b) thymeleaf-starter**

**c) data-jpa-starter**

**d) h2 driver**

**e) lombok**

**f) validation-starter**

**g) devtools**

**2) Configure Datasource properties in application.properties file**

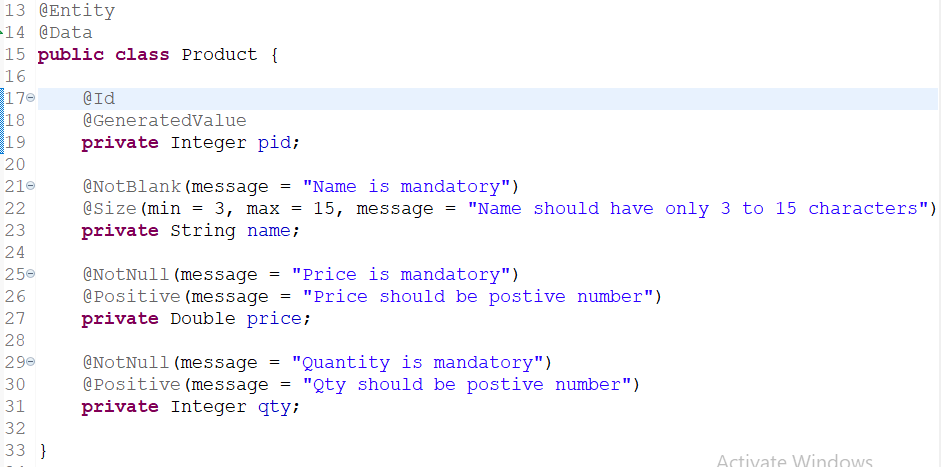
**3) Create Entity class & Repository interface**

**4) Create Controller class with required methods**

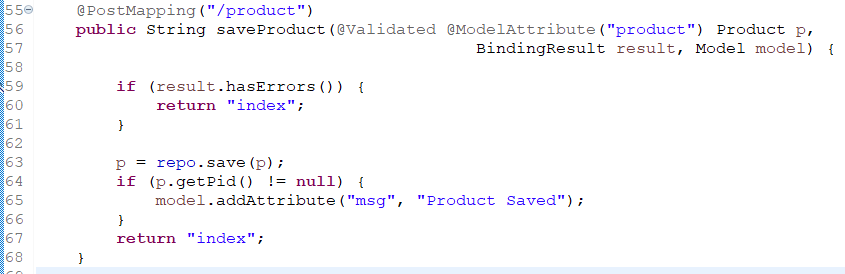
**5) Create View Files using Thymeleaf and Bootstrap**

**Implementing Form Validations**

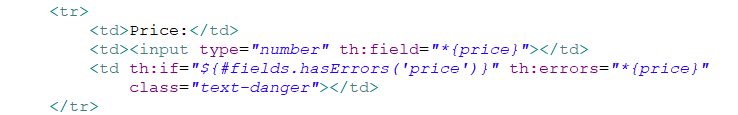
1. **Write Validation rules using annotations in binding class like below**



1. **Make changes to controller method to valid form data. If form validations are failed then return same page**

****

1. **Print Validation message in the form for every field like below**

****