مثالی برای Controller Class Name Handler Mapping برای Spring MVC

در مثال زیر چگونگی استفاده از Controller Class Name Handler Mapping با استفاده از Spring Web MVC Framework نشان داده شده است. کلاس ControllerClassNameHandlerMapping، کلاسی قراردادی برای نگاشت هندلر است که درخواست های URL را در اسم controller های بیان شده در پیکربندی نگاشت می کند. این کلاس اسم کنترلرها را می گیرد و آن ها را به همراه "/" در ابتدای آن ها به حرف کوچک تبدیل می کند.

برای مثال، HelloController به URL "/hello*" نگاشت می شود.

```
<beans>
  <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.mvc.support.ControllerClassNameHandlerMapping"/>

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

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

برای مثال، با استفاده از پیکربندی بالا، اگر URI

- /helloWorld.htm یا /hello{any letter}.htm درخواست شوند، DispatcherServlet این درخواست را به HelloController ارسال می کند.

- /welcome.htm در خواست شود، DispatcherServlet این درخواست را به WelcomeController ارسال می کند.
- /Welcome.htm در خواست شود، به گونه ای که W حرف بزرگ باشد، DispatcherServlet هیچ کنترلری پیدا نمی کند و سرور خطای 404 می دهد.

برای شروع Eclipse IDE را آماده کنید و جهت توسعه ی یک برنامه ی وب پویا با استفاده از Spring Web Framework مراحل زیر را دنبال کنید.

مرحله	توضیحات
1	پروژه ای با نام TestWeb در بسته ی com.tutorialspoint همان طور که در بخش Spring MVC - Hello World توضیح داده شده است، ایجاد کنید.
2	در بسته ی com.tutorialspoint کلاس های جاوای HelloController و WelcomeController را ایجاد کنید.
3	در زیر پوشه ی jsp فایل های ویوی hello.jsp و welcome.jsp را ایجاد کنید.
4	مرحله ی نهایی ایجاد محتوای فایل های منبع و پیکربندی و اکسپورت کردن برنامه به صورت زیر است.

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/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.mvc.support.ControllerClassNameHandlerMapping"/>


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


  <bean 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>
```

