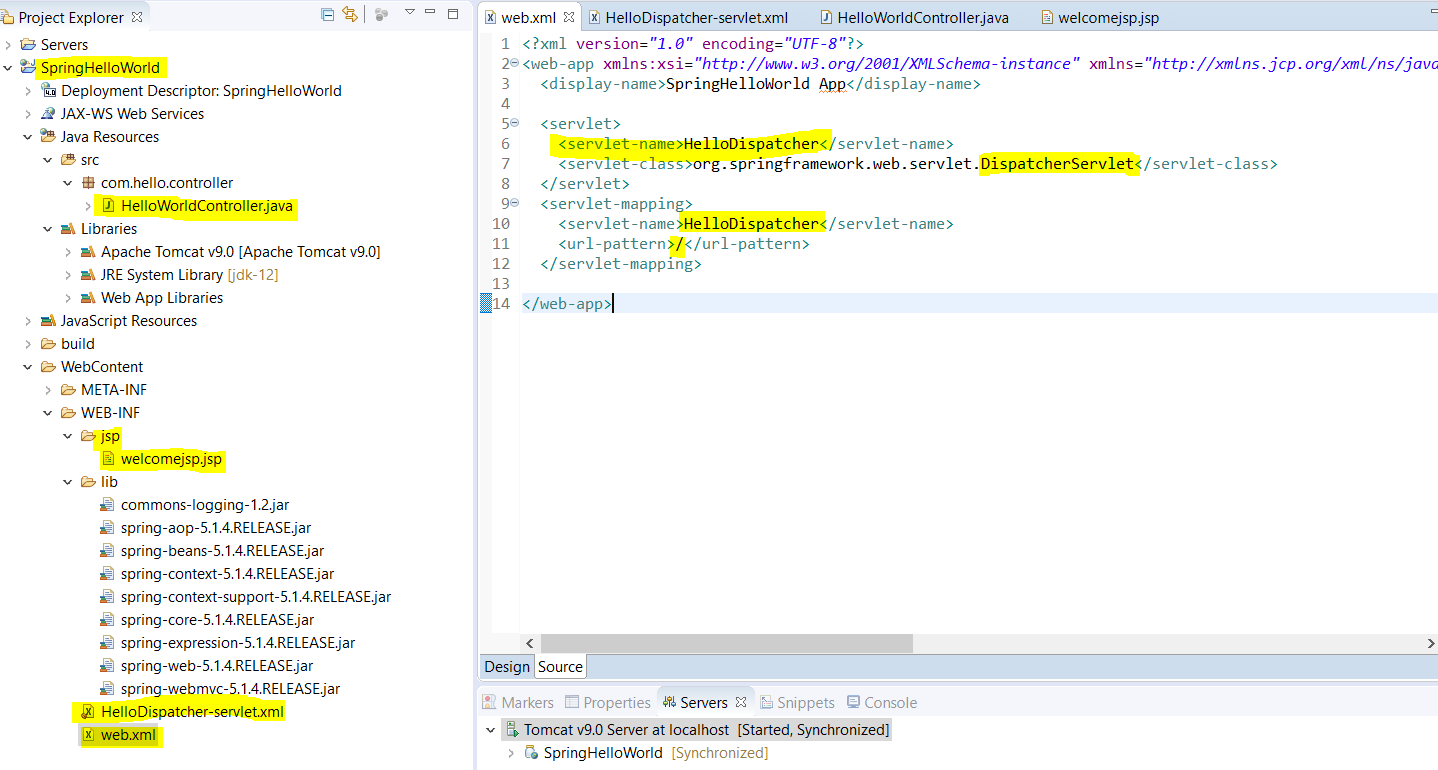
**Spring MVC:** It helps in building loosely coupled web application. It helps in separating Business Logic, Presentation Logic and Navigation logic:

**MODEL**: Responsible for encapsulating application data.

**VIEW**: The view renders response to the user with the help of Model Objects.

**CONTROLLERS**: Responsible for receiving the request form the users and calling the backend services.



|  |
| --- |
| **Web.xml:**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>SpringHelloWorld App</display-name>    <servlet>  <servlet-name>HelloDispatcher</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>HelloDispatcher</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    </web-app> |

**In web.xml we have done fallowing configuration.**

1. Created HelloDispatcher servlet name.
2. Configure the front controller **DispatcherServlet class** under <servlet-class> tag
3. Define the URL-pattern
4. Now next step is we will create one more xml file with the naming conventions (syntax is: **HelloDispatcher-servlet.xml)**

**HelloDispatcher-servlet.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>  <context:component-scan base-package = *"com.hello.controller"* />  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name = *"prefix"* value = *"/WEB-INF/jsp/"* />  <property name = *"suffix"* value = *".jsp"* />  </bean>    </beans> |

1. Configure the base-package name to be scanned by the controller for getting particular controller class.
2. Configure *InternalResourceViewResolver* for view and under this bean tag we have configured the location as prefix and suffix as type/extenuation of the view page (.jsp).

**HelloWorldController.java**

|  |
| --- |
| **package** com.hello.controller;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.servlet.ModelAndView;  @Controller  @RequestMapping (value = "/")  **public** **class** HelloWorldController {  //@RequestMapping (method = RequestMethod.***GET***)  @GetMapping  **public** ModelAndView welcome () {    **return** **new** ModelAndView ("welcomejsp", "welcomeMessages", "Welcome to Spring HelloWorld");  }  } |

1. Created one class HelloWorldController and mentioned with stereotype annotation **@Controller**
2. **@RequestMapping** (value = "/") Define the entry point URL-pattern.
3. Created on method with return type ModelAndView and method (**method = RequestMethod.*GET) OR use* @GetMapping** for calling default method GET.
4. Here first argument is for view page (jsp page), second argument is being used in jsp for **${welcomeMessages}** for displaying message “Welcome to Spring HelloWorld”

**welcomejsp.jsp**

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h2>${welcomeMessages}</h2>  </body>  </html> |

