Spring MVC Basics
- Spring MVC - Home
- Spring MVC - Overview
- Spring MVC - Environment Setup
- Spring MVC - Hello World Example

Spring MVC - Form Handling
- Spring MVC - Form Handling
- Spring MVC - Page Redirection
- Spring MVC - Static Pages
- Spring MVC - Textbox
- Spring MVC - Password
- Spring MVC - Textarea
- Spring MVC - Checkbox
- Spring MVC - Checkboxes
- Spring MVC - Radiobutton
- Spring MVC - Radiobuttons
- Spring MVC - Dropdown
- Spring MVC - Listbox
- Spring MVC - Hidden
- Spring MVC - Errors
- Spring MVC - Upload

Spring MVC - Handler Mapping
- Bean Name Url Handler Mapping

Spring MVC - Parameterizable View Controller Example

Advertisements

Previous Page Next Page

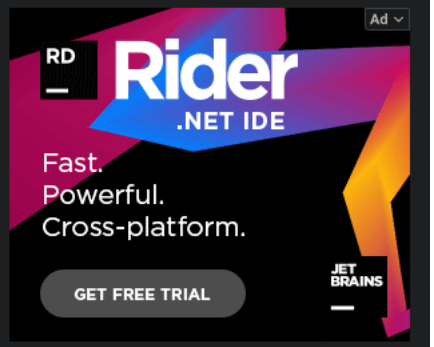
The following example shows how to use the Parameterizable View Controller method of a Multi Action Controller using the Spring Web MVC framework. The Parameterizable View allows mapping a webpage with a request.

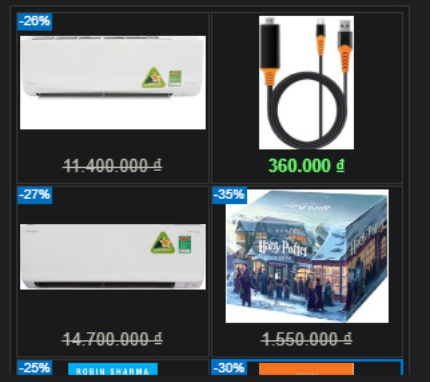
```
package com.tutorialspoint;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
public class UserController extends MultiActionController{

   public ModelAndView home(HttpServletRequest request,
      HttpServletResponse response) throws Exception {
      ModelAndView model = new ModelAndView("user");
      model.addObject("message", "Home");
      return model;
   }
}
```

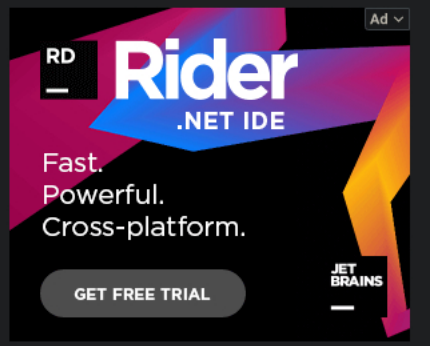
```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
   <property name="mappings">
      <value>
         index.htm=userController
      </value>
   </property>
</bean>
<bean id="userController" class="org.springframework.web.servlet.mvc.ParameterizableVi
   <property name="viewName" value="user"/>
</bean>
```

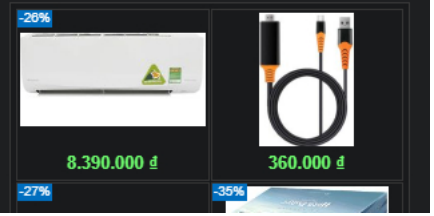
For example, using the above configuration, if URI.

- /index.htm is requested, DispatcherServlet will forward the request to the









- Simple Url Handler Mapping

Spring MVC - Controller

- Spring MVC - Multi Action Controller
- Properties Method Name Resolver
- Parameter Method Name Resolver
- Parameterizable View Controller

Spring MVC - View Resolver

- Internal Resource View Resolver
- Spring MVC - Xml View Resolver
- Resource Bundle View Resolver
- Multiple Resolver Mapping

Spring MVC - Integration

- Spring MVC - Hibernate Validator
- Spring MVC - Generate RSS Feed
- Spring MVC - Generate XML
- Spring MVC - Generate JSON
- Spring MVC - Generate Excel
- Spring MVC - Generate PDF
- Spring MVC - Using log4j

Spring Questions and Answers

- Spring - Questions and Answers

Spring Useful Resources

- Spring MVC - Quick Guide
- Spring MVC - Useful Resources
- Spring MVC - Discussion

Selected Reading

- UPSC IAS Exams Notes
- Developer's Best Practices
- Questions and Answers
- Effective Resume Writing
- HR Interview Questions
- Computer Glossary

UserController controller with viewName set as user.jsp.

To start with it, let us have a working Eclipse IDE in place and stick to the following steps to develop a Dynamic Form based Web Application using Spring Web Framework.

