

# SpringMVC 公开课

笔记

高浩阳

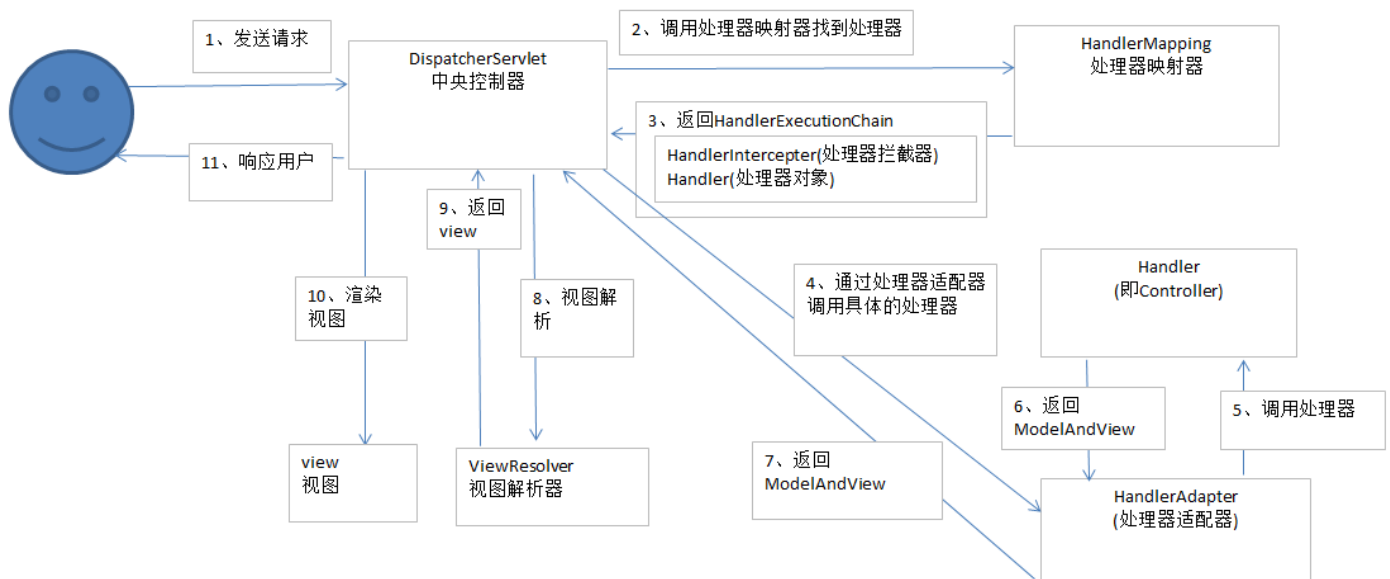
2014-11-29

## 目录

1	SpringMVC 框架.....	2
1.1	SpringMVC 框架.....	2
1.2	SpringMVC 组件总结 .....	2
2	开发 SpringMVC 的第一个程序.....	3
2.1	准备环境 .....	3
2.2	开发 SpringMVC 的第一个程序.....	3
2.2.1	创建 Java Web 工程 .....	3
2.2.2	向工程中填充 SpringMVC 的 jar 包.....	3
2.2.3	配置前端控制器.....	4
2.2.4	创建一个配置文件 springmvc.xml(名称不固定) .....	7
2.2.5	配置处理器映射器 HandlerMapping.....	8
2.2.6	配置处理器适配器 HandlerAdapter.....	9
2.2.7	配置视图解析器 ViewResolver .....	12
2.2.8	编写 Handler.....	13
2.2.9	在 springmvc.xml 中配置 helloAction.java.....	16
2.2.10	将工程部署到 tomcat , 启动 tomcat .....	17
2.3	小结 : .....	19
3	注解开发第一个例子 .....	19
3.1	新建工程 SpringMVCTest02 .....	19
3.2	在 springmvc.xml 中配置 .....	20
3.3	开发 action .....	21
3.4	配置 action .....	22
3.5	部署工程 , 运行 Tomcat.....	24
4	注解开发学生信息管理功能.....	24
5	SpringMVC 特点.....	33
6	和 JQuery easyui 整合完成数据列表 .....	33

# 1 SpringMVC 框架

## 1.1 SpringMVC 框架



1. 用户发起请求 request ( 比如请求链接叫 <http://www.xxx/user.action> ) 注册用户信息。
2. SpringMVC 通过 DispatcherServlet 接受请求。  
DispatcherServlet 是一个前端控制器 ( 想到 struts2 在 web.xml 配置一个 filter 前端控制器 ) 相当于控制器 Controller
3. DispatcherServlet 调用 HandlerMapping( 处理器映射器 ) 根据 user.action 找到处理器( Handler )  
HandlerMapping 负责根据 user.action 这个链接找到 Handler 根据 xml 配置或注解配置找到 Handler
4. HandlerMapping 将找到的 Handler 给 DispatcherServlet 前端控制器
5. DispatcherServlet 前端控制器调用 HandlerAdapter ( 处理器适配器 ) 去执行 Handler  
处理器适配器负责执行 Handler
6. Handler 将处理结果返回给 HandlerAdapter ( 处理器适配器 )  
处理结果就是 ModelAndView ( 封装了模型数据和视图 )
7. DispatcherServlet 调用视图解析器 ViewResolver 去解析视图
8. 将 View 给用户相应

## 1.2 SpringMVC 组件总结

1. DispatcherServlet 前端控制器 ( 不需要程序员写 )  
负责框架调度，相当于中央处理器  
基本 controller 控制器功能：  
**接收用户 request 请求和给用户 response 响应**
2. HandlerMapping ( 处理器映射器 ) ( 不需要程序员写 )  
负责根据 action 的连接找到 Handler 处理器 ( 理解成写的 action )

### 3. HandlerAdapter ( 处理器适配器 ) ( 不需要程序员写 )

负责去执行 Handler

### 4. \*\*Handler 处理器 需要程序员写

理解成 struts 里边的 action , 需要程序员写 action 类 , 这个 action 类符合适配器的执行规则。

### 5. ViewResolver ( 视图解析器 ) ( 不需要程序员写 )

负责将模型数据填充到 View

### 6. \*\*View 视图

需要程序员写 jsp 页面

## SpringMVC 是什么 ?

SpringMVC 和 struts 一样是一个表现层的框架。

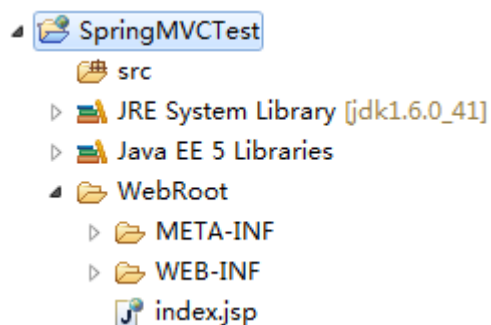
SpringMVC 是 Spring 的一个模块。

## 2 开发 SpringMVC 的第一个程序

### 2.1 准备环境

### 2.2 开发 SpringMVC 的第一个程序

#### 2.2.1 创建 Java Web 工程



#### 2.2.2 向工程中填充 SpringMVC 的 jar 包

包括 :

