

SpringMVC的HelloWorld

-使用XML配置

SpringMVC的HelloWorld (XML)

□开发步骤






- 1.添加必需的jar包
- 2.创建Controller
- 3.在web.xml中配置SpringMVC前端控制器
- 4.创建并配置SpringMVC的配置文件
- 5.创建视图
- 6.添加项目进web容器，启动并进行测试

SpringMVC的HelloWorld (XML)




□开发步骤

➤ 1.添加必需的jar包

📁 spring4Core

- ▷  commons-logging-1.1.3.jar - E:\课
- ▷  spring-beans-4.1.8.RELEASE.jar - E
- ▷  spring-context-4.1.8.RELEASE.jar -
- ▷  spring-core-4.1.8.RELEASE.jar - E:\
- ▷  spring-expression-4.1.8.RELEASE.j

📁 Referenced Libraries

- ▷  spring-web-4.1.8.RELEASE.jar
- ▷  spring-webmvc-4.1.8.RELEASE.jar
- ▷  spring-aop-4.1.8.RELEASE.jar

SpringMVC的HelloWorld (XML)

□开发步骤

- 2.创建HelloWorldControl类
- HelloWorldControl等价于Struts2中的 Action ,
handleRequest() 相当于 execute() 方法

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

public class HelloWorldControl implements Controller{

    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        //返回视图标识，视图控制器将根据标识获取资源响应
        return new ModelAndView("success");
    }
}
```

SpringMVC的HelloWorld (XML)

- **jsp 文件的名字，是根据我们在 HelloWorldControl 中的具体业务返回的字符串决定**
- **“ success ” 或者 其他任意自定义字符串**

SpringMVC的HelloWorld (XML)

□开发步骤

➤3.配置SpringMVC前端控制器

```
<servlet>
  <servlet-name>MVC</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <!--指定配置文件的位置-->
    <param-name>contextConfigLocation</param-name>
    <!--SpringMVC配置文件的路径及文件名 -->
    <param-value>classpath:spring-mvc.xml</param-value>
  </init-param>
  <!--服务器启动即加载-->
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>MVC</servlet-name>
  <!--□也可如struts2实现*.do之类的拦截，"/"表示对所有请求进行拦截 -->
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

SpringMVC的HelloWorld (XML)

□开发步骤

➤4.创建并配置SpringMVC的配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.1.xsd">

  <!--配置Controller类 -->
  <bean name="helloWorldController" class="com.springmvc.control.HelloWorldController">
```

SpringMVC的HelloWorld (XML)

```
<!-- 配置 HandleMapping 组件，用于实现请求与处理器之间的映射 -->
<bean id="urlMapping"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="helloWorldController">helloWorldController</prop>
    </props>
  </property>
</bean>
```


SpringMVC的HelloWorld (XML)