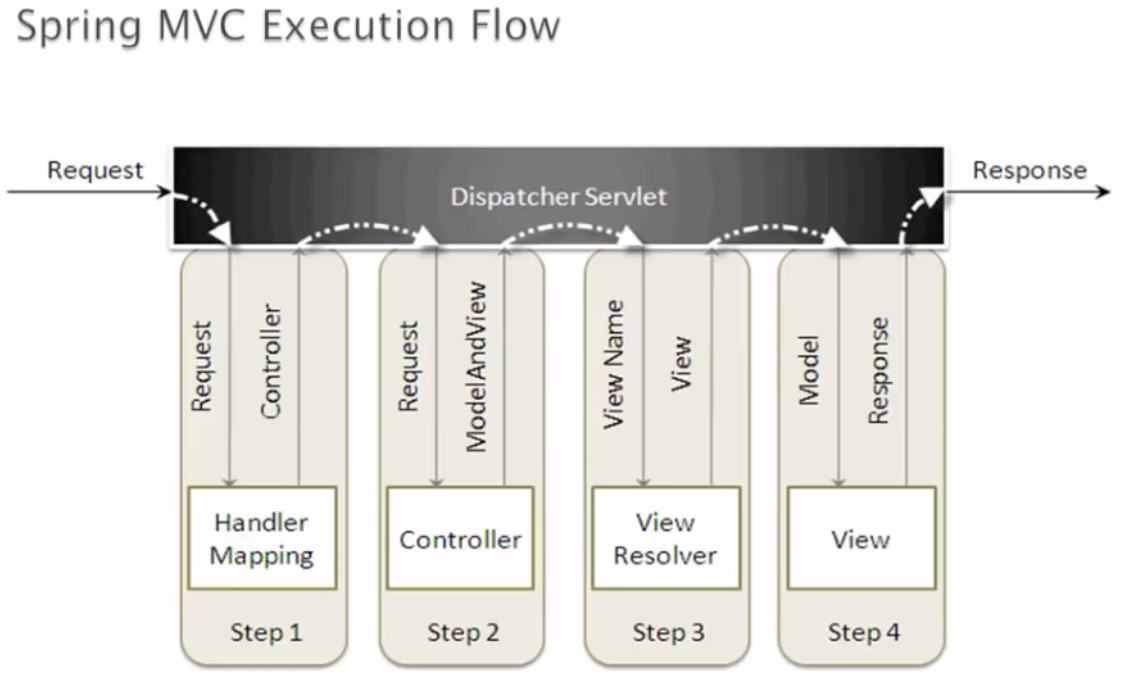
## Now run the project using URL (<http://localhost:8080/SpringHelloWorld/>) and get output: Welcome to Spring HelloWorld!!!

So, in the above project we have fallowing main files.

1. Web.xml deployment descriptor file having front controller configuration.
2. Spring configuration file having base-package, view resolver, location and extenuation of the file.
3. Controller class which receiving the request form the users and calling the backend services.
4. JSP page for output rendering.
5. So far, we have used fallowing annotations. (
6. **@Controller**
7. **@RequestMapping (value = "/")**
8. **@RequestMapping (method = RequestMethod.GET)**
9. **@GetMapping**
10. **<context: component-scan base-package = *"com. hello. Controller"* />**

**Q: When hit the URL** [**http://localhost:8080/SpringHelloWorld/**](http://localhost:8080/SpringHelloWorld/) **then what is the execution flow?**

**A:**



**As we already know that**

**Spring MVC:** It helps in building loosely coupled web application. It helps in separating Business Logic, Presentation Logic and Navigation logic:

**MODEL**: Responsible for encapsulating application data.

**VIEW**: The view renders response to the user with the help of Model Objects.

**CONTROLLERS**: Responsible for receiving the request form the users and calling the backend services.

In the above diagram we can see that when a http request sent to spring MVC framework the fallowing event happens

1. The dispatcher servlet (Front controller) will revive the request first and consult with handler mapping.
2. The handler mapping will pass this request to the particular controller class.
3. The controller will process the request by calling the appropriate service and returns ModelAndView object to the dispatcher servlet.
4. The dispatcher servlet sends the view name from ModelAndView object to the View Resolver to find the actual view. Like in the previous example in spring configuration file we have set prefix and suffix. Prefix will take view name[welcomejsp] from the mentioned location [*"/WEB-INF/jsp/"] and then* add suffix [.jsp] with prefix. Finally, we will get final view name [welcomejsp.jsp]
5. Once view gets resolved then view will be rendered to the user with the help of model.

**Q: What is Handler mapping:**

The **BeanNameUrlHandlerMapping** class maps URL requests to beans **names**. It is the default **handler mapping** class, so it is the one created by the DispatcherServlet when **Spring** cannot find any **handler mapping** class declared.

**Q: What is ModelAndView object?**

**Q: What does** <context: component-scan base-package = *"com. hello. Controller"* />?

**Note: if we don’t want fallow naming convention for putting spring configuration file name then we have to do some changes as fallows.**

**Customizing spring configuration file name using Init-param**

As we have seen that we have to fallow naming convention to create spring configuration file (**ServletName-servlet.xml**). But we can customize the spring configuration file name according to our choice.

Let’s say if we have more than one spring configuration file then in that case if we fallow the naming convention to create file then container will get ambiguity that is why here in this case customized spring configuration file name is required.

1. Create our own custom spring configuration file.
2. Configure these custom spring configuration file in web.xml under <init-param> tag.
3. Inside this tag we have to mention <param-name> **contextConfigLocation** (This name is case sensitive).
4. Then mention the custom spring configuration file name with absolute path.
5. In the below web.xml we have created two custom spring configuration files under WEF-INF
6. **/WEB-INF/MyCustomeSpringConfig.xml, /WEB-INF/Beans.xml**
7. **In MyCustomeSpringConfig.xml file we have configured prefix and suffix (jsp file location and file type)**
8. **In Beans.xml we have configured all the package name to be scanned by container.**

|  |
| --- |
| **Web.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>SpringCustomeConfigName</display-name>  <servlet>  <servlet-name>HelloDispatcher</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  **<init-param>**  **<param-name>contextConfigLocation</param-name>**  **<param-value>/WEB-INF/MyCustomeSpringConfig.xml, /WEB-INF/Beans.xml</param-value>**  **</init-param>**  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>HelloDispatcher</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

**MyCustomeSpringConfig.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name = *"prefix"* value = *"/WEB-INF/jsp/"* />  <property name = *"suffix"* value = *".jsp"* />  </bean>  </beans> |

**Beans.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>  <context:component-scan base-package = *"com.custome.spring.config.file.controller"* />  <context:component-scan base-package = *"com.custome.spring.config.file.service"* />    </beans> |

**What happens when hit this URL:** [**http://localhost:8080/SpringCustomeConfigName/helloAgain**](http://localhost:8080/SpringCustomeConfigName/helloAgain)

1. First http request goes to web.xml who send the request to Dispatcher servlet (front controller) and load the spring configuration file declared inside <init-param> tag into spring web container.
2. Dispatcher servlet consult with handler mapping, the handler mapping will scan the package and suggest the particular controller class for which http request has come.
3. The controller will send this request to the particular method based on URL pattern defined in the controller class (@RequestMapping (value =”/”).
4. Finally, method will process the request by calling the appropriate service and returns ModelAndView object to the controller class which in-turn returns the same object to dispatcher servlet.
5. The dispatcher servlet sends the view name from ModelAndView object to the View Resolver to find the actual view. Like in the previous example in spring configuration file we have set prefix and suffix. Prefix will take view name[welcomejsp] from the mentioned location ["/WEB-INF/jsp/"] and then add suffix [.jsp] with prefix. Finally, we will get final view name [welcomejsp.jsp]
6. Once view gets resolved then view will be rendered to the user with the help of model object.