Spring 的 jar 包

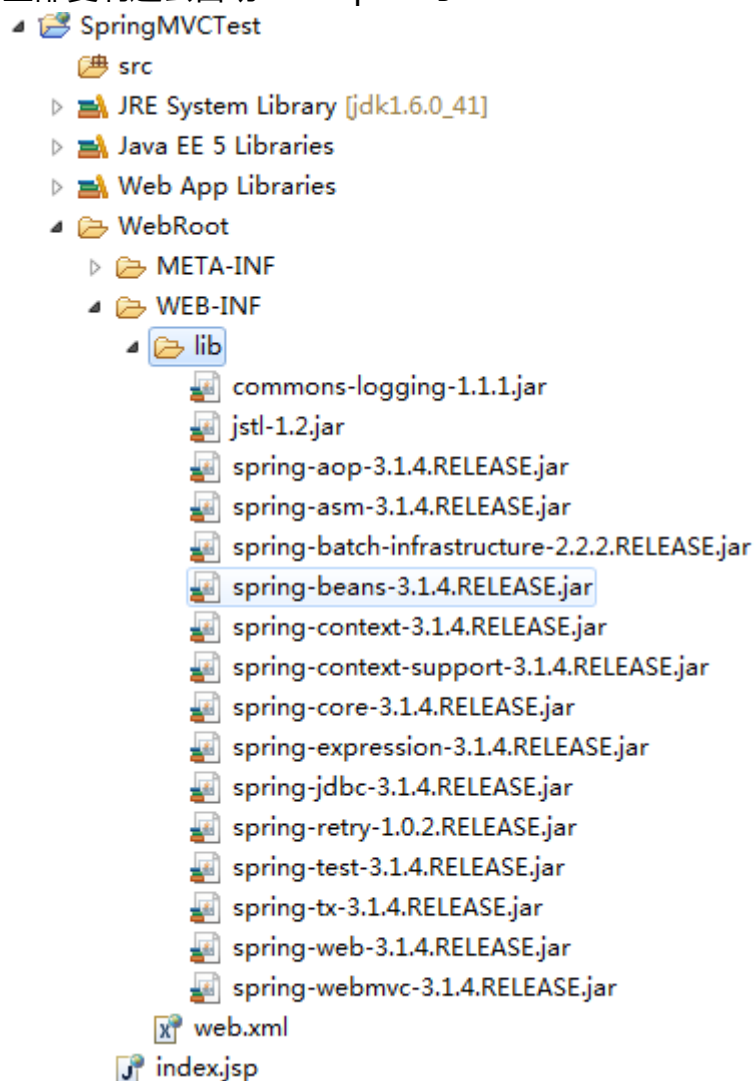
SpringMVC 模块的 jar

依赖的 jar

D:\BaiduYunDownload\spring mvc 公开课课件代码\spring mvc 公开课课件代码\jar 包  
\springmvc

JavaWeb 工程自身有一个 lib 文件

全部复制进去自动 build path 了

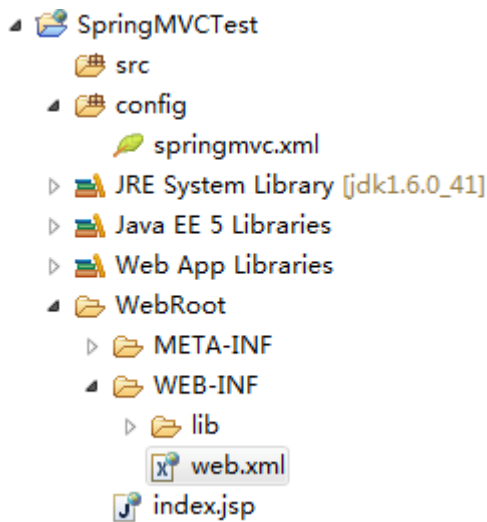


### 2.2.3 配置前端控制器

在 web.xml 中配置

前端控制器是一个 Servlet

在 web.xml



## web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <display-name>springMVCTest</display-name>

  <!-- 前端控制器 -->
  <servlet>
    <servlet-name>springmvc</servlet-name>

    <servlet-class>org.springframework.web.servlet.DispatcherServlet.class</servlet-class>

    <!-- 加载springmvc.xml配置文件 -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <!-- 请求链接以.action结尾由DispatcherServlet进行解析 -->
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

其中：

`<servlet-class>org.springframework.web.servlet.DispatcherServlet.class</servlet-class>`

在

- SpringMVCTest
  - src
  - config
  - JRE System Library [jdk1.6.0\_41]
  - Java EE 5 Libraries
  - Web App Libraries
    - commons-logging-1.1.1.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - jstl-1.2.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-aop-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-asm-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-batch-infrastructure-2.2.2.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-beans-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-context-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-context-support-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-core-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-expression-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-jdbc-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-retry-1.0.2.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-test-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-tx-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-web-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - spring-webmvc-3.1.4.RELEASE.jar - D:\workspace\SpringMVCTest\WebRoot\WEB-INF\lib
    - org.springframework.web.servlet
      - DispatcherServlet.class
      - FlashMap.class
      - FlashMapManager.class
      - FrameworkServlet.class
      - HandlerAdapter.class
      - HandlerExceptionResolver.class
      - HandlerExecutionChain.class
      - HandlerInterceptor.class
      - HandlerMapping.class
      - HttpServletBean.class
      - LocaleResolver.class
      - ModelAndView.class
      - ModelAndViewDefiningException.class
      - package-info.class
      - RequestToViewNameTranslator.class
      - ResourceServlet.class
      - SmartView.class
      - ThemeResolver.class
      - View.class
      - ViewRendererServlet.class
      - ViewResolver.class
      - DispatcherServlet.properties

右击 DispatcherServlet.class/Copy Qualified Name , 再粘贴。

这里面有一个错误导致后来部署后没有发布成功，类找不到错误。  
先按下不表。

#### 2.2.4 创建一个配置文件 springmvc.xml(名称不固定)

springmvc.xml 是 SpringMVC 的一个全局配置文件，配置

处理器映射器 HandlerMapping

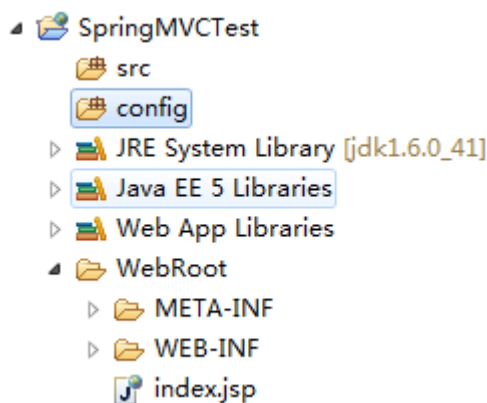
处理器适配器 HandlerAdapter

视图解析器 ViewResolver

编写的 Handler

新建 source 文件 config

右击工程/new/source folder



在 config 里新建 springmvc.xml

springmvc.xml 中的文件头

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:aop="http://www.springframework.org/schema/aop"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
            http://www.springframework.org/schema/mvc
            http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-3.1.xsd
            http://www.springframework.org/schema/aop
            http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
            http://www.springframework.org/schema/tx
            http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

</beans>
```



## springmvc.xml 中要配置的内容

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:aop="http://www.springframework.org/schema/aop"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
            http://www.springframework.org/schema/mvc
            http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-3.1.xsd
            http://www.springframework.org/schema/aop
            http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
            http://www.springframework.org/schema/tx
            http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

    <!-- 处理器映射器HandlerMapping -->

    <!-- 处理器适配器HandlerAdapter -->

    <!-- 视图解析器ViewResolver -->

</beans>
```

### 2.2.5 配置处理器映射器 HandlerMapping

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:aop="http://www.springframework.org/schema/aop"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
            http://www.springframework.org/schema/mvc
            http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-3.1.xsd
            http://www.springframework.org/schema/aop
            http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
            http://www.springframework.org/schema/tx
            http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

    <!-- 配置程序员编写的handler -->
    <bean name="/hello.action" class="" />
```

```

<!-- 处理器映射器HandlerMapping -->
<!-- 根据配置的Handler的bean的name(action的链接)进行查找Handler 将action的url配置在
bean的名称中 -->
<bean
    class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />

<!-- 处理器适配器HandlerAdapter -->

<!-- 视图解析器ViewResolver -->

</beans>

```

## 2.2.6 配置处理器适配器 HandlerAdapter

```

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.1.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

    <!-- 配置程序员编写的handler -->
    <bean name="/hello.action" class="" />

    <!-- 处理器映射器HandlerMapping -->
    <!-- 根据配置的Handler的bean的name(action的链接)进行查找Handler 将action的url配置在
    bean的名称中 -->
    <bean
        class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />

    <!-- 处理器适配器HandlerAdapter -->
    <!-- springmvc的框架根据接口判断是一个适配器，所有的适配器都要实现HandlerAdapter接口
    规定了Handler的编写规则，编写Handler实现Controller接口 -->
    <bean
        class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />