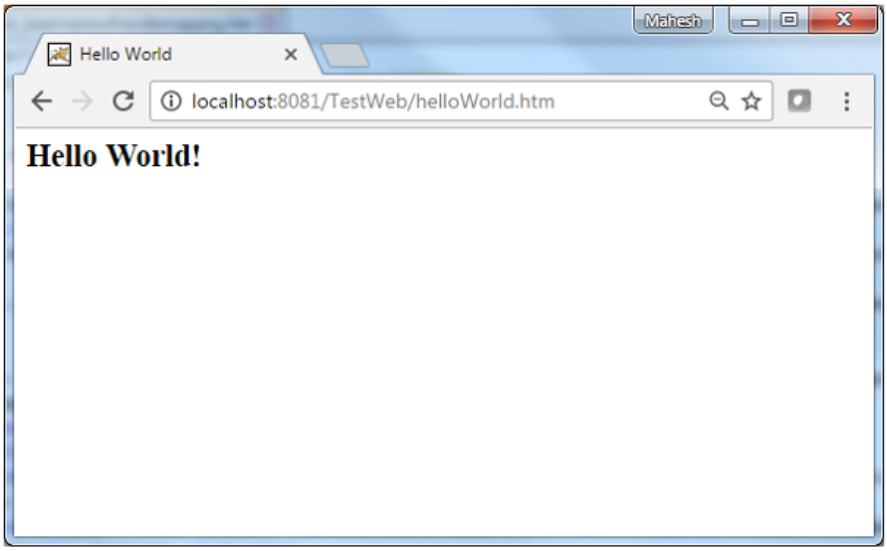
welcome.jsp

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

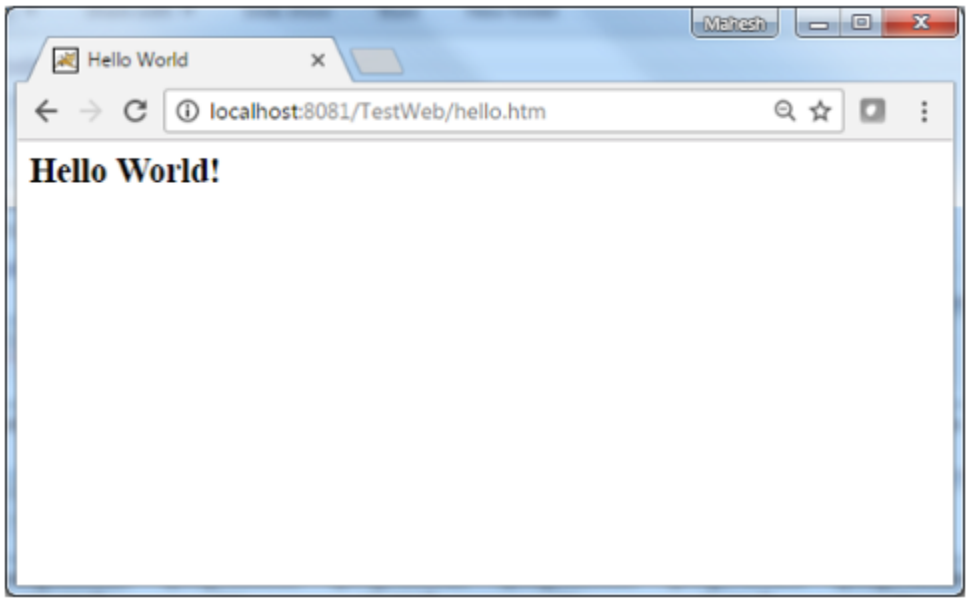
بعد از آن که کار فایل های پیکربندی و منبع تمام شد، برنامه ی خود را اکسپورت کنید. بر روی برنامه ی خود کلیک راست کنید، از گزینه ی Export → WAR File استفاده کنید و فایل TestWeb.war را داخل پوشه ی webapps متعلق به Tomcat ذخیره کنید.

حالا سرور Tomcat را اجرا کنید و مطمئن شوید که از طریق پوشه ی webapps و با استفاده از یک مرورگر استاندارد می توانید به دیگر صفحات وب دسترسی پیدا کنید. حالا در صورت وارد کردن آدرس

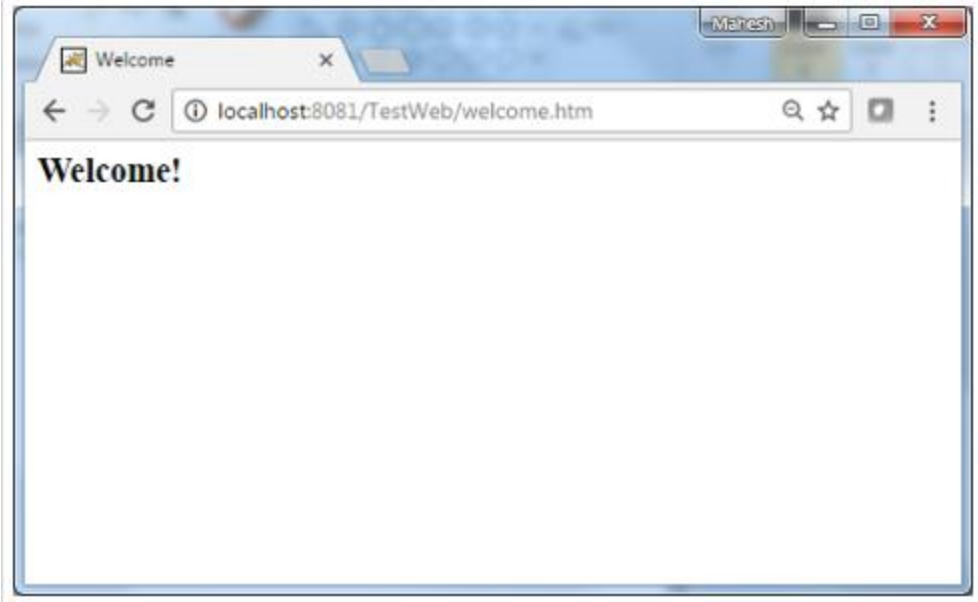
http://localhost:8080/ TestWeb/ helloWorld.htm و در صورت نبود مشکل در برنامه ی Spring Web صفحه ی زیر نمایش داده می شود.



آدرس http://localhost:8080/TestWeb/hello.htm را امتحان کنید. در صورتی که در برنامه ی Spring Web مشکلی وجود نداشته باشد، صفحه ی زیر نمایش داده می شود.



آدرس http://localhost:8080/TestWeb/welcome.htm را امتحان کنید. در صورتی که در برنامه ی Spring Web مشکلی وجود نداشته باشد، صفحه ی زیر نمایش داده می شود.



آدرس http://localhost:8080/TestWeb/Welcome.htm را امتحان کنید. در صورتی که در برنامه ی Spring Web مشکلی وجود نداشته باشد، صفحه ی زیر نمایش داده می شود.