**Multi Controller class in Spring MVC application:**

**Web.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>MultiControllerSpringMVC</display-name>    <servlet>  <servlet-name>MultiController</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/multcontroller.xml, /WEB-INF/beans.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>MultiController</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

**beans.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>  <context: component-scan base-package=*"com.multi.controller"* />  <context: component-scan base-package=*"com.multi.service"* />    </beans> |

**multcontroller.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>    <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> |

**HomePageController.java**

|  |
| --- |
| **package** com.multi.controller;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  @Controller  @RequestMapping(value="/")  **public** **class** HomePageController {  @RequestMapping(value="/", method = RequestMethod.***GET***)  **public** String welcome() {  **return** "homePage";  }  } |

**UserController.java:**

|  |
| --- |
| **package** com.multi.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.servlet.ModelAndView;  **import** com.multi.model.Users;  **import** com.multi.service.HomePageService;  @Controller  **public** **class** UserController {  @Autowired  **private** HomePageService homePageService;  @RequestMapping(value="/listUsers")  **public** ModelAndView listUser() {  **return** **new** ModelAndView("listusers","listUsersMessage",homePageService.listUsersMessage());  }  @GetMapping(value="/saveUsers")  **public** ModelAndView saveUsers(Users users) {  **return** **new** ModelAndView("saveUsers","saveUsersMessage", homePageService.saveUsers());  }  @GetMapping(value="/deleteUsers")  **public** ModelAndView deleteUsers(Users user) {  **return** **new** ModelAndView("deleteUsers","deleteUserMessage",homePageService.deleteUsers());  }  } |
| **HomePageService.java**  **package** com.multi.service;  **import** org.springframework.stereotype.Service;  @Service  **public** **class** HomePageService {  **public** String listUsersMessage() {  **return** "List of users returned";  }  **public** String saveUsers() {  **return** "List of users is saved";  }  **public** String deleteUsers() {  // **TODO** Auto-generated method stub  **return** "List of users is deleted";  }  } |
| **Users.java**  **package** com.multi.model;  **public** **class** Users {  } |

|  |  |
| --- | --- |
| **homepage.jsp**  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h1>Welcome to home page</h1>  </body>  </html> | **listusers.jsp**  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h1>${listUsersMessage}</h1>  </body>  </html> |
| **saveUsers.jsp**  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h1>${saveUsersMessage}</h1>  </body>  </html> | **deleteUsers.jsp**  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h2>${deleteUserMessage}</h2>  </body>  </html> |

1. **Web.xml**
2. **beans.xml**
3. **multcontroller.xml**
4. **HomePageController.java**
5. **HomePageService.java**
6. **Users.java**
7. **homepage.jsp**
8. **listusers.jsp**
9. **saveUsers.jsp**
10. **deleteUsers.jsp**

[**http://localhost:8080/MultiControllerSpringMVC/**](http://localhost:8080/MultiControllerSpringMVC/)

# **output: Welcome to home page**

[**http://localhost:8080/MultiControllerSpringMVC/saveUsers**](http://localhost:8080/MultiControllerSpringMVC/saveUsers)

# **output: List of users is saved**

[**http://localhost:8080/MultiControllerSpringMVC/deleteUsers**](http://localhost:8080/MultiControllerSpringMVC/deleteUsers)

## output: List of users is deleted

**Spring MVC application by implementing Controller interface instead of @Controller annotation:**

**Example:**

**create-spring-mvc-controller-by-extending-abstractcontroller-class**

Example

**@Pathvariable-annotation-in-spring-mvc:**

1. @PathVaraible annotation is used to extract any value which is embedded in the URL itself.
2. Spring call it a URI template, where @PathVaraible is used to obtain some placeholders from the URI itself.
3. In RESTful Web services, we know that the [REST URIs](http://www.java67.com/2016/09/when-to-use-put-or-post-in-restful-web-services.html) contains values e.g. a REST API to retrieve a book using ISBN number looks like following:
4. **URL:** [**http://localhost:8080/book/9783827319333**](http://localhost:8080/book/9783827319333)

Now, to extract the value of ISBN number from the URI in your Spring MVC Controller’s handler method, you can use @PathVaraible annotation as shown in following code:

|  |
| --- |
| @RequestMapping(value="/book/{ISBN}", method= RequestMethod.GET)    public String showBookDetails(@Pathvariable("ISBN") String id, Model model) {    model. addAttribute ("ISBN", id);  return "bookDetails";  } |

**public @interface PathVariable**

**Hence @PathVariable** **Annotation indicates that a method parameter should be bound to a URI template variable. Supported for** [**RequestMapping**](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html) **annotated handler methods**.

## Difference between @PathVariable and @RequestParam in Spring

Now let’s understand both the difference as well as how to use both  
@RequestParam and @PathVariable in Spring MVC application, let’s revise the key difference between them from an interview perspective.

1) The @RequestParam is used to extract query parameters while @PathVariable is used to extract data right from the URI.

2) @RequestParam is more useful on a traditional web application where data is mostly passed in the query abatements while @PathVariable is more suitable for RESTful web services where URL contains values e.g. http://localhost:8080/book/9783827319333, here data, which is ISBN number is part of URI.

3) @RequestParam annotation can specify default values if query parameter is not present or empty by using a defaultValue attribute, provided required attribute is false.

4) Spring MVC allows you to use multiple @PathVariable annotations in the same method, provided, no more than one argument has the same pattern.

That’s all about the **difference between @PathVariable and @RequestParam in Spring MVC**. Even though both are used to extract data from URL, @RequestParam is used to retrieve query parameters, anything after ? in the URL, while @PathVariable is used to retrieve values from URI itself.  This concept is very important for both traditional web application development as well as developing RESTful Web Services using Spring, so you must spend some time to understand it better.