    <!-- 视图解析器ViewResolver -->

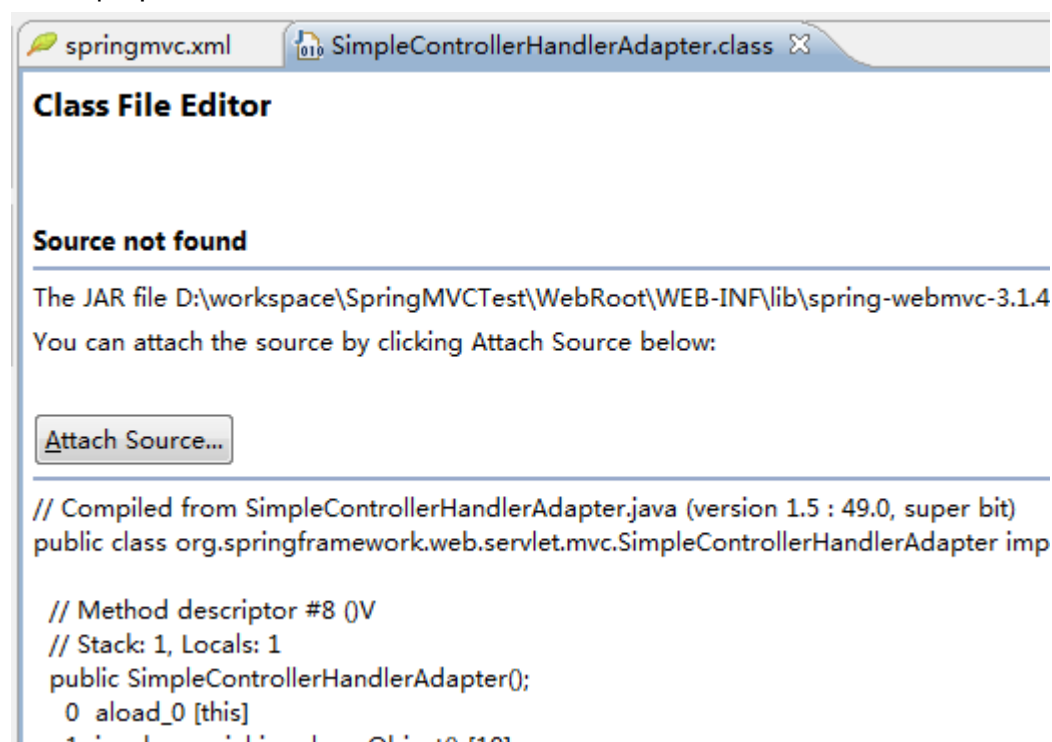
</beans>

```

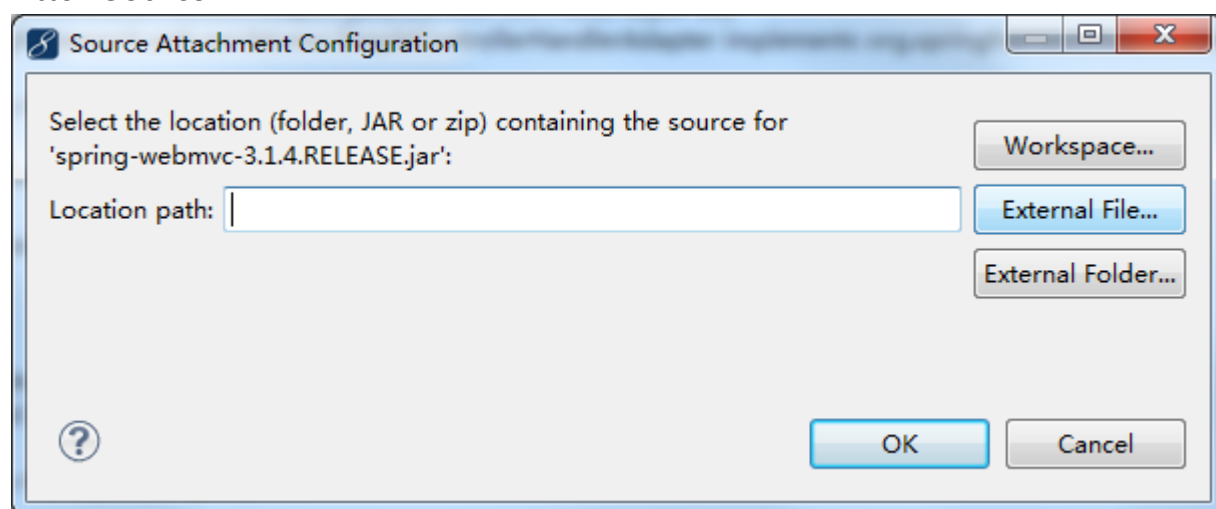
加载源码

```
<bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />
```

ctrl+单击

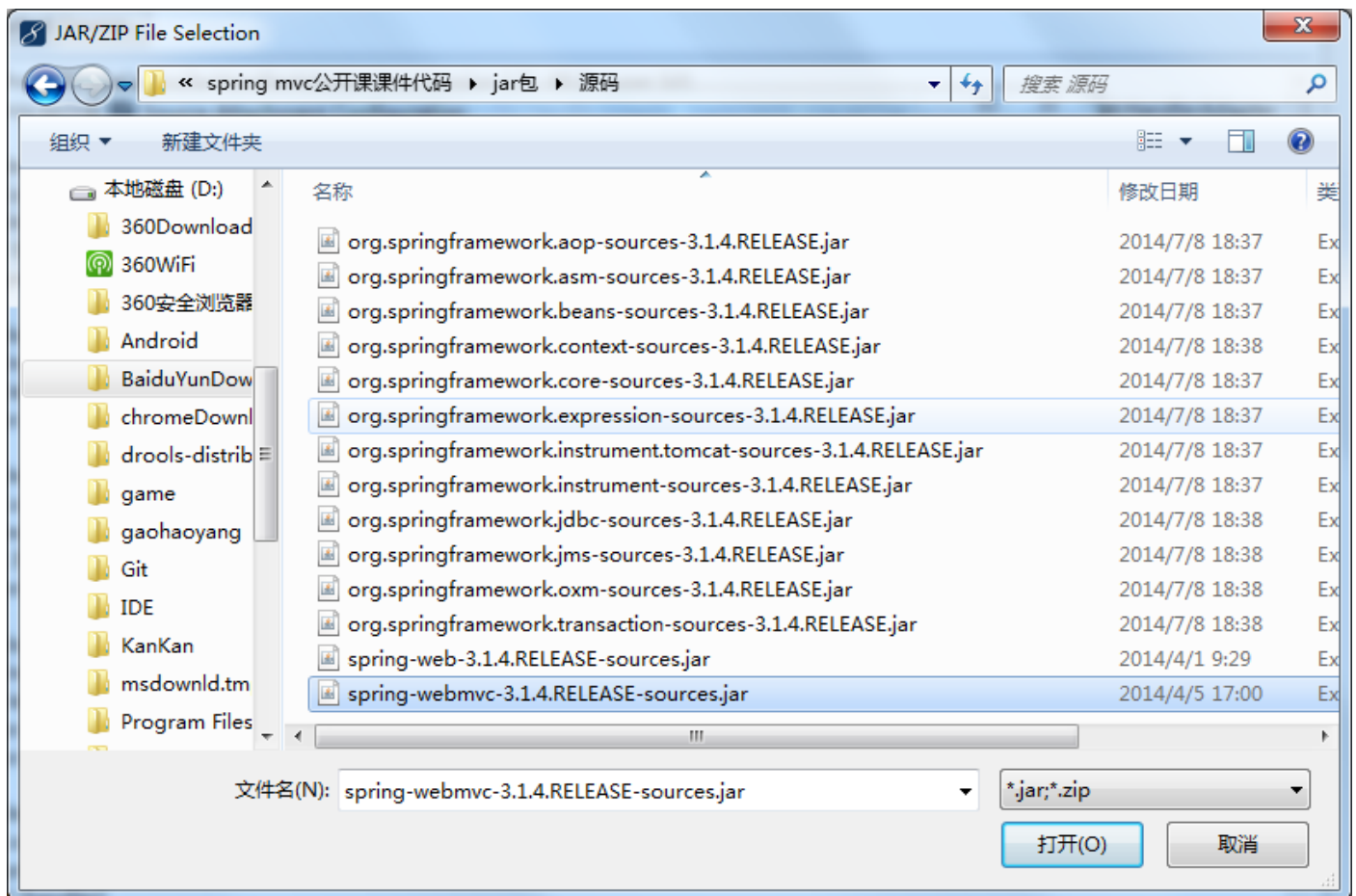


Attch Source



External File...

D:\BaiduYunDownload\spring mvc 公开课课件代码\spring mvc 公开课课件代码\jar 包\源码



## 源码

```

/*
 * Copyright 2002-2007 the original author or authors.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```
package org.springframework.web.servlet.mvc;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import org.springframework.web.servlet.HandlerAdapter;
```

```
import org.springframework.web.servlet.ModelAndView;
```

```

/**
 * Adapter to use the plain {@link Controller} workflow interface with
 * the generic {@link org.springframework.web.servlet.DispatcherServlet}.
 * Supports handlers that implement the {@link LastModified} interface.
 *
 * <p>This is an SPI class, not used directly by application code.
 *
 * @author Rod Johnson
 * @author Juergen Hoeller
 * @see org.springframework.web.servlet.DispatcherServlet
 * @see Controller
 * @see LastModified
 * @see HttpRequestHandlerAdapter
 */
public class SimpleControllerHandlerAdapter implements HandlerAdapter {

    public boolean supports(Object handler) {
        return (handler instanceof Controller);
    }

    public ModelAndView handle(HttpServletRequest request, HttpServletResponse
response, Object handler)
        throws Exception {

        return ((Controller) handler).handleRequest(request, response);
    }

    public long getLastModified(HttpServletRequest request, Object handler) {
        if (handler instanceof LastModified) {
            return ((LastModified) handler).getLastModified(request);
        }
        return -1L;
    }

