**What this project does??**

Note that this is third project in the series of Spring MVC which explains functioning of controller layer and service layer( it does not include DAO layers and any database operation). Please go through projects ' SpringMVCApplicationFlow' which is first project in Spring MVC Series. It does not contains any web related operations, it explains functionality of 'spring-servlet.xml' in Spring MVC context and ' SpringMVCHelloWorld' which explains how controller layer functions in Spring MVC.

Thus

' SpringMVCApplicationFlow': explains functionality of 'spring-servlet.xml' in Spring MVC context

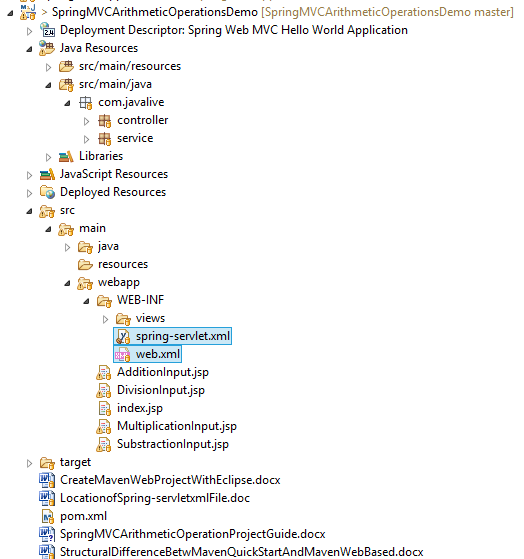
' SpringMVCHelloWorld': explains how controller layer functions in Spring MVC

'SprinArithematicOperation': explains how controller layer and service layer functions in Spring MVC

This project explains basic functioning of Spring MVC. How controller acts and handle request, makes call to service layer which contains business logic, collects the result of business logic from service layer and render the specified jsp accordingly.

This is web based maven project.

**Note that this is xml based spring MVC project. We have web config file web.xml and spring configuration file spring-servlet.xml. (Location of these files are shown in screen shot of project structure below.)**

****

**Steps to create project:-**

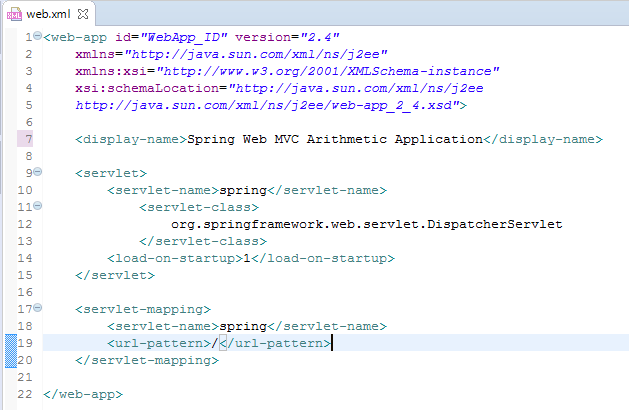
1. This is web based maven project. Create webapp maven project. Please refer the file 'CreateWebBasedMavenProjectInEclipse.docx' in this project to know more about creation of web based maven project.

2. Add require dependencies for spring as shown in pom.xml file.

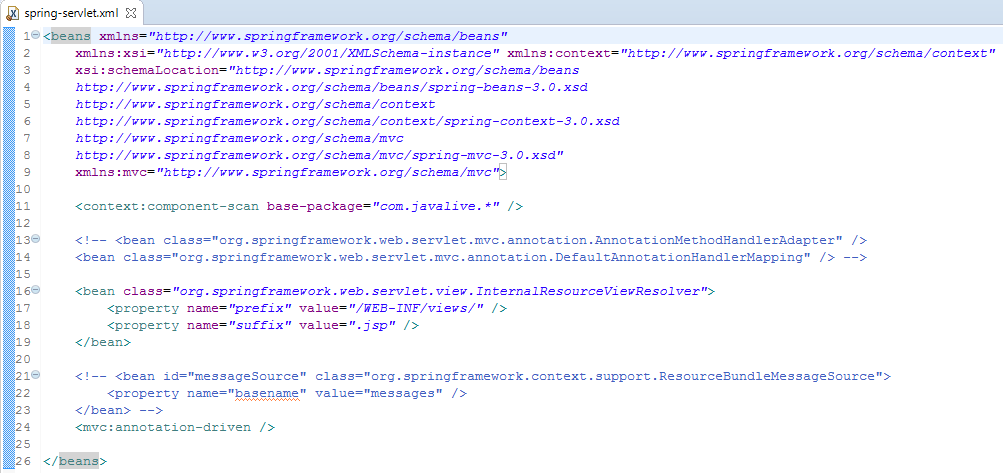
3. Create required packages and add the files.

**Functioning of the project**

1. web.xml where we have mentioned the spring functionality as DispatcherServlet.



2. spring-servlet.xml where we have defined bean 'org.springframework.web.servlet.view.InternalResourceViewResolver' which is responsible for rendering specified page mentioned in controller file.



Please study the role of <context: component-scan> and <mvc:annotation-driven> tags very very carefully. (Refer project 'SpringMVCApplicationFlow' for this purpose.)