**Note** that the template variable name (between the curly braces) and the parameter name should match.

|  |
| --- |
| **@RequestMapping (value = "/orders/{id}", method = RequestMethod.GET)**  **@ResponseBody**  **public String getOrder (@PathVaraible final String id) {**  **return "Order ID: " + id;**  } |

**If we wanted** to use a different parameter name, we could specify the template variable name in the annotation:

|  |
| --- |
| **@RequestMapping (value = "/orders/{id}", method = RequestMethod.GET)**  **@ResponseBody**  **public String getOrder(@PathVaraible("id") final String orderId) {**  **return "Order ID: " + orderId;**  }  Here (“id”) is declared as orderId |

**Example**

|  |
| --- |
| package com.codebyamir.demo.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.ResponseBody;  @Controller  @RequestMapping (value = "/orders", method=RequestMethod.GET)  public class DemoController {  **@RequestMapping (method = RequestMethod.GET)**  **@ResponseBody**  **public String getAllOrders () {**  **return "All orders";**  **}** |
| **@RequestMapping (value = "/{orderId}", method = RequestMethod.GET)**  **@ResponseBody**  **public String getOrder (@PathVaraible final String orderId) {**  **return "Order ID: " + orderId;**  **}** |
| **@RequestMapping (value = "/{orderId}/items", method=RequestMethod.GET)**  **@ResponseBody**  **public String getItemsByOrder (@PathVaraible final String orderId) {**  **return "Items for Order ID " + orderId;**  **}** |
| **@RequestMapping (value = "/{orderId}/items/{itemId}", method=RequestMethod.GET)**  **@ResponseBody**  **public String getItem (@PathVaraible final String orderId, @PathVaraible final**  **String itemId) {**  **return "Order ID: " + orderId + ", Item ID: " + itemId;**  **}**  **}** |

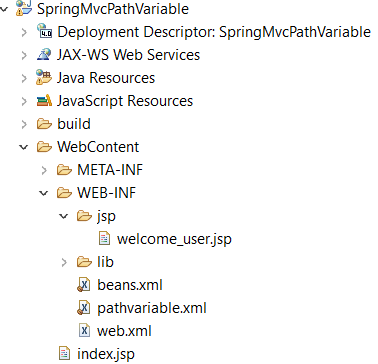
This controller supports the following request URI's:

GET /orders  
GET /orders/100  
GET /orders/100/items  
GET /orders/100/items/1

**Example:**

|  |
| --- |
| **Web.xml:**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>SpringMvcPathVariable</display-name>  <servlet>  <servlet-name>SpringMvcPathVariabl</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/pathvariable.xml,/WEB-INF/beans.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>SpringMvcPathVariabl</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |
| **pathvariable.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>    <context:component-scan base-package=*"com.pathvariable.controller"* />  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> |

|  |
| --- |
| **package** com.pathvariable.controller;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.bind.annotation.RequestParam;  **import** org.springframework.web.servlet.ModelAndView;  **import** com.pathvariable.service.UserService;  @Controller  @RequestMapping(value="/users")  **public** **class** UserController {  @RequestMapping(value ="/user/{userName}/{age}", method=RequestMethod.***GET***)  **public** ModelAndView helloUser(@PathVariable("userName") String userName,  @PathVariable("age") **int** age,  @RequestParam("country") String country) {    **return** **new** ModelAndView("welcome\_user","welcomeUserMessage","Hello,"+userName+"you  are"+age+"years old"+"belongs to "+country);  }  } |
| <title>Insert title here</title>  </head>  <body>  ${welcomeUserMessage}  </body>  </html> |
| **Index.jsp**  <title>Insert title here</title>  </head>  <body>  <a href=*"/SpringMvcPathVariable/users/user/Arun/32?country=India"*>Click Here</a>  </body>  </html> |



**Explanation:** When we run above application.

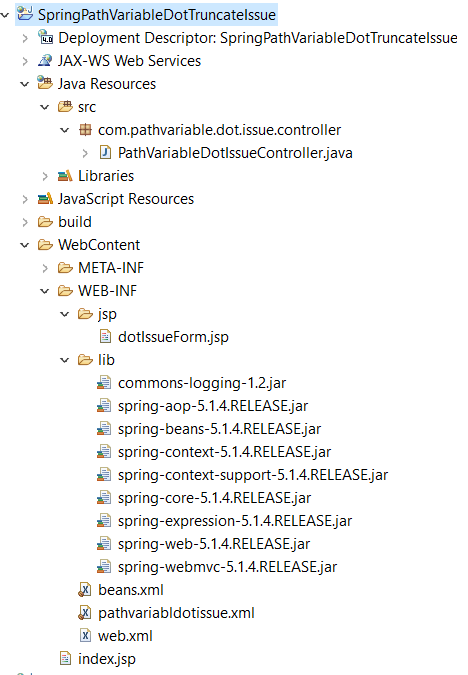
1. First index.jsp will populate with link **Click Here.** When we click this link, it will form complete URI as in the index.jsp we have already mentioned absolute path inside Anker tag.
2. <http://localhost:8080/SpringMvcPathVariable/users/user/Arun/32?country=India>
3. Here /users/user indicates RequestMapping URL. /Arun/32 indicates **@PathVariable** parameter to extracted from this URI
4. Finally ?country=India indicates **@RequestParam** to be extracted from URI known as Query parameter.

**Output**:

[Click Here](http://localhost:8080/SpringMvcPathVariable/users/user/Arun/32?country=India)

**Hello, Arun you are 32 years old belongs to India**

**624-8-Pathvariable-spring-mvc-dot-get-truncated-issue-path-variable-annotation**



|  |
| --- |
| **Web.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>SpringPathVariableDotTruncateIssue</display-name>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <servlet>  <servlet-name>SpringPathVariableDotTruncate</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/pathvariabldotissue.xml, /WEB-INF/beans.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>SpringPathVariableDotTruncate</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

|  |
| --- |
| **pathvariabldotissue.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  [*http://www.springframework.org/schema/context/spring-context-3.0.xsd*](http://www.springframework.org/schema/context/spring-context-3.0.xsd)*"*>  <context: component-scan base-package=*"com.pathvariable.dot.issue.controller"* />  <bean class=*"org.springframework.web. servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> |
| *Controller Class*  package com.pathvariable.dot.issue.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.servlet.ModelAndView;  @Controller  @RequestMapping(value="/domains")  public class PathVariableDotIssueController {  @RequestMapping(value="/domain**/{domainName:.+}**", method=RequestMethod.GET)  public ModelAndView dotIssueMethod(@PathVariable("domainName") String domainNames) {    return new ModelAndView("dotIssueForm","dotIssueMessage","This is my DomainName "+domainNames);  }  } |
| **Index.jsp**  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  **<a href=*"/SpringPathVariableDotTruncateIssue/domains/domain/www.yahoo.com"*>Test Domain</a>**  </body>  </html> |
| **dotIssueForm.jsp**  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  ${dotIssueMessage}  </body>  </html> |

**When we run this program then first of all index.jsp will be executed where we will get one Link “Test Domain”.**

[Test Domain](http://localhost:8080/SpringPathVariableDotTruncateIssue/domains/domain/www.yahoo.com)

**Once we will click on this link it will form one complete URI as given below.**

[**https://localhost:8080*/SpringPathVariableDotTruncateIssue/domains/domain/www.yahoo.com***](https://localhost:8080/SpringPathVariableDotTruncateIssue/domains/domain/www.yahoo.com)

**and we will get fallowing output:**

This is my DomainNamewww.yahoo.com

**Note: In the controller class inside @RequestMapping we have used one regular expression (.+) as given below.**

@RequestMapping(value="/domain**/{domainName:.+}**", method=RequestMethod.GET)