```
<!-- 定义 ViewResolver 组件，实现根据视图标识获取 JSP 响应 -->
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <!-- View 中用到的相关技术 -->
  <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView"></property>
  <!-- 后缀 -->
  <property name="suffix" value=".jsp"></property>
  <!-- 前缀 -->
  <property name="prefix" value="/"></property>
</bean>
</beans>
```

SpringMVC的HelloWorld (XML)

如此设置“前缀”和“后缀”表明，

ViewResolver 对象将到 WEB-INF/jsp 目录下找*.jsp 文件

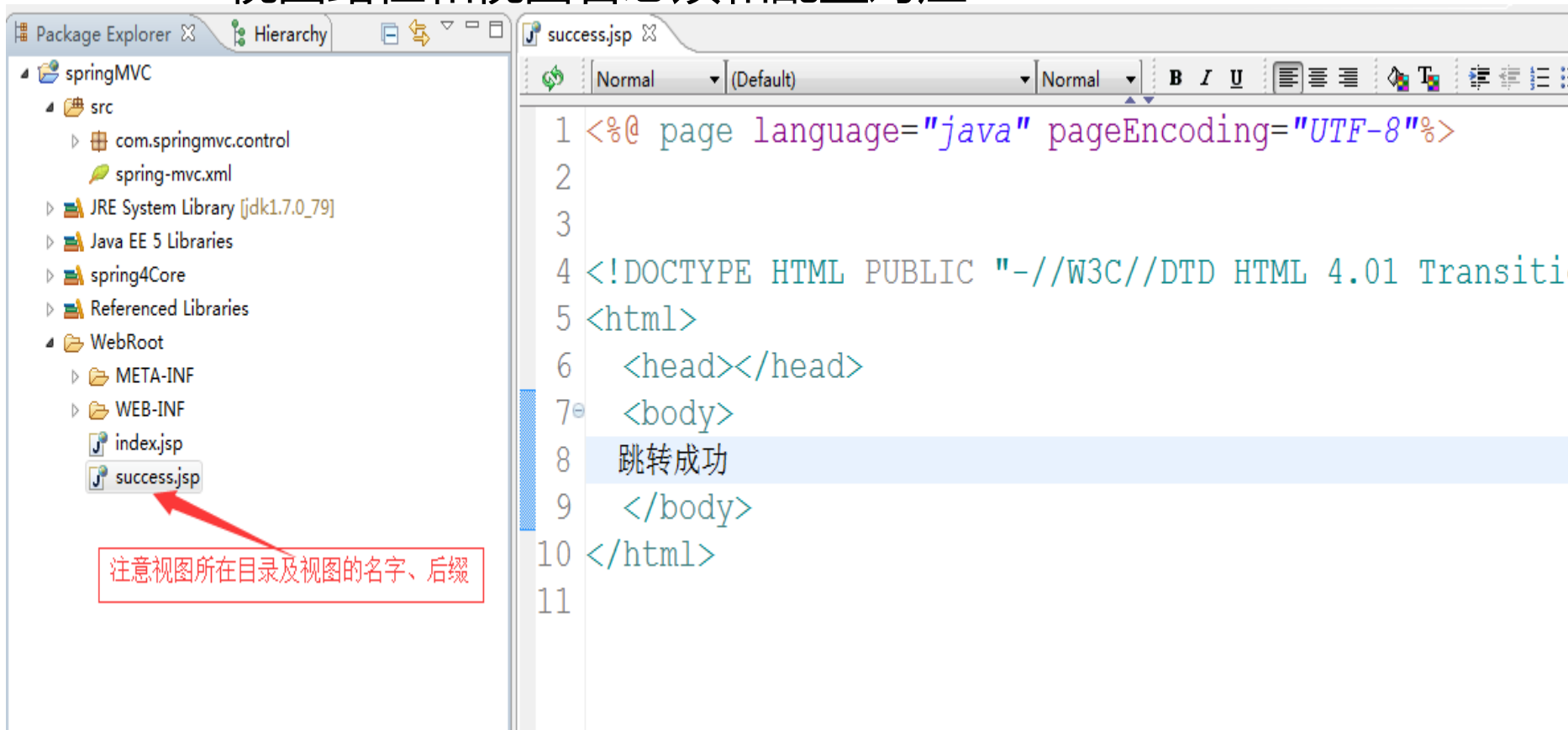
```
22 <bean id="viewResolver" class="org.springframework.w
23     <property name="viewClass" value="org.springfram
24     <property name="suffix" value=".jsp" </property>
25     <property name="prefix" value="WEB-INF/jsp"><
26 </bean>
27
```

SpringMVC的HelloWorld (XML)

开发步骤

➤ 4. 创建视图

➤ 视图路径和视图名必须和配置对应



The screenshot displays an IDE interface with two main panels. The left panel, titled 'Package Explorer', shows the project structure for 'springMVC'. Under the 'src' folder, there is a 'com.springmvc.control' package containing a 'spring-mvc.xml' file. Below this, the 'WebRoot' folder is expanded, showing 'META-INF', 'WEB-INF', 'index.jsp', and 'success.jsp'. A red arrow points from a text box to the 'success.jsp' file. The right panel shows the content of 'success.jsp', which is a JSP page with the following code:

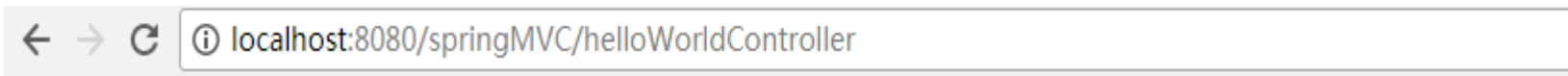
```
1 <%@ page language="java" pageEncoding="UTF-8"%>
2
3
4 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional
5 <html>
6   <head></head>
7   <body>
8     跳转成功
9   </body>
10 </html>
11
```

注意视图所在目录及视图的名字、后缀

SpringMVC的HelloWorld (XML)

□开发步骤

- 6.添加项目进web容器，启动并进行测试



跳转成功

SpringMVC的HelloWorld (XML)

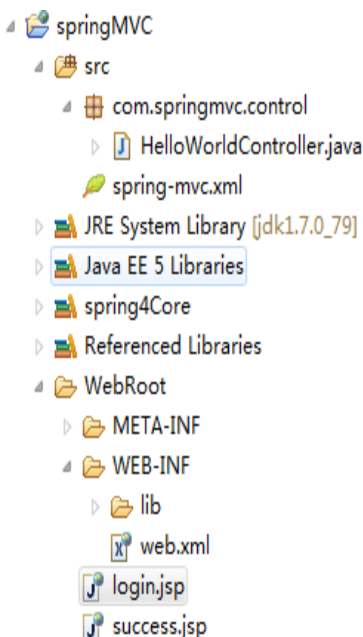
□如果有需要 将数据传入下一个视图的需求 的话，我们借助于 ModelMap ， 我们就可以传参数

□ModelMap

ModelMap对象主要用于传递控制方法处理数据到结果页面，也就是说我们把结果页面上需要的数据放到ModelMap对象中即可，他的作用类似于request对象的setAttribute方法的作用，用来在一个请求过程中传递处理的数据。

SpringMVC的HelloWorld

□1.添加页面login.jsp