}

```

### 2.2.7 配置视图解析器 ViewResolver

```

<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:aop="http://www.springframework.org/schema/aop"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
            http://www.springframework.org/schema/mvc

```

```

    http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.1.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

<!-- 配置程序员编写的handler -->
<bean name="/hello.action" class="" />

<!-- 处理器映射器HandlerMapping -->
<!-- 根据配置的Handler的bean的name(action的链接)进行查找Handler 将action的url配置在
bean的name中 -->
<bean
    class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />

<!-- 处理器适配器HandlerAdapter -->
<!-- springmvc的框架根据接口判断是一个适配器，所有的适配器都要实现HandlerAdapter接口 规
定了Handler的编写规则，编写Handler实现Controller接口 -->
<bean
    class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />

<!-- 视图解析器ViewResolver -->
<!-- 解析jsp，默认支持jstl -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>

</beans>

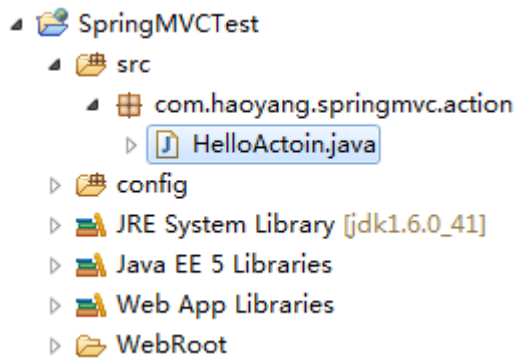
```

## 2.2.8编写 Handler

按照适配器要求的规则来编写。

SimpleControllerHandlerAdapter 实现 Controller 接口

新建 Action , HelloAction.java



## HelloAction.java

```
package com.haoyang.springmvc.action;

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 HelloActoin implements Controller { //继承Controller接口

    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws Exception {

        //在页面上提示一行信息
        String message = "hello world!";

        //通过request对象将信息在页面上展示
        //request.setAttribute("message", message);

        ModelAndView modelAndView = new ModelAndView();
        // 相当于request.setAttribute(), 将数据传到页面展示
        //model数据
        modelAndView.addObject("message", message);
        //设置视图
        modelAndView.setViewName(viewName);

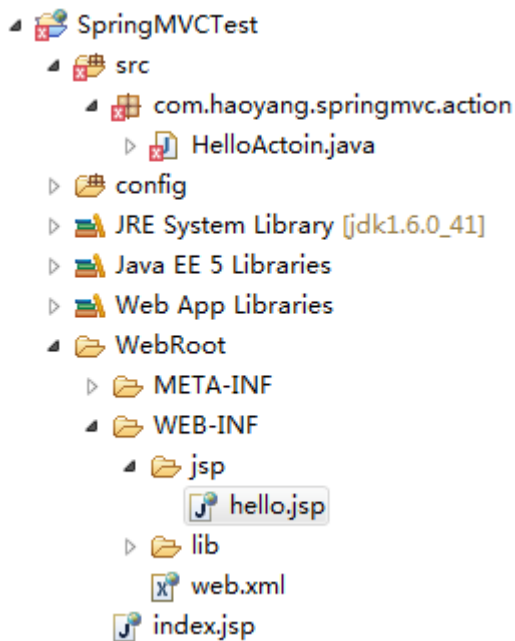
        return null;
    }
}
```

//设置视图

```
modelAndView.setViewName(viewName);
```

需要jsp 页面，新建jsp 页面

在工程 SpringMVCTest/WebRoot/WEB-INF/jsp/hello.jsp



hello.jsp 中

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/"
;
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">

<title>第一个SpringMVC程序</title>

<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->

</head>

<body>
    ${message}
</body>
```



</html>

`${message}`我现在也不知道是什么意思，应该是直接把它展示出来

## helloAction.java 补充完整

```
package com.haoyang.springmvc.action;

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 HelloActoin implements Controller { //继承Controller接口

    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws Exception {

        //在页面上提示一行信息
        String message = "hello world!";

        //通过request对象将信息在页面上展示
        //request.setAttribute("message", message);

        ModelAndView modelAndView = new ModelAndView();
        // 相当于request.setAttribute(), 将数据传到页面展示
        //model数据
        modelAndView.addObject("message", message);
        //设置视图
        modelAndView.setViewName("hello");//已经在springmvc.xml中添加了前后缀了，即
//WEB-INF/jsp/hello.jsp

        return modelAndView;//返回这个对象
    }
}
```

### 2.2.9在 springmvc.xml 中配置 helloAction.java

相当于让 Spring 管理 helloAction

Copy Qualified Name of HelloAction.java

在 springmvc.xml 中

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
```

```

xmlns:context="http://www.springframework.org/schema/context"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.1.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

<!-- 配置程序员编写的handler -->
<bean name="/hello.action" class="com.haoyang.springmvc.action.HelloActoin" />

<!-- 处理器映射器HandlerMapping -->
<!-- 根据配置的Handler的bean的name(action的链接)进行查找Handler 将action的url配置在
bean的名称中 -->
<bean
    class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />

<!-- 处理器适配器HandlerAdapter -->
<!-- springmvc的框架根据接口判断是一个适配器，所有的适配器都要实现HandlerAdapter接口 规
定了Handler的编写规则，编写Handler实现Controller接口 -->
<bean
    class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />

<!-- 视图解析器ViewResolver -->
<!-- 解析jsp，默认支持jstl -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>

</beans>

```

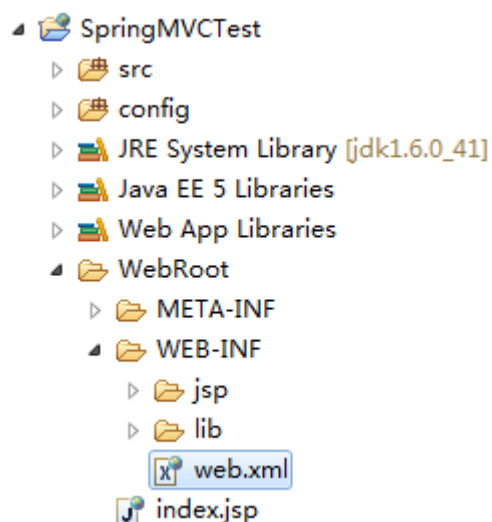
## 2.2.10 将工程部署到 tomcat , 启动 tomcat

访问路径：http://localhost:8080/SpringMVCTest/hello.action

失败！！！！

已找到问题所在。

在 web.xml 中



前端控制器的配置写错了

```
<!-- 前端控制器 -->
<servlet>
    <servlet-name>springmvc</servlet-name>

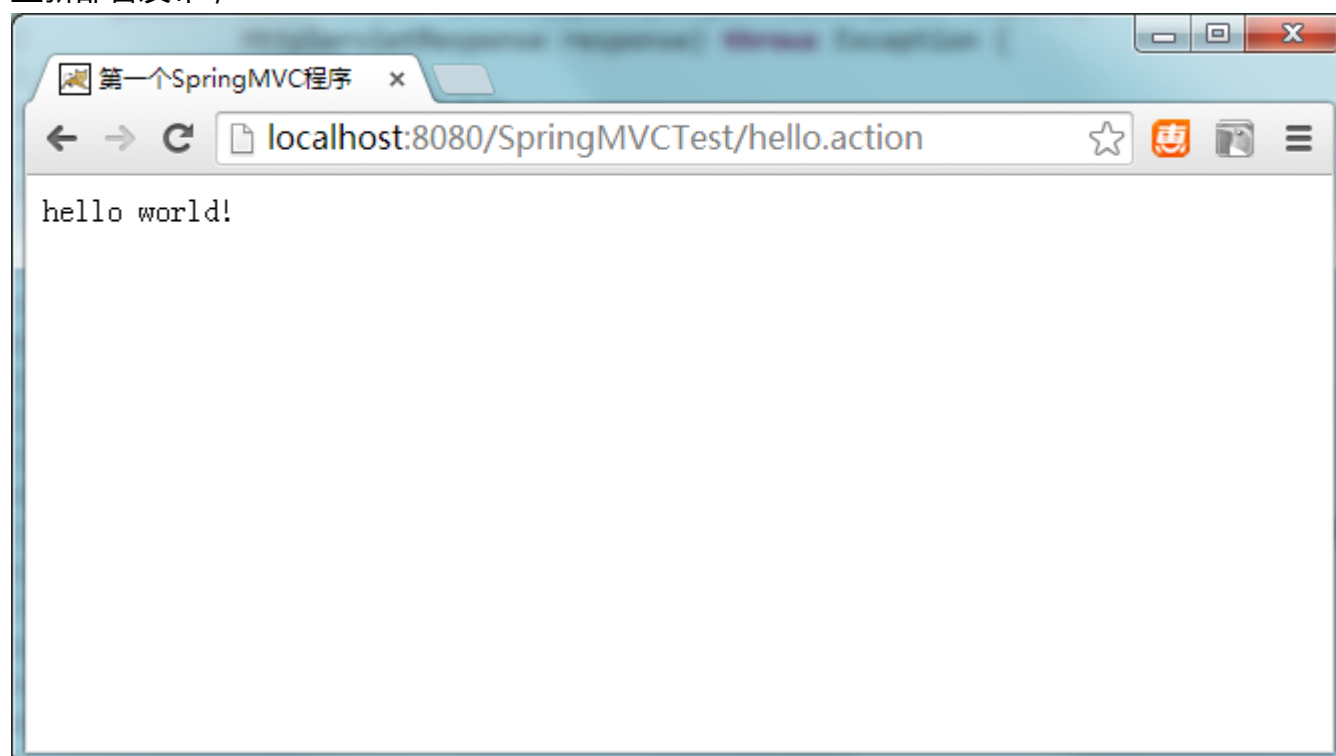
    <servlet-class>org.springframework.web.servlet.DispatcherServlet.class</servlet-class>
```

应该把.class 删掉，这样就好了