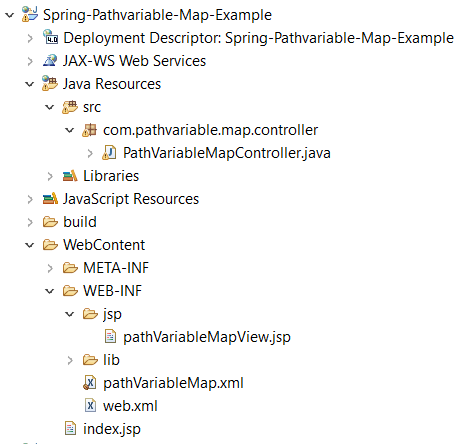
**If we don’t use this regular expression then in the output (.com) will be truncated and output will be [This is my DomainNamewww.yahoo**]. **To avoid this or to resolve this problem we have used (:.+) regular expression.**

**Note: Another approach is that we need to declare [DefaultAnnotationHandlerMapping or *RequestMappingHandlerMapping*] in the spring configuration file as given below. And we need to make *useSuffixPatternMatch false.***

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>    **<bean class=*"org.springframework.web.servlet.mvc.annotation.RequestMappingHandlerMapping"*>**  **<property name=*"useSuffixPatternMatch"* value=*"false"*/>**  **</bean>**  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> |

**spring-@Pathvariable-map-example**

|  |
| --- |
| **Web.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>Spring-Pathvariable-Map-Example</display-name>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <servlet>  <servlet-name>Spring-Pathvariable-Map-Example</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/pathVariableMap.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>Spring-Pathvariable-Map-Example</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |
| **pathVariableMap.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>    <context:component-scan base-package=*"com.pathvariable.map.controller"*></context:component-scan>  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> |
| **PathVariableMapController.java**  package com.pathvariable.map.controller;  import java.util.Map;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.servlet.ModelAndView;  @Controller  @RequestMapping(value="/users")  public class PathVariableMapController {    /\*@GetMapping(value="/user/{userName}/{age}")  public ModelAndView pathVaribaleMapMethod(@PathVariable("userName") String usrName,  @PathVariable("age") int age,  @RequestParam("country") String country) {\*/  @GetMapping(value="/user/{userName}/{age}")  public ModelAndView pathVaribaleMapMethod (@PathVariable Map<String, String> pathName,  @RequestParam("country") String country) {    String usrName= pathName.get("userName");  int age = Integer.parseInt(pathName.get("age"));    return new ModelAndView ("pathVariableMapView", "pathVariableMapMessage", "Your Name is "+usrName+" "+"and age is "+age+” from "+ country);  }  }  **In the commented code we have written @PathVariable to extract each query string from the mentioned in URL**  **<h1><a href=*"/Spring-Pathvariable-Map-Example/users/user/Bunty/30?country=US"*>Test PathVaraible Map</a></h1>**  **Note:** Same path variable we have extracted from URL using @PathVariable Map. In the above example we have used Map get() method to get its value by passing **@**GetMapping variable as key. Based on this key it will get value form the Query String mentioned in the URL. |
| **index.jsp**  <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h1><a href=*"/Spring-Pathvariable-Map-Example/users/user/Bunty/30?country=US"*>Test PathVarable Map</a></h1>  </body>  </html>  **Explanation**:  In the href we have mentioned (Application Name as context root) then class level @RequestMapping(value="/users") and then method level @GetMapping(value="/user/{userName}/{age}") and then we have passed dynamic value username {Bunty} and age {30} to be extracted by @PathVariable and finally we have provided {*?country=US"} to extracted by @RequestParam annotation.* |
| <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>pathVariableMapView</title>  </head>  <body>  <h1>${pathVariableMapMessage}</h1>  </body>  </html> |



**Now when execute the above application:**

1. Index.jsp is loaded. And href link **Test PathVaraible Map** will be displayed. Inside this **Anker** **tag** we have defined the url =*"/Spring-Pathvariable-Map-Example/users/user/Bunty/30?country=US"*
2. When we click the above link, it calls to controller class annotated by (/users) RequestMapping and inside this class it calls method annotated by (/user) and mapped with @PathVariable and @RequestParam annotation.
3. In method pathVariableMapView is mapped in ModelAndView, so finally pathVariableMapView. Jsp will be called and inside this jsp pathVariableMapMessage is configured inside ${pathVariableMapMessage} which is nothing but the key for message written inside ModelAndView.
4. Finally, “Your Name is Bunty and age is 30 from US” is displayed on the browser.

**10-specifying-multiple url-mapping-for-string-mvc-handler-method**

Let’s say in a controller class if we have more than one request handler method then how to do url mapping for those handler methods.

As we know that Spring MVC provide @RequestMapping for specifying URL mapping. This annotation can be used at the class level as well as at method level.

**When we annotate @RequestMapping at class level then class becomes single action controller on the other hand when we annotate @RequestMapping at method level then class becomes multi action controller.**

**Case-1: When we have @RequestMapping(value=”/”) at class level only**

|  |  |
| --- | --- |
| **package com.multi.controller;**  **import org.springframework.stereotype.Controller;**  **import org.springframework.web.bind.annotation.RequestMapping;**  **import org.springframework.web.bind.annotation.RequestMethod;**  **@Controller**  **@RequestMapping(value="/")**  **public class HomePageController {**  **@RequestMapping (method = RequestMethod.GET)**  **public String welcome() {**  **return "homePage";**  **}**  **}** | <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Insert title here</title>  </head>  <body>  <h1>Welcome to home page</h1>  </body>  </html> |

**Here at the class level we have annotated @RequestMapping(value=”/”) and at the method level have mentioned GET method. When we execute this class then it will call welcome method by default and will return homepage jsp.** [**http://localhost:8080/MultiControllerSpringMVC/**](http://localhost:8080/MultiControllerSpringMVC/)

**Case-2: When we have @RequestMapping(value=”/”) at class level and method level.**

|  |  |
| --- | --- |
| **package com.multi.controller;**  **import org.springframework.stereotype.Controller;**  **import org.springframework.web.bind.annotation.RequestMapping;**  **import org.springframework.web.bind.annotation.RequestMethod;**  **@Controller**  **@RequestMapping(value="/")**  **public class HomePageController {**  **@RequestMapping(value="/homeVisit", method = RequestMethod.GET)**  **public String welcome() {**  **return "homePage";**  **}**  **}** | **In this case the URL will be :**  [**http://localhost:8080/MultiControllerSpringMVC/homeVisit**](http://localhost:8080/MultiControllerSpringMVC/homeVisit)  **We have to mention method level URI /homeVisit**  [**/MultiControllerSpringMVC/homeVisit**](http://localhost:8080/MultiControllerSpringMVC/homeVisit) |

**Case-3:** When **we have annotated @RequestMapping(value=”/”) only at method level as shown in the below code.**

|  |
| --- |
| package com.multi.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.servlet.ModelAndView;  import com.multi.model.Users;  import com.multi.service.HomePageService;  @Controller  public class UserController {  @Autowired  private HomePageService homePageService;  @RequestMapping(value="/listUsers")  public ModelAndView listUser () {  return new ModelAndView ("listusers", "listUsersMessage", homePageService.listUsersMessage());  }  @GetMapping(value="/saveUsers")  public ModelAndView saveUsers (Users users) {  return new ModelAndView ("saveUsers","saveUsersMessage", homePageService.saveUsers());  }  @GetMapping(value="/deleteUsers")  public ModelAndView deleteUsers(Users user) {  return new ModelAndView ("deleteUsers", "deleteUserMessage",homePageService.deleteUsers());  }  } |

The above code is multi action controller. Here we have annotated @RequestMapping at methods level only. So in this case to call the particular method we have to mention Method level URI only.