```
5 %>
6
7 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
8 <html>
9   <head>
10     <title>登陆页面</title>
11   </head>
12
13   <body>
14     <form action="helloWorldController" method="post">
15       <label for="username">请输入姓名</label>
16       <input id="username" name="username">
17       <input type="submit" value="登陆">
18     </form>
19
```

SpringMVC的HelloWorld (XML)

□2.修改HelloWorldController

```
public ModelAndView handleRequest(HttpServletRequest request,
    HttpServletResponse response) throws Exception {
    //获取用户姓名
    String username = request.getParameter("username");
    //创建ModelMap对象
    ModelMap map = new ModelMap();
    //将需要传递的参数添加入ModelMap对象中,并将ModelMap对象放入ModelAndView
    map.put("username", username);

    //返回视图标识,视图控制器将根据标识获取资源响应
    return new ModelAndView("success", map);
}
```

SpringMVC的HelloWorld (XML)

- 3.修改success.jsp
- 使用EL表达式获取username

```
<body>  
    跳转成功， 欢迎${username}  
</body>
```

- <http://localhost:8080/springMVC/login.jsp>
- 输入用户名进行访问

SpringMVC的HelloWorld (XML)

- 有一点还需要了解，默认是采用 dispatcher 方式跳转，
- 可以这样设置为 redirect 方式：

```
return new  
ModelAndView("redirect:/success.jsp",map);
```

参数传递方式会发生变化

<http://localhost:8080/springMVC/success.jsp?username=sass>

SpringMVC的HelloWorld (XML)

□需要注意的是

□HandleMapping 组件 有很多的实现技术，我们使用的是较为简单的

`org.springframework.web.servlet.handler.Simple
UrlHandlerMapping`

□ViewResolver 组件 也有很多的实现技术，我们使用

`org.springframework.web.servlet.view.InternalRes
ourceViewResolver`

SpringMVC的HelloWorld (XML)

□此外，在 ViewResolver 组件中使用的主要解析技术是 JstlView (以预防 && 配合页面使用 jstl 标签的情况)

`org.springframework.web.servlet.view.JstlView`

使用注解配置的HelloWorld

□使用注解配置HelloWorld，能够更加的简单，也是目前使用最广泛的。

使用注解配置的HelloWorld

□开发步骤

- 1.添加必需的jar包（同XML配置）
- 2.创建Controller
- 3.在web.xml中配置SpringMVC前端控制器（同XML配置）
- 4.创建并配置SpringMVC的配置文件（servlet-handler、注解扫描、视图映射等）
- 5.创建视图（同XML配置）
- 6.添加项目进web容器，启动并进行测试

测试项目结构

- springMVC
 - src
 - com.springmvc.control
 - HelloWorldController.java
 - spring-mvc.xml
 - JRE System Library [jdk1.7.0_79]
 - Java EE 5 Libraries
 - spring4Core
 - Referenced Libraries
 - WebRoot
 - META-INF
 - WEB-INF
 - lib
 - web.xml
 - login.jsp
 - success.jsp

使用注解配置的HelloWorld

□2.创建Controller类

□使用注解法，无需实现

org.springframework.web.servlet.mvc.Controller
r接口，但须使用@Controller对类进行注解，并对需要访问的方法使用@RequestMapping注解进行地址映射

使用注解配置的HelloWorld

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HelloWorldController{

    //当发起 项目名/helloWorld的访问时，进入此方法处理
    @RequestMapping(value="helloWorld")
    public String HelloWorld(){
        return "success";
    }
}
```


使用注解配置的HelloWorld

□注意：

□1. 首先要在类的前面添加 “@Controller” 注解，表示是spring的控制器，这里会写一个方法**HelloWorld ()**

□2. **HelloWorld**方法上方有一个@RequestMapping， 是用于匹配请求的路径，比如这里匹配的请求路径就是“http://localhost:8080/springMVC/helloworld”，即当tomcat服务启动后，在浏览器输入这个url时，如果在这个方法打断点了，就会跳入该方法。

使用注解配置的HelloWorld

- 3. 这个return的结果不是乱写的，这个返回的字符串就是与下面spring-mvc.xml中配置的请求前后缀进行配合的，spring-mvc.xml中声明了prefix和suffix，而夹在这两者之间的就是这里返回的字符串，所以执行完这个方法后，我们可以得到这样的请求资源路径
“/success.jsp”，这个success.jsp是需要我们新建的

使用注解配置的HelloWorld

□4.创建并配置SpringMVC的配置文件（servlet-handler、注解扫描、视图映射等）

```
<!-- 配置springMV自动扫描-->
<context:component-scan base-package="com.springmvc.control" />

<!-- 加载springMVC默认的servlet-handler（访问路径）-->
<mvc:default-servlet-handler />

<!-- 加载springMVC注解驱动 -->
<mvc:annotation-driven></mvc:annotation-driven>

<!--配置请求的前后缀 -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/"></property>
    <property name="suffix" value=".jsp"></property>
</bean>
```

使用注解配置的HelloWorld

□6.添加项目进web容器，启动并进行测试

① localhost:8080/springMVC/helloWorld

谢谢！