



LEARN SPRING MVC

absolute beginners

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

Controller Class Name Handler Mapping

## Spring MVC - Simple Url Handler Mapping Example

Advertisements

**YouTrack**

Try now

FREE FOR SMALL  
TEAMS FOREVER

JET  
BRAINS

The issue tracker for every team in your company

Previous Page

Next Page

The following example shows how to use Simple URL Handler Mapping using the Spring Web MVC framework. The `SimpleUrlHandlerMapping` class helps to explicitly-map URLs with their controllers respectively.

```
<beans>
    <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name = "prefix" value = "/WEB-INF/jsp/" />
        <property name = "suffix" value = ".jsp" />
    </bean>

    <property name = "urlMappings">
        <props>
            <prop key = "/welcome.htm">welcomeController</prop>
            <prop key = "/helloWorld.htm">helloController</prop>
        </props>
    </property>
</bean>

<bean id = "helloController" class = "com.tutorialspoint.HelloController" />

<bean id = "welcomeController" class = "com.tutorialspoint.WelcomeController" />
</beans>
```

For example, using above configuration, if URI

- /helloWorld.htm is requested, `DispatcherServlet` will forward the request to the **HelloController**.
- /welcome.htm is requested, `DispatcherServlet` will forward the request to the **WelcomeController**.

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

Step	Description
1	Controller Class Name Handler Mapping

TIKI

-33%



TIKI

-33%

» [Controller Class Name Handler Mapping](#)

» [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](#)

- 1 Create a project with a name `TestWeb` under a package `com.tutorialspoint` as explained in the [Spring MVC - Hello World](#) chapter.
- 2 Create Java classes `HelloController` and `WelcomeController` under the `com.tutorialspoint` package.
- 3 Create view files `hello.jsp` and `welcome.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.

## HelloController.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.AbstractController;

public class HelloController extends AbstractController{

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        ModelAndView model = new ModelAndView("hello");
        model.addObject("message", "Hello World!");
        return model;
    }
}
```

## WelcomeController.java

```
package com.tutorialspoint;

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

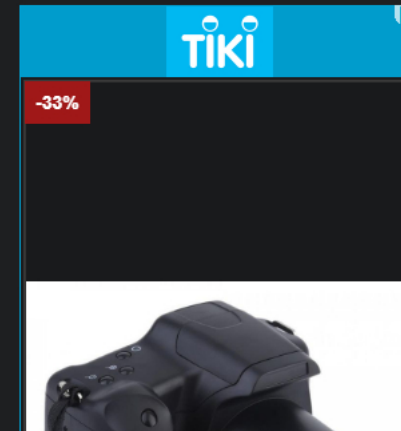
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;

public class WelcomeController extends AbstractController{

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        ModelAndView model = new ModelAndView("welcome");
        model.addObject("message", "Welcome!");
        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.xsd"
    >
```



```
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">
    <props>
      <prop key = "/welcome.htm">welcomeController</prop>
      <prop key = "/helloWorld.htm">helloController</prop>
    </props>
  </property>
</bean>

<bean id = "helloController" class = "com.tutorialspoint.HelloController" />

<bean id = "welcomeController" class = "com.tutorialspoint.WelcomeController" />
</beans>
```

## hello.jsp

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

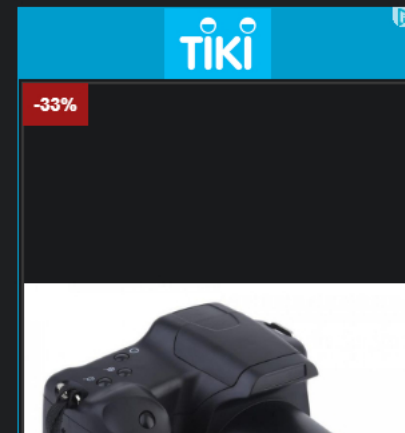
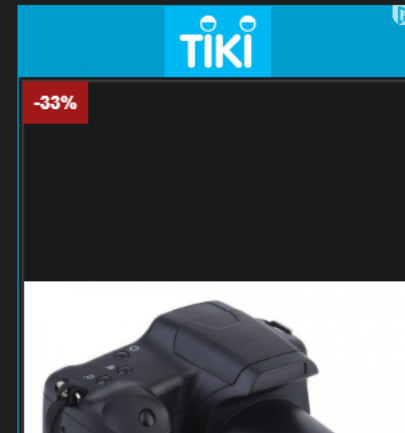
## welcome.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<html>
<head>
  <title>Welcome</title>
</head>
<body>
  <h2>${message}</h2>
</body>
</html>
```