```
<!-- 前端控制器 -->
<servlet>
    <servlet-name>springmvc</servlet-name>

    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
```

重新部署发布，success



## 2.3 小结：

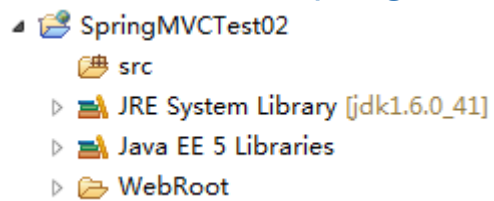
掌握 SpringMVC 开发的步骤。

使用实现 Controller 接口的方式开发存在问题：

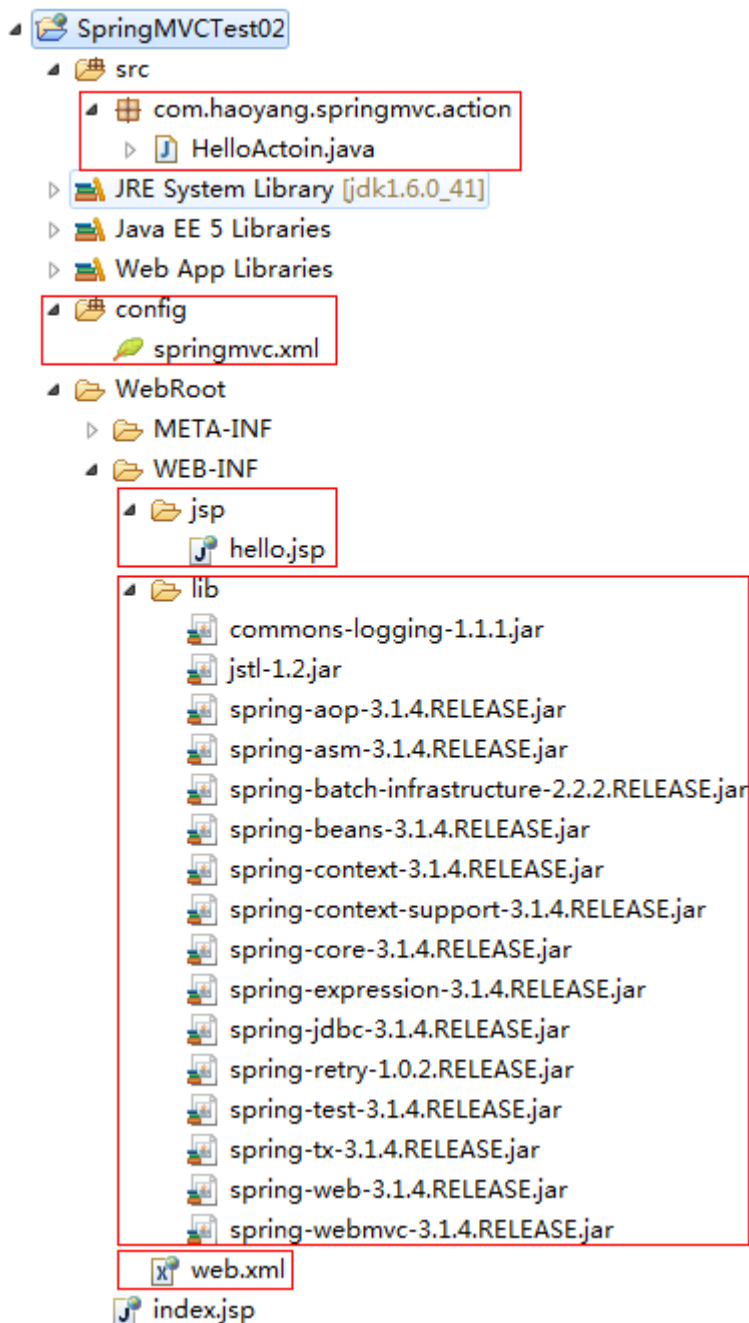
实现了 Controller 接口的 Action 类中只能写一个方法，不方便实际开发。

## 3 注解开发第一个例子

### 3.1 新建工程 SpringMVCTest02



将 lib, jsp, web.xml, config, com.haoyang.springmvc.action 包全部复制进来



### 3.2 在 springmvc.xml 中配置

修改 config 资源文件中的 springmvc.xml

处理器映射器 HandlerMapping，支持注解，通过注解找 action

处理器适配器 HandlerAdapter，支持注解，解析 action 中的各种注解，执行 action 类。

视图解析器，不用改

springmvc.xml 中

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:aop="http://www.springframework.org/schema/aop"
```

```

xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.1.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

<!-- 配置程序员编写的handler -->
<bean name="/hello.action" class="com.haoyang.springmvc.action.HelloActoin" />

<!--注解映射器HandlerMapping -->
<bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>

<!--注解适配器HandlerAdapter -->
<bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>

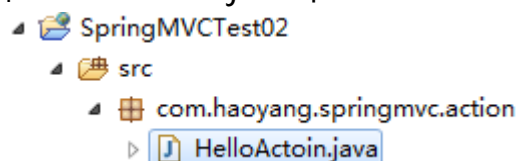
<!-- 视图解析器ViewResolver -->
<!-- 解析jsp, 默认支持jstl -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>

</beans>

```

### 3.3 开发 action

在 HelloAction.java 中



HelloAction.java

```

package com.haoyang.springmvc.action;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller:标记此类是一个Handler处理器
@Controller
public class HelloActoin {

    //在页面显示一行信息
    //RequestMapping:指定该方法对应的url链接
    @RequestMapping("/hello")
    public String hello(Model model) {

        String message = "hello world! gaohaoyang";

        //通过model将信息显示到页面
        model.addAttribute("message", message);

        //指定视图地址
        //逻辑视图名，根据视图解析器中的前缀和后缀加上逻辑视图名拼接视图的完整路径
        //拼接：前缀+逻辑视图名+后缀 = 视图的完整路径
        return "hello";
    }
}

```

### 3.4 配置 action

springmvc.xml 中

不需要再使用这个了，删除

```

<!-- 配置程序员编写的handler -->
<bean name="/hello.action" class="com.haoyang.springmvc.action.HelloActoin" />

```

使用组件扫描

```

<!-- 使用组件扫描 -->
<!-- 将action扫描出来，在spring容器中进行注册，自动对action在spring容器中进行配置 -->
<context:component-scan base-package="com.haoyang.springmvc.action" />

```

扫描@Component、@controller、@service、@repository的注解

注意：如果使用组件扫描则 controller 不需要在 springmvc.xml 中配置

完整 springmvc.xml

```

<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"