<http://localhost:8080/MultiControllerSpringMVC/listUsers>

<http://localhost:8080/MultiControllerSpringMVC/saveUsers>

<http://localhost:8080/MultiControllerSpringMVC/deleteUsers>

Since we have not mentioned any annotation at class level so in this case, we will mention method level URI only.

**Case-4**: Note: the “value” attribute inside @RequestMapping or @GetMapping represents an Array. So we can mention an array of URI as given in the below code.

|  |
| --- |
| @Controller  @RequestMapping(value = "/")  public class HelloWorldController {  //@RequestMapping(value= {"/hello","/hi"},method = RequestMethod.GET)  @GetMapping(value= {"/hello","/hi"})  public ModelAndView welcome() {  return new ModelAndView("welcomejsp", "welcomeMessages", "Welcome to Spring HelloWorld!!!tttt");  }  } |

The above method welcome will be executed on both either on /hello or /hi.

**Specifying-http-request-methods-for-handler-method-in-spring-mvc**

Note that, by default, [DispatcherServlet](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html) supports GET, HEAD, POST, PUT, PATCH and DELETE only. DispatcherServlet will process TRACE and OPTIONS with the default HttpServlet behavior unless explicitly told to dispatch those request types as well: Check out the "dispatchOptionsRequest" and "dispatchTraceRequest" properties, switching them to "true" if necessary.

public enum RequestMethod

{

GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE;

}

All Http methods are public static and final:

**public static final**[**RequestMethod**](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMethod.html) **GET** **OR** HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE

|  |  |
| --- | --- |
| GET |  |
| HEAD |  |
| POST |  |
| PUT |  |
| PATCH |  |
| DELETE |  |
| OPTIONS |  |
| TRACE |  |

**624-12-Using-@RequestParam -to-handler-methodpart**

**@RequestParam uses in the spring:**

1. Web.xml
2. mappingReqeustParam.xml
3. UserController.java
4. UserModel.java
5. Register\_user.jsp
6. Success\_registered.jsp

|  |
| --- |
|  |
| **Web.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>MappingRequestParameters-to-HandlerMethod</display-name>  <servlet>  <servlet-name>MappingRequestParameters-to-HandlerMethod</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/mappingReqeustParam.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>MappingRequestParameters-to-HandlerMethod</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |
| Explanation: **Configuring dispatcher** servlet and mentioning mappingReqeustParam.xml inside init-param. Configuring url pattern. |
| **mappingReqeustParam.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>    <context: component-scan base-package=*"com.mapping.req. param. Controller" /*>  <bean class=*"org.springframework.web. servlet. view. InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> | |
| **UserModel.java**  **package com.mapping.req.param.model;**  **public class UserModel {**  **private String name;**  **private String email;**  **private int age;**  **private String country;**    **public UserModel(String name, String email, int age, String country) {**  **super();**  **this.name = name;**  **this.email = email;**  **this.age = age;**  **this.country = country;**  **}**  **public String getName() {**  **return name;**  **}**  **public String getEmail() {**  **return email;**  **}**  **public int getAge() {**  **return age;**  **}**  **public String getCountry() {**  **return country;**  **}**  **}**  **In the Model class we have created four variable and one parametrized constructor and its getter methods.** | |
| **The variable name, jsp input text name**  **And in controller class @RequestParam name and jsp input text name should be same.**  In the below controller class same functionality has been written in three ways.   1. In the first way we have written @RequestParam variable equal to jsp text variable which is getting value from field jsp input text box value. And same @RequestParam value is being used inside constructor to get UserModel class object. 2. In the second way we have used @RequestParam Map and then using Map object we are getting jsp input values. 3. In the third way instead of creating ModelAndView object using new keyword we have just created ModelAndView object reference inside handler method and then using ref we are setting view.   **UserController.java**  **package** com.mapping.req.param.controller;  **import** java.util.Map;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.bind.annotation.RequestParam;  **import** org.springframework.web.servlet.ModelAndView;  **import** com.mapping.req.param.model.UserModel;  @Controller  **public** **class** UserController {    @RequestMapping(value="/", method=RequestMethod.***GET***)  **public** String userRegisterHome() {  **return**"Register\_user";  }    @RequestMapping(value = "/registerUser", method = RequestMethod.***POST***)  **public** ModelAndView registerSuccess(@RequestParam("name") String userName,  @RequestParam String email,  @RequestParam **int** age,  @RequestParam String country) {  UserModel user = **new** UserModel (userName, email, age, country);  ModelAndView modelAndView = **new** ModelAndView("Success\_registered");  modelAndView.addObject("userModelKey", user);  **return** modelAndView;  }  In the above code we are getting jsp input text value by using RequestParam. Here the variable in the RequestParam , Model class variable and inside jsp page variable should be same.  **OR**  @RequestMapping (value = "/registerUser", method = RequestMethod.***POST***)  **public** ModelAndView registerSuccess (@RequestParam Map<String, String> reqParam) {  UserModel user = **new** UserModel(reqParam.get("name"),  reqParam.get("email"),  Integer.*parseInt*(reqParam.get("age")),  reqParam.get("country"));  ModelAndView modelAndView = **new** ModelAndView("Success\_registered");  modelAndView.addObject("userModelKey", user);  **return** modelAndView;  }  Here we create Map of RequestParam and inside UserModel we are getting value by using map object reqParam.  **OR**  @RequestMapping (value="/registerUser", method= RequestMethod.***POST***)  **public** ModelAndView registerSuccess (@RequestParam Map<String, String> reqParam,  ModelAndView modelAndView) {    UserModel user= **new** UserModel(reqParam.get("name"),  reqParam.get("email"),  Integer.*parseInt*(reqParam.get("age")),  reqParam.get("country"));  modelAndView.setViewName("Success\_registered");  modelAndView.addObject("userModelKey", user);    **return** modelAndView;  }  }  **Here ModelAndView is being used inside the handler method, we don’t need to create separate object as in case of above code.** | |
| **Register.jsp**  <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>UserRegisterform</title>  </head>  <body>  <form action=*"/MappingRequestParameters-to-HandlerMethod/registerUser"* method=*"post"*>  <table>  <tr>  <td>User Name</td>  <td><input type=*"text"* name=*"name"*></td>  </tr>  <tr>  <td>Email</td>  <td><input type=*"text"* name=*"email"*></td>  </tr>  <tr>  <td>Age</td>  <td><input type=*"text"* name=*"age"*></td>  </tr>  <tr>  <td>Country</td>  <td><select name=*"country"*>  <option value=*"India"*>India</option>  <option value=*"Australia"*>Australia</option>  <option value=*"US"*>US</option>  </select>  </td>  </tr>  <tr>  <td><input type=*"submit"* value=*"Register"*></td>  </tr>  </table>  </form>  </body>  </html> | |
| **Success\_Regiesterd.jsp**  <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Successfully Registered</title>  </head>  <body>  <h3>You have successfully registered with following information:</h3>  </body>  <a href=*"/MappingRequestParameters-to-HandlerMethod/"*>Back</a>  <table>  <tr>  <td>User Name</td>  <td>${userModelKey.name}</td>  </tr>  <tr>  <td>Email</td>  <td>${userModelKey.email}</td>  </tr>  <tr>  <td>Age</td>  <td>${userModelKey.age}</td>  </tr>  <tr>  <td>Country</td>  <td>${userModelKey.country}</td>  </tr>  </table>  </html> | |