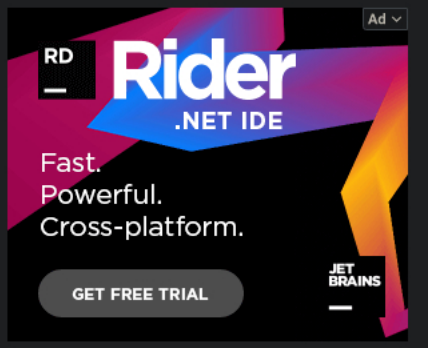
Step	Description
1	Create a project with a name TestWeb under a package com.tutorialspoint as explained in the Spring MVC - Hello World chapter.
2	Create a Java class UserController under the com.tutorialspoint package.
3	Create a view file user.jsp under the jsp sub-folder.
4	The final step is to create the content of the source and configuration files and export the application as explained below.

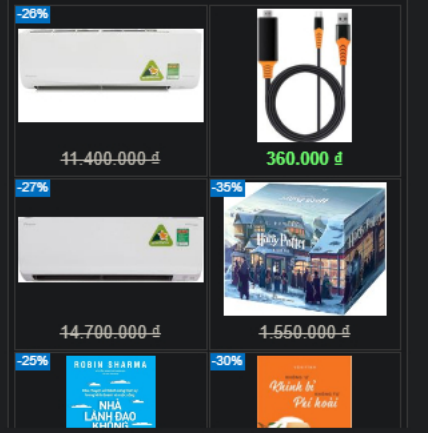
UserController.java

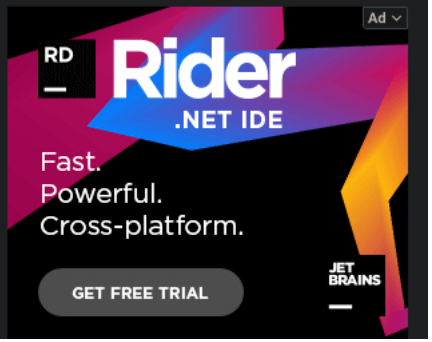
```
package com.tutorialspoint;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;

public class UserController extends MultiActionController{

   public ModelAndView home(HttpServletRequest request,
      HttpServletResponse response) throws Exception {
      ModelAndView model = new ModelAndView("user");
      model.addObject("message", "Home");
      return model;
   }
}
```

TestWeb-servlet.xml

```
<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>

   <bean class = "org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
      <property name = "mappings">
         <value>
            index.htm = userController
         </value>
      </property>
```







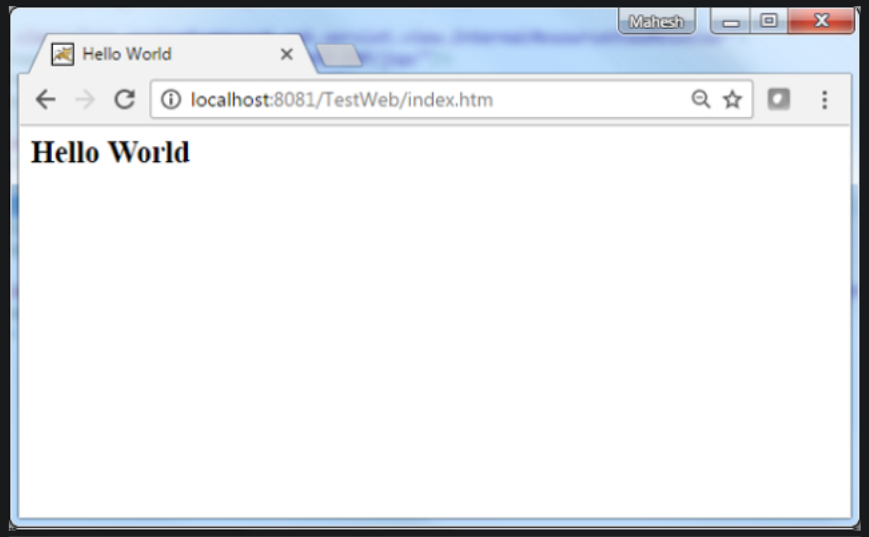
```
    </bean>
    <bean id = "userController" class = "org.springframework.web.servlet.mvc.Parameteri
       <property name = "viewName" value="user"/>
    </bean>
</beans>
```

user.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      <h2>Hello World</h2>
   </body>
</html>
```

Once you are done with creating source and configuration files, export your application. Right click on your application, use **Export → WAR File** option and save the **TestWeb.war** file in Tomcat's webapps folder.

Now, start your Tomcat server and make sure you are able to access other webpages from webapps folder using a standard browser. Now, try a URL – **http://localhost:8080/TestWeb/index.htm** and you will see the following screen, if everything is fine with the Spring Web Application.



Previous Page Next Page

Advertisements

-26% 11.400.000 ₫ 360.000 ₫
-27% 14.700.000 ₫ -35% 1.550.000 ₫
-25% 8.390.000 ₫ -30% 360.000 ₫
-27% 10.700.000 ₫ -35% 1.007.345 ₫
-25% 60.000 ₫ -30% 69.300 ₫
-31% 64.000 ₫ -35% 98.000 ₫
TIKI

We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy. Accept Learn mo