```

```
xmlns:context="http://www.springframework.org/schema/context"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.1.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

<!-- 使用组件扫描 -->
<!-- 将action扫描出来，在spring容器中进行注册，自动对action在spring容器中进行配置 -->
<context:component-scan base-package="com.haoyang.springmvc.action" />

<!--注解映射器HandlerMapping -->
<bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>

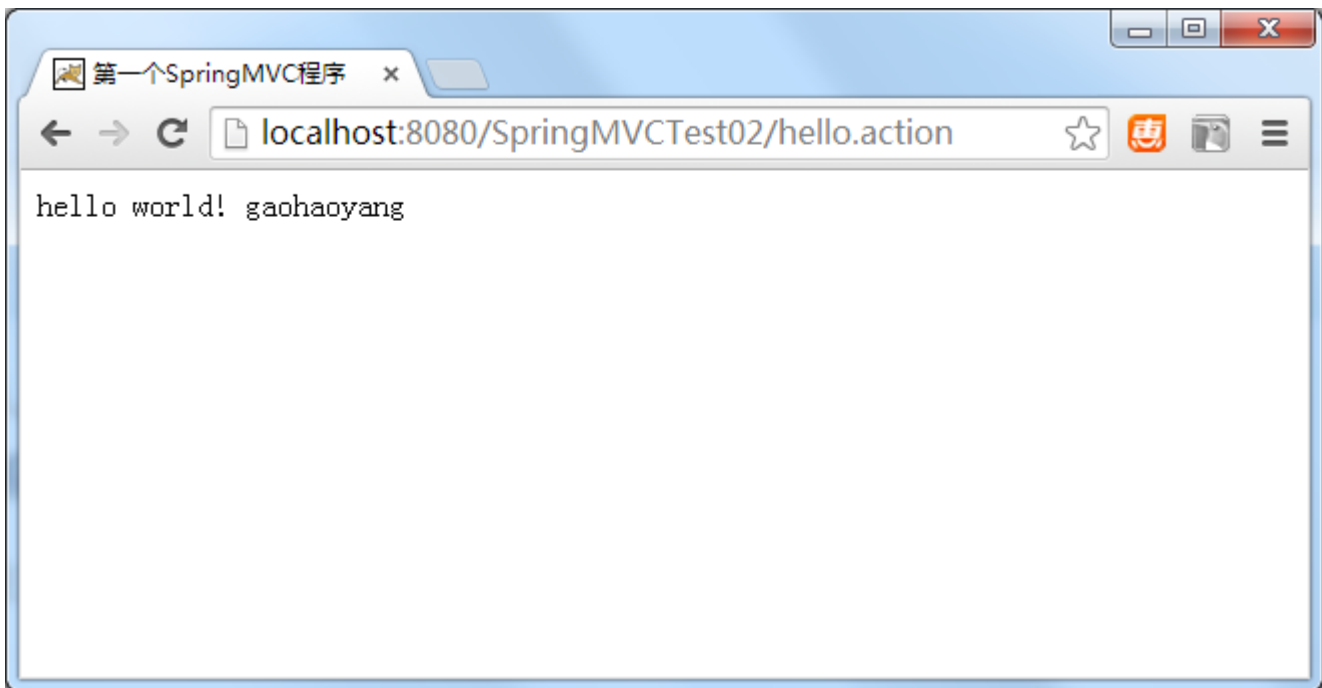
<!--注解适配器HandlerAdapter -->
<bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>

<!-- 视图解析器ViewResolver -->
<!-- 解析jsp，默认支持jstl -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>

</beans>
```



### 3.5 部署工程，运行 Tomcat



http://localhost:8080/SpringMVCTest02/hello.action  
success !

## 4 注解开发学生信息管理功能

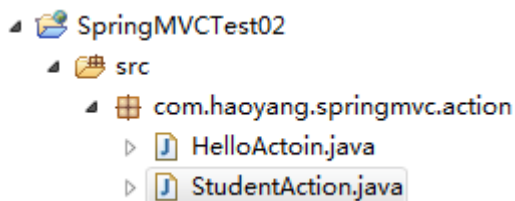
需求：

学生信息管理，包括查询学生信息、添加学生信息、修改学生信息、删除学生信息。

分析：

创建一个 StudentAction.java

编写各个功能的操作方法：查询学生信息、添加学生信息、修改学生信息、删除学生信息



StudentAction.java ( 未完成 )

```
package com.haoyang.springmvc.action;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

/**
 * 学生信息管理
```

```

* @author Administrator
*
*/
@Controller
public class StudentAction {

    //学生查询
    @RequestMapping("queryStudent")
    public String queryStudent(Model model) {

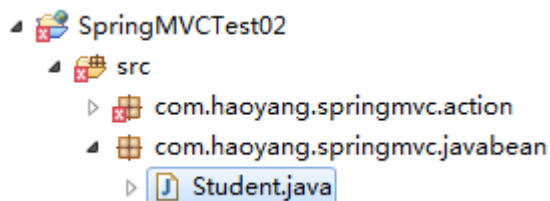
        //将学生信息显示到页面上
        //模拟静态数据
    }

    //学生修改
    //学生添加
}

```

pojo 指纯粹的 javabean,就是只有属性及其 set/get 方法 ,但是 javabean 可以是数据 bean,也可以是业务逻辑 bean

视频中使用的包名是 pojo , 这里我还是用 JavaBean 吧



## Student.java

```

package com.haoyang.springmvc.javabean;

/**
 * 学生信息类
 * @author Administrator
 *
 */
public class Student {
    private String username;
    private int age;
    private String address;
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
}

```

```

    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
}

```

回到 StudentAction.java ( 未完成 )

```

package com.haoyang.springmvc.action;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import com.haoyang.springmvc.javabean.Student;

/**
 * 学生信息管理
 * @author Administrator
 *
 */
@Controller
public class StudentAction {

    //学生查询
    @RequestMapping("queryStudent")
    public String queryStudent(Model model) {

        //将学生信息显示到页面上
        List<Student> list = new ArrayList<Student>();

        //模拟静态数据

        Student student1 = new Student();
        student1.setUsername("高浩阳");
    }
}

```

```

        student1.setAge(23);
        student1.setAddress("BJ");

        Student student2 = new Student();
        student2.setUsername("Yingying");
        student2.setAge(24);
        student2.setAddress("DL");

        Student student3 = new Student();
        student3.setUsername("HQXY");
        student3.setAge(21);
        student3.setAddress("WS");

        list.add(student1);
        list.add(student2);
        list.add(student3);

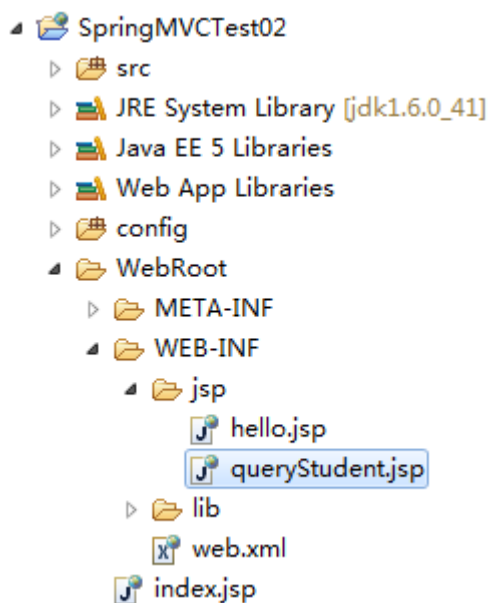
        model.addAttribute("studentList", list);

        //返回逻辑视图名
        return "queryStudent";
    }

    //学生修改
    //学生添加
}

```

## 创建一个jsp 页面



## 加载 jstl 标签，前缀为 c

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

## queryStudent.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/"
;
%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">

<title>学生信息查询</title>

<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->

</head>

<body>
<table width="100%" border="1">
<tr>
<td>姓名</td>
<td>年龄</td>
<td>地址</td>
</tr>

<c:forEach items="${studentList}" var="student">
<tr>
<td>${student.username }</td>
<td>${student.age }</td>
<td>${student.address }</td>
</tr>
</c:forEach>
</table>
```

```
</body>
</html>
```

## StudentAction.java中添加分链接 @RequestMapping("/student")

```
package com.haoyang.springmvc.action;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import com.haoyang.springmvc.javabean.Student;

/**
 * 学生信息管理
 * @author Administrator
 *
 */
@Controller
@RequestMapping("/student")
public class StudentAction {

    //学生查询
    @RequestMapping("queryStudent")
    public String queryStudent(Model model) {

        //将学生信息显示到页面上
        List<Student> list = new ArrayList<Student>();

        //模拟静态数据

        Student student1 = new Student();
        student1.setUsername("高浩阳");
        student1.setAge(23);
        student1.setAddress("BJ");

        Student student2 = new Student();
        student2.setUsername("Yingying");
        student2.setAge(24);
        student2.setAddress("DL");

        Student student3 = new Student();
        student3.setUsername("HQXY");
```

```

        student3.setAge(21);
        student3.setAddress("WS");

        list.add(student1);
        list.add(student2);
        list.add(student3);

        model.addAttribute("studentList", list);