**624-14@Requestparam-of-type-map-spring-mvc-example:**

**@RequestParam with Map collection:**

**624-15modelandview-in-spring-mvc**

|  |
| --- |
| @RequestMapping(value="/", method=RequestMethod.***GET***)  **public** String userRegisterHome() {  **return**"Register\_user";  } |

The above method simply returns one string, which is nothing but the logical name of view (jsp page) whose location is mentioned in spring configuration file. This view resolve is taken care by InternalViewResolver and dispatcher servlet. Here in the above method we are not return any extra information (Model) with the view. But if we want to return some extra information (Model) along with view of the spring MVC then in that case we must use ModelAndView. In this we can return model object (extra information) along with view. Like given below.

|  |
| --- |
| @RequestMapping(value="/registerUser", method= RequestMethod.***POST***)  **public** ModelAndView registerSuccess(@RequestParam Map<String,String> reqParam) {    UserModel user= **new** UserModel(reqParam.get("name"),  reqParam.get("email"),  Integer.*parseInt*(reqParam.get("age")),  reqParam.get("country"));    ModelAndView modelAndView = **new** ModelAndView("Success\_registered");  modelAndView.addObject("userModelKey", user);    **return** modelAndView;  } |

In the above example ModelAndView have one method addObject to add model object key and value.

Note: We don’t need to write a separate line of code for creating ModelAndView object. We just need to instantiate this ModelAndView object inside the handler method and the using ModelAndView object we just need to set view Name as given below.

|  |
| --- |
| @RequestMapping(value="/registerUser", method= RequestMethod.***POST***)  **public** ModelAndView registerSuccess(@RequestParam Map<String,String> reqParam, ModelAndView modelAndView) {    UserModel user= **new** UserModel(reqParam.get("name"),  reqParam.get("email"),  Integer.*parseInt*(reqParam.get("age")),  reqParam.get("country"));    modelAndView.setViewName("Success\_registered");  modelAndView.addObject("userModelKey", user);    **return** modelAndView;  } |

**624-Using Model object in Spring MVC (MultiWaysToDisplayModelDataOnJSP)**

**In how many ways we can display or map Model data on the JSP page?**

1. By using ModelAndView. (As we have done in the above example)
2. By Using Model Object.
3. By Using Map.

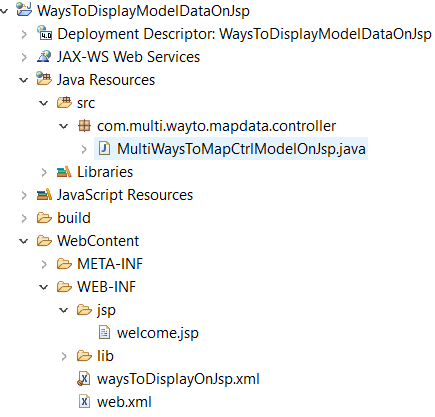
So far, we came to know that if we want to send model (object or information) along with view then we can use ModelAndView. Now let’s say if we don’t want to use ModelAndView and we want to send data from controller to view (jsp) page then in that case we can use either Model object or Map object.

**By Using Model Object**. Example:

|  |
| --- |
| @RequestMapping(value = "/")  **public** String usingModel(Model model) {  model.addAttribute("welocmeMessage", "Message by using Model...");  model.addAttribute("Header", "This is my header..");  **return** "welcome";  } |