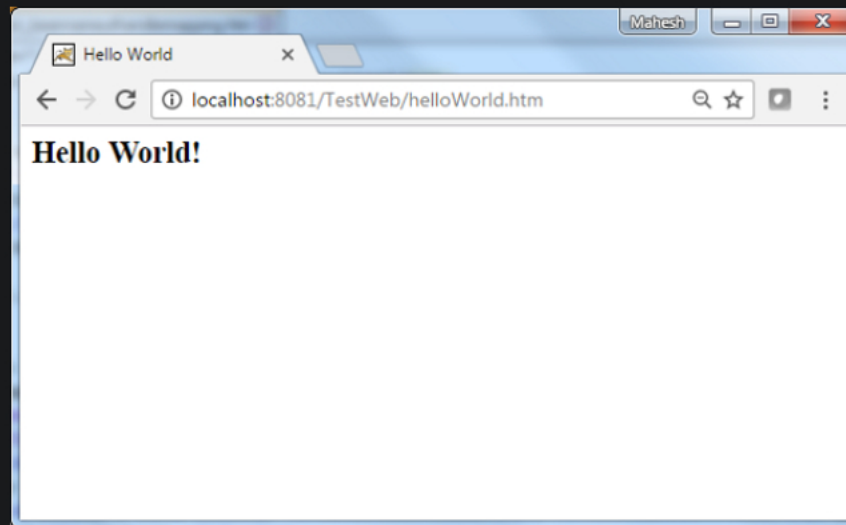
Once you are done with creating source and configuration files, export your application. Right click on your application, use the **Export** → **WAR File** option and save your **TestWeb.war** file in Tomcat's webapps folder.

Now, start your Tomcat server and make sure you are able to access other webpages from the webapps folder by using a standard browser. Try a URL – **http://localhost:8080/TestWeb/helloWorld.htm** and we will see the following screen, if

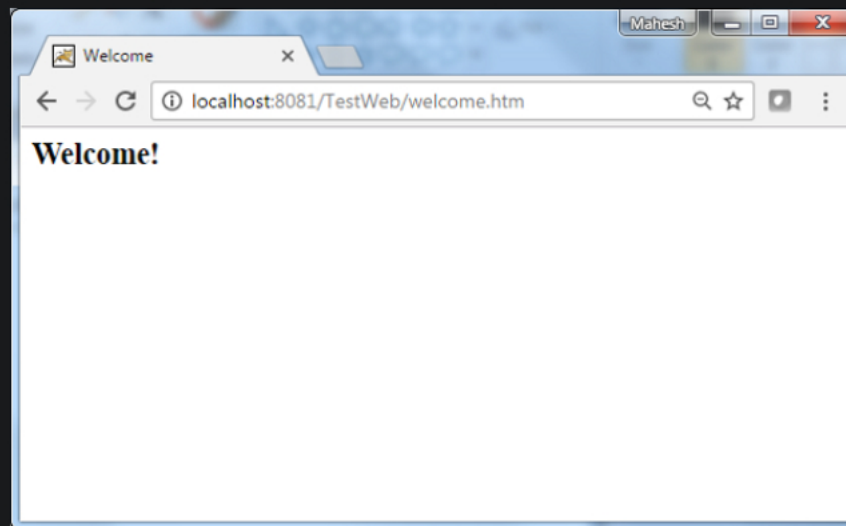
everything is set up correctly.



everything is fine with the Spring Web Application.



Try a URL <http://localhost:8080/TestWeb/welcome.htm> and you should see the following result if everything is fine with your Spring Web Application.



[Previous Page](#)

[Next Page](#)

Advertisements



[About us](#)

[Terms of use](#)

[Cookies Policy](#)

[FAQ's](#)

[Helping](#)

[Contact](#)

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

Accept

Learn more 