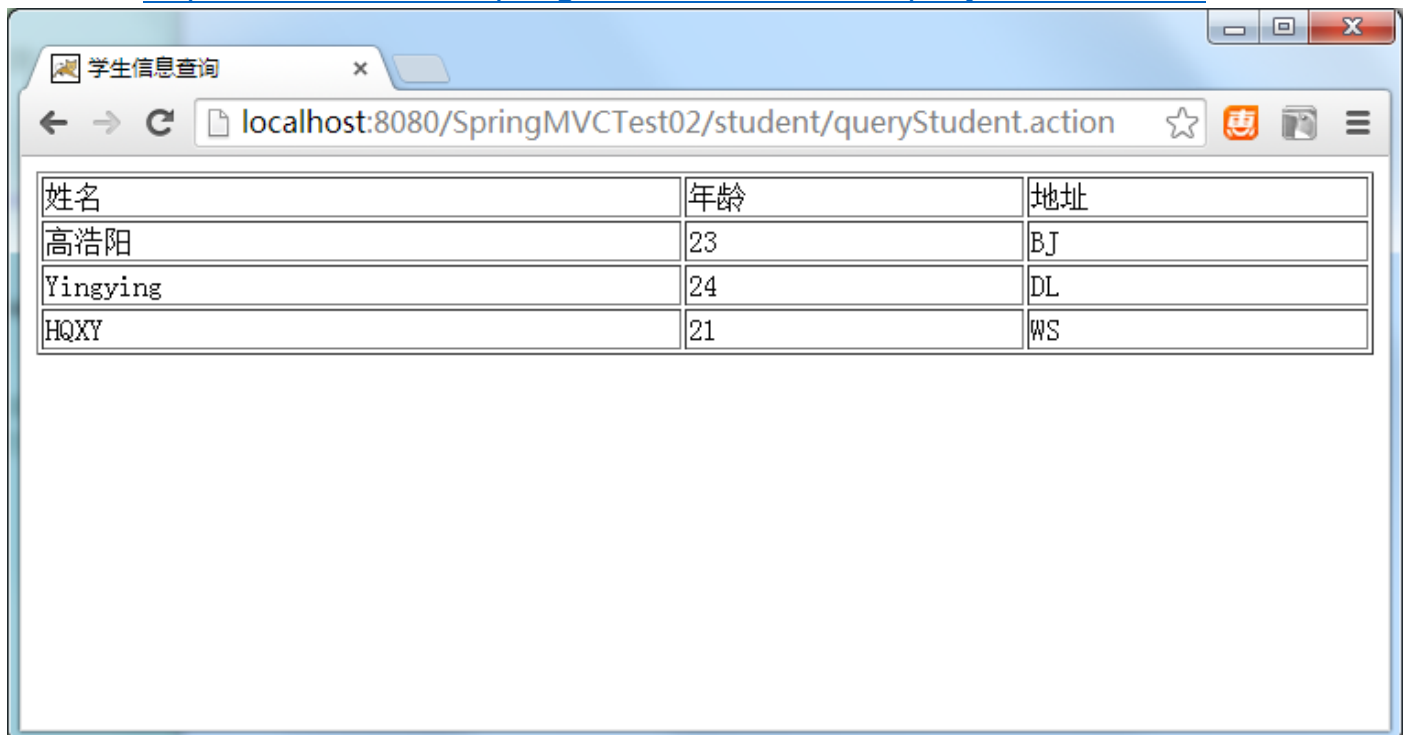
        //返回逻辑视图名
        return "queryStudent";
    }

    //学生修改
    //学生添加
}

```

部署发布

访问地址 <http://localhost:8080/SpringMVCTest02/student/queryStudent.action>



The screenshot shows a web browser window with the title '学生信息查询' (Student Information Query). The address bar displays 'localhost:8080/SpringMVCTest02/student/queryStudent.action'. The main content area contains a table with three columns: '姓名' (Name), '年龄' (Age), and '地址' (Address). The table lists three students: 高浩阳 (Gao Haoyang) with age 23 and address BJ, Yingying with age 24 and address DL, and HQXY with age 21 and address WS.

姓名	年龄	地址
高浩阳	23	BJ
Yingying	24	DL
HQXY	21	WS

以上是页面展示数据，现在从页面向后台传输数据

StudentAction.java 中添加

```

//学生查询
@RequestMapping("queryStudent")
public String queryStudent(Model model, String type) {

    //学生类型
}

```

```
System.out.println("type = "+type);
```

## StudentAction.java

```
package com.haoyang.springmvc.action;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import com.haoyang.springmvc.javabean.Student;

/**
 * 学生信息管理
 * @author Administrator
 *
 */
@Controller
@RequestMapping("/student")
public class StudentAction {

    //学生查询
    @RequestMapping("queryStudent")
    public String queryStudent(Model model, String type) {

        //学生类型
        System.out.println("type = "+type);

        //将学生信息显示到页面上
        List<Student> list = new ArrayList<Student>();

        //模拟静态数据

        Student student1 = new Student();
        student1.setUsername("高浩阳");
        student1.setAge(23);
        student1.setAddress("BJ");

        Student student2 = new Student();
        student2.setUsername("Yingying");
        student2.setAge(24);
        student2.setAddress("DL");

        Student student3 = new Student();
```



```

        student3.setUsername("HQXY");
        student3.setAge(21);
        student3.setAddress("WS");

        list.add(student1);
        list.add(student2);
        list.add(student3);

        model.addAttribute("studentList", list);