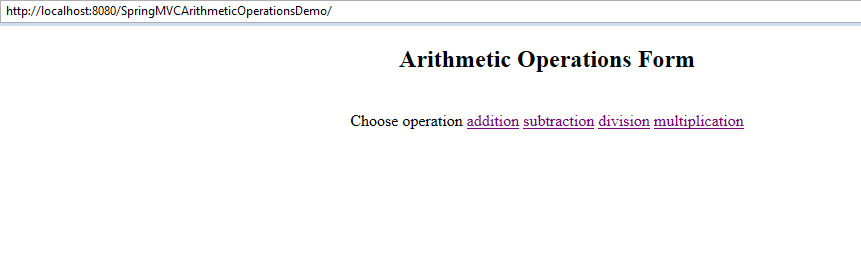
**3. com.javalive.controller. ArithmeticController.java is controller layer file of this application. Please go through annotation @RequstMapping and @RequestParam and parameter ModelMap very carefully. For this purpose refer following link:** [**https://dzone.com/articles/using-the-spring-requestmapping-annotation**](https://dzone.com/articles/using-the-spring-requestmapping-annotation)**.**

4. com.javalive.service.ArithmeticService.java is a service layer file which contains business logic and call to DAO layer(which is not done in this project as it contains only numeric operations

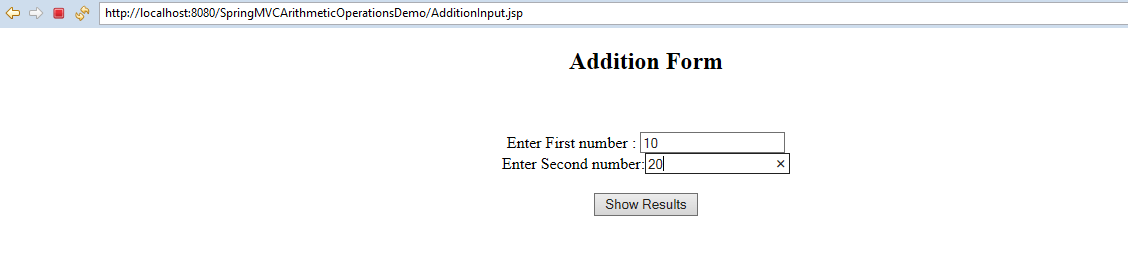
**Functioning:**

**Before move on please study functioning of spring-servlet.xml, how beans get loaded at runtime and how component scan works.(@Controller, @Service are subtype of @Component)**

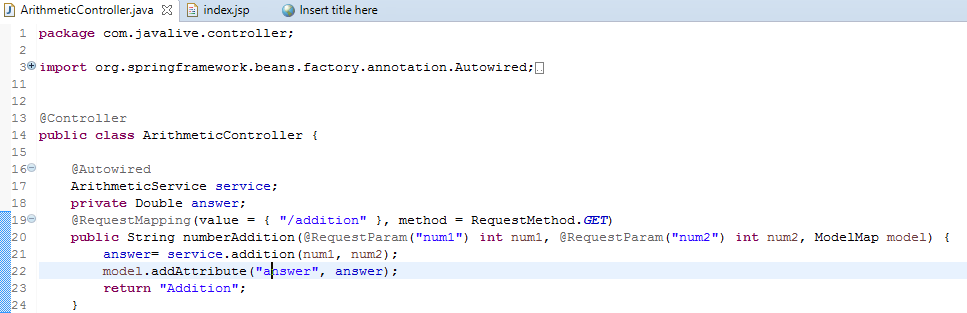
**index.jsp is welcome file of this project which contains links to various pages for different numeric operations.**

****

**When we select particular link respective .jsp page get loaded.**

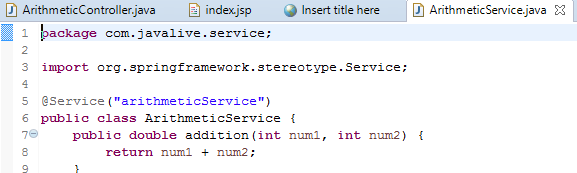
****

**This page accepts two numbers and after clicking the submit button it call respective action in controller.**

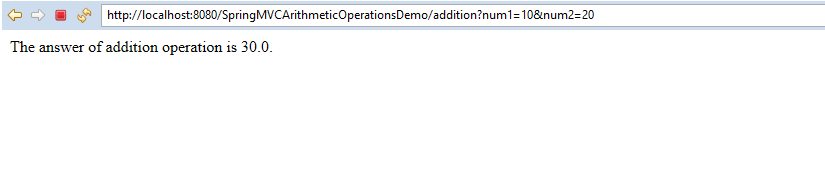
****

**Note here @Autowired annotation does dependency injection of ArithmeticService.**

**Controller then call the respective business method of service layer i.e. ArithmeticService.java in this case,**

****

**collects the result and then renders particular .jsp page with this result.**

****

Thus this project includes functionality of two layers of Spring MVC viz. controller and service layer.

Spring MVC contains one more layer viz. DAO i.e. Data Access Object which we will study in upcoming projects like 'SpringMVCHibernateWithXMLFull' and 'SpringHibernateExample'.

So please go through this project and try to understand flow of request in Spring MVC and role of controller and service layer very neatly.