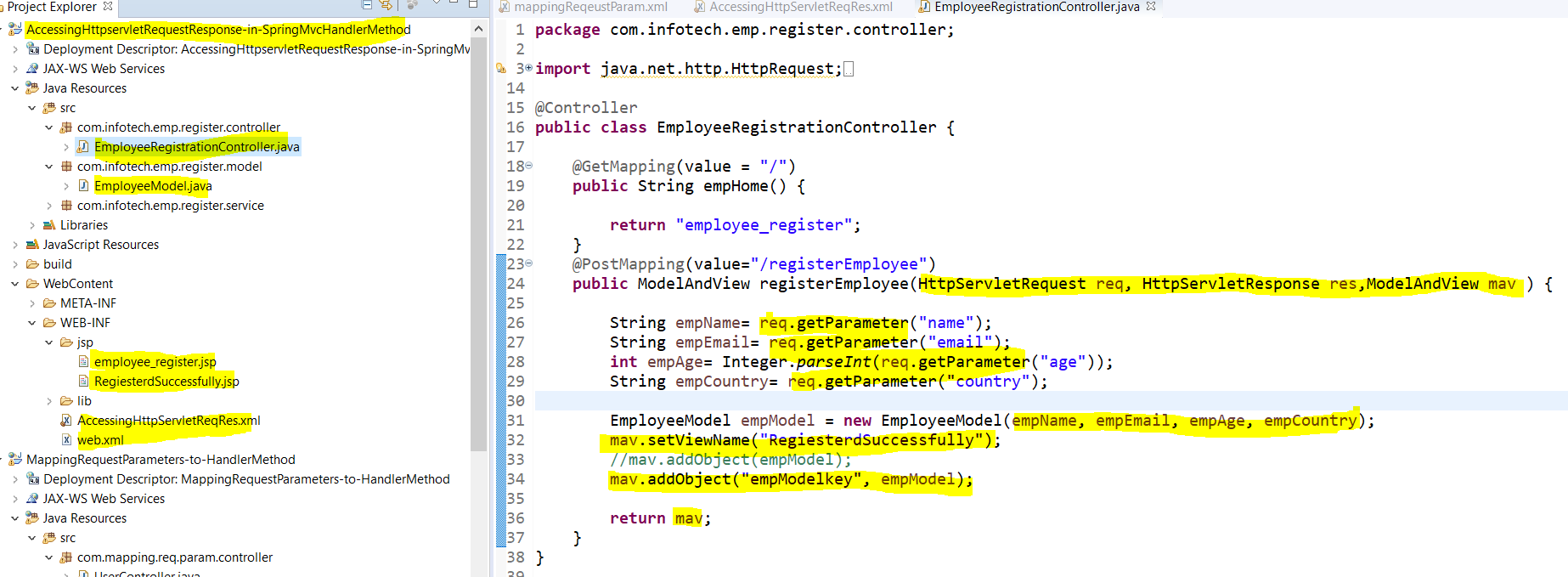
**By Using Map. Example**

|  |
| --- |
| @RequestMapping(value="/")  **public** String usingMap(Map<String,String>usingMap) {    usingMap.putIfAbsent("welocmeMessage”, Message by using Map...");  usingMap.put ("Header", "This is my header.");  **return** “welcome";  } |
| In the jsp page welocmeMessage and Header are being mapped by using jsp expression language <h1>${welocmeMessage}</h1>, <h1>${Header}</h1>  So, in this way we can map Controller model or Map object with jsp page. Now whenever we execute the any of the controller, the controller messages will be displayed in the welcome.jsp page  **welcome.jsp page**  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>WaysToDisplayModelDataOnJsp</title>  </head>  <body>  <h1>Welcome to WaysToDisplayModelDataOnJsp</h1>  <hr>  <h1>${welocmeMessage}</h1>  <hr>  <h1>${Header}</h1>  </body>  </html> |



**624-17accessing-httpservletrequest-and-httpservletresponse-in-spring-mvc-handler-method**

Here we will see that how to access HttpServlet request and HttpServlet response object in spring mvc request handler method. Some time we need to access HttpServlet request and response object for some functionality. Like set cookies or session etc.



|  |
| --- |
| Web.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>AccessingHttpservletRequestResponse-in-SpringMvcHandlerMethod</display-name>  <servlet>  <servlet-name>AccessingHttpServletReqRes</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/AccessingHttpServletReqRes.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>AccessingHttpServletReqRes</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |
| **AccessingHttpServletReqRes.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns = *"http://www.springframework.org/schema/beans"*  xmlns:context = *"http://www.springframework.org/schema/context"*  xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation = *"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>    <context:component-scan base-package=*"com.infotech.emp.register.controller"* />  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"*/>  </bean>  </beans> |
| **EmployeeModel.java**  **package** com.infotech.emp.register.model;  **public** **class** EmployeeModel {  **private** String name;  **private** String email;  **private** **int** age;  **private** String country;    **public** EmployeeModel(String name, String email, **int** age, String country) {  **super**();  **this**.name = name;  **this**.email = email;  **this**.age = age;  **this**.country = country;  }  **public** String getName() {  **return** name;  }  **public** String getEmail() {  **return** email;  }  **public** **int** getAge() {  **return** age;  }  **public** String getCountry() {  **return** country;  }  }  We have created Employee Model variable and its getter method and same variable name is getting assigned in employee\_register.jsp <input> name |
| **employee\_register. Jsp**  <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>EmployeeRegisterform</title>  </head>  <body>  **<form action=*"/AccessingHttpservletRequestResponse-in-SpringMvcHandlerMethod/registerEmployee"* method=*"post"*>**  <table>  <tr>  <td>Employee Name</td>  <td><input type=*"text"* name=*"name"*></td>  </tr>  <tr>  <td>Email</td>  <td><input type=*"text"* name=*"email"*></td>  </tr>  <tr>  <td>Age</td>  <td><input type=*"text"* name=*"age"*></td>  </tr>  <tr>  <td>Country</td>  <td><select name=*"country"*>  <option value=*"India"*>India</option>  <option value=*"Australia"*>Australia</option>  <option value=*"US"*>US</option>  </select>  </td>  </tr>  <tr>  <td><input type=*"submit"* value=*"Register"*></td>  </tr>  </table>  </form>  </body>  </html>  Note: On clicking on Register Button, it will action url mentioned in the <form> tag. |
| EmployeeRegistrationController.java  package com.infotech.emp.register.controller;  import java.net.http.HttpRequest;  import java.net.http.HttpResponse;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.servlet.ModelAndView;  import com.infotech.emp.register.model.EmployeeModel;  @Controller  public class EmployeeRegistrationController {  @GetMapping (value = "/")  public String empHome () {    return "employee\_register";  }  @PostMapping(value="/registerEmployee")  public ModelAndView registerEmployee (HttpServletRequest req, HttpServletResponse res, ModelAndView mav) {    String empName= req. getParameter("name");  String empEmail= req. getParameter("email");  int empAge= Integer.parseInt(req. getParameter("age"));  String empCountry= req. getParameter("country");    EmployeeModel empModel = new EmployeeModel (empName, empEmail, empAge, empCountry);  mav. setViewName("RegiesterdSuccessfully");  //mav. addObject(empModel);  mav. addObject ("empModelkey", empModel);  **return mav;**  }  }  Note: We are using HttpServletRequest getParameter method to get jsp input value. |
| **RegiesterdSuccessfully.jsp**  <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Successfully Registered</title>  </head>  <body>  <h3>You have successfully registered with following information:</h3>  </body>  <a href=*"/AccessingHttpservletRequestResponse-in-SpringMvcHandlerMethod/"*>Back</a>  <table>  <tr>  <td>Employee Name</td>  <td>${empModelkey.name}</td>  </tr>  <tr>  <td>Email</td>  <td>${empModelkey.email}</td>  </tr>  <tr>  <td>Age</td>  <td>${empModelkey.age}</td>  </tr>  <tr>  <td>Country</td>  <td>${empModelkey.country}</td>  </tr>  </table>  </html>  Note: using empModelkey set in the ModelAndView object inside controller class we are setting the value in the jsp using jsp expression. |

**624-18 -Page-redirection-example in-spring-web-mvc**

**624-19modelattribute-annoatation handling-form-submission-in-spring-mvcpart:**

**(Spring MVC form Submission)**

**624-21spring-mvc-form-bean-validation:**

**624-22spring-mvc-form-bean-validation-using-resource-bundle-form-validation**

**624-23more-things-on-spring-mvc-form-bean-validation-using-resource-bundle**

**624-24spring-mvc-hibernate-registration-and-login-example**