        //返回逻辑视图名
        return "queryStudent";
    }

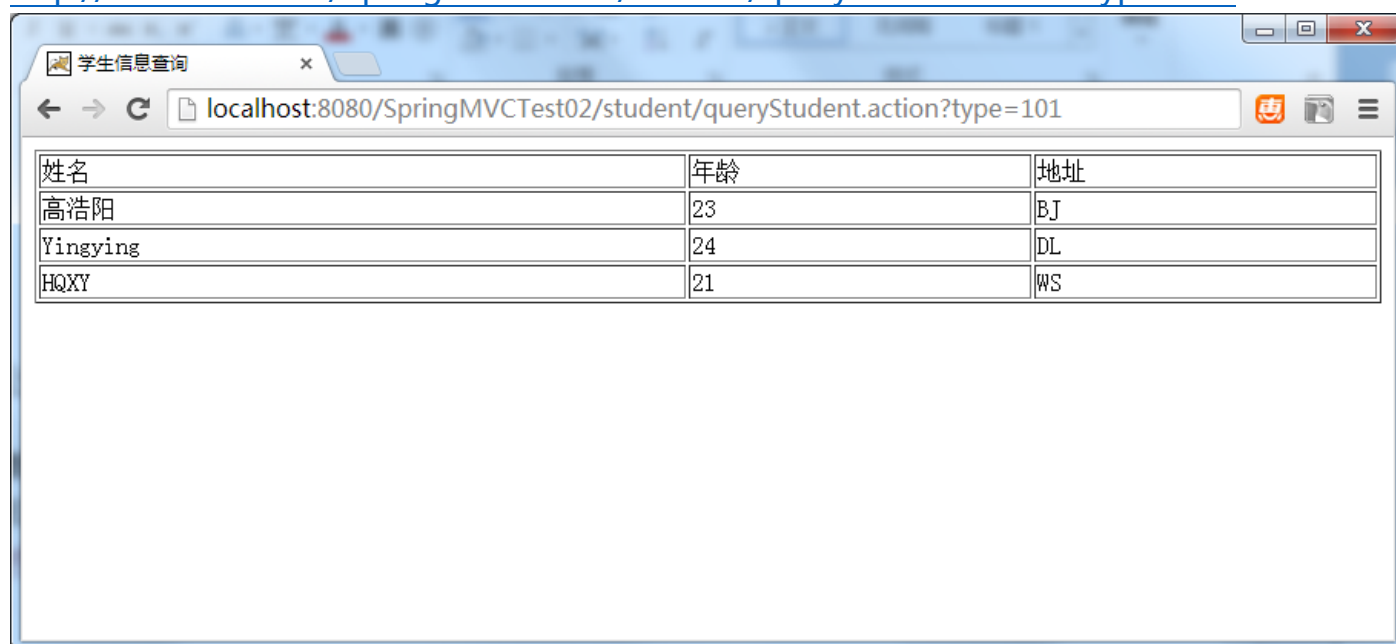
    //学生修改
    //学生添加
}

```

重新部署

在浏览器中

<http://localhost:8080/SpringMVCTest02/student/queryStudent.action?type=101>



The screenshot shows a web browser window with the title '学生信息查询' (Student Information Query). The address bar displays the URL 'localhost:8080/SpringMVCTest02/student/queryStudent.action?type=101'. The main content area contains a table with three columns: '姓名' (Name), '年龄' (Age), and '地址' (Address). The table lists three students: 高浩阳 (Gao Haoyang) with age 23 and address BJ, Yingying with age 24 and address DL, and HQXY with age 21 and address WS.

姓名	年龄	地址
高浩阳	23	BJ
Yingying	24	DL
HQXY	21	WS

Console

type = 101

## 5 SpringMVC 特点

SpringMVC 是基于方法开发的，通过方法形参接收请求的参数，可以使用单例或多例，建议使用单例。

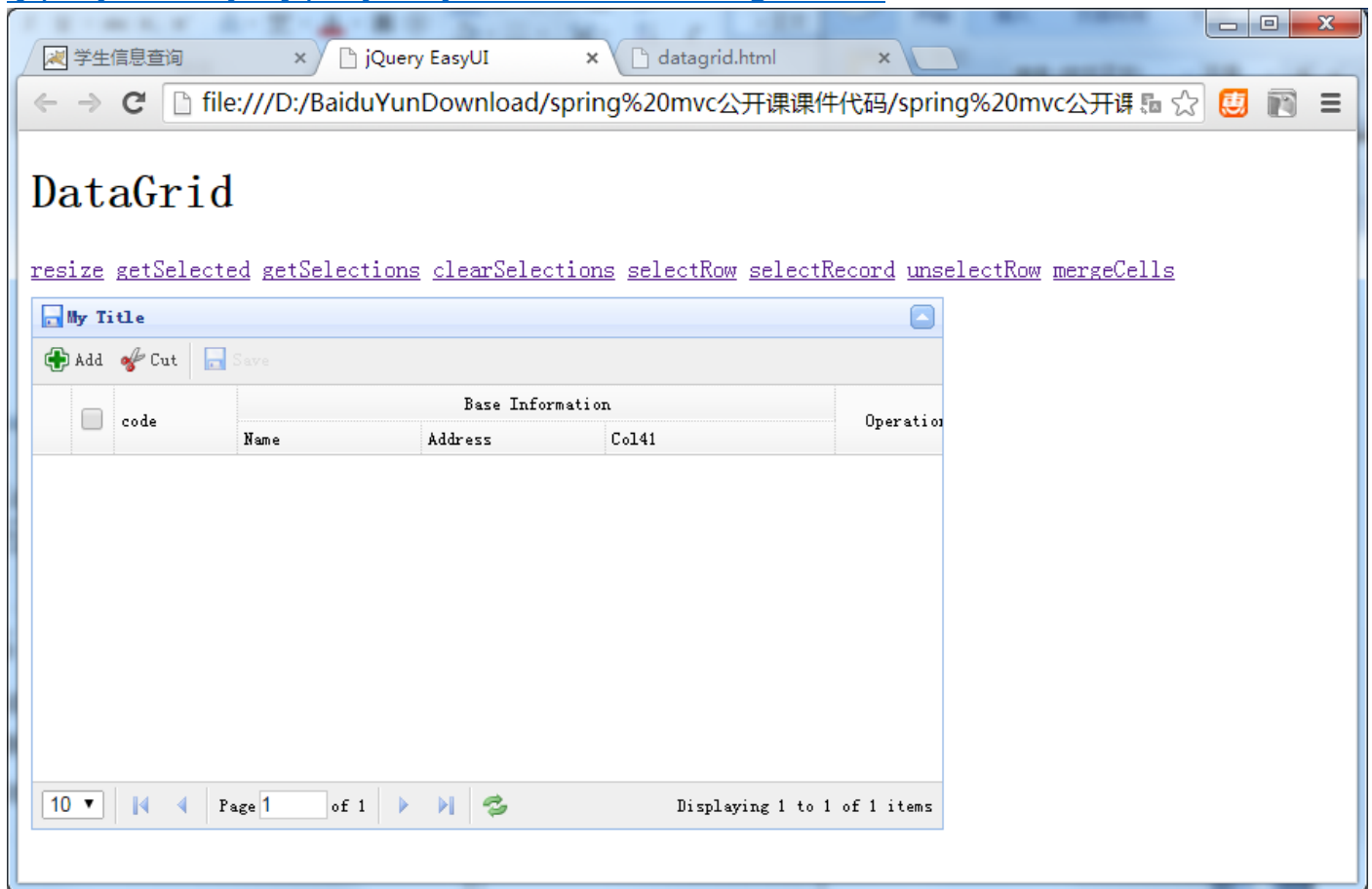
Struts 通过成员变量存储参数，只能设计为多例。

## 6 和 JQuery easyui 整合完成数据列表

数据列表的效果：

我也不知道为什么没有显示出 demo 中的数据

<file:///D:/BaiduYunDownload/spring%20mvc%E5%85%AC%E5%BC%80%E8%AF%BE%E8%AF%BE%E4%BB%B6%E4%BB%A3%E7%A0%81/spring%20mvc%E5%85%AC%E5%BC%80%E8%AF%BE%E8%AF%BE%E4%BB%B6%E4%BB%A3%E7%A0%81/%E8%B5%84%E6%96%99/jquery%20easyui/jquery-easyui-1.2.2/demo/datagrid.html>



使用 jQuery easyUI 的 datagrid 组件

datagrid.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```

<title>jQuery EasyUI</title>
<link rel="stylesheet" type="text/css" href="../../themes/default/easyui.css">
<link rel="stylesheet" type="text/css" href="../../themes/icon.css">
<script type="text/javascript" src="../../jquery-1.4.4.min.js"></script>
<script type="text/javascript" src="../../jquery.easyui.min.js"></script>
<script>
    $(function(){
        $('#test').datagrid({
            title:'My Title',//数据列表标题
            iconCls:'icon-save',//显示图标
            width:600,//数据列表的宽度
            height:350,//列表的高度
            nowrap: false,//单元格中的数据不换行，如果为true表示不换行，不换行情况下数据
            //加载性能高，如果为false就是换行，换行数据加载性能不高
            striped:true,//条纹显示效果
            collapsible:true,//是否可以折叠
            url:'datagrid_data.json',//加载数据的连接，引连接请求过来是json数据
            sortName: 'code',//排序字段名称
            sortOrder: 'desc',//排序的顺序
            remoteSort: false,//是否远程排序，如果要使用排序建议使用远程排序，对数据库中
            //所有数据进行排序，通过远程排序datagrid将sortName和sortOrder传到后台，这时后台的代码调用
            //service及dao进行数据库排序
            idField:'code',//此字段很重要，数据结果集的唯一约束
            frozenColumns:[[//冻结列
                {field:'ck',checkbox:true},
                {title:'code',field:'code',width:80,sortable:true}
            ]],
            columns:[[//普通列
                {title:'Base Information',colspan:3},
                {field:'opt',title:'Operation',width:100,align:'center', rowspan:2,
                    formatter:function(value,rec){
                        return '<span style="color:red">Edit Delete</span>';
                    }
                },
            ],[
                {field:'name',title:'Name',width:120},
                {field:'addr',title:'Address',width:120,rowspan:2,sortable:true,
                    sorter:function(a,b){
                        return (a>b?1:-1);
                    }
                },
            ],
                {field:'col4',title:'Col41',width:150,rowspan:2}
            ]],
            pagination:true,//是否显示分页
            rownumbers:true,//是否显示行号
            toolbar:[{//工具栏
                id:'btnadd',

```

```

        text: 'Add',
        iconCls: 'icon-add',
        handler: function(){
            $('#btnsave').linkbutton('enable');
            alert('add')
        }
    },{
        id: 'btncut',
        text: 'Cut',
        iconCls: 'icon-cut',
        handler: function(){
            $('#btnsave').linkbutton('enable');
            alert('cut')
        }
    }, '-', {
        id: 'btnsave',
        text: 'Save',
        disabled: true,
        iconCls: 'icon-save',
        handler: function(){
            $('#btnsave').linkbutton('disable');
            alert('save')
        }
    }
    ]});
var p = $('#test').datagrid('getPager');
if (p){
    $(p).pagination({
        onBeforeRefresh: function(){
            alert('before refresh');
        }
    });
}
});
function resize(){
    $('#test').datagrid('resize', {
        width: 700,
        height: 400
    });
}
function getSelected(){
    var selected = $('#test').datagrid('getSelected');
    if (selected){
        alert(selected.code+": "+selected.name+": "+selected.addr+": "+selected.col4);
    }
}

```

```

function getSelections(){
    var ids = [];
    var rows = $('#test').datagrid('getSelections');
    for(var i=0;i<rows.length;i++){
        ids.push(rows[i].code);
    }
    alert(ids.join(':'));
}
function clearSelections(){
    $('#test').datagrid('clearSelections');
}
function selectRow(){
    $('#test').datagrid('selectRow',2);
}
function selectRecord(){
    $('#test').datagrid('selectRecord','002');
}
function unselectRow(){
    $('#test').datagrid('unselectRow',2);
}
function mergeCells(){
    $('#test').datagrid('mergeCells',{
        index:2,
        field:'addr',
        rowspan:2,
        colspan:2
    });
}
</script>
</head>
<body>
<h1>DataGrid</h1>
<div style="margin-bottom:10px;">
    <a href="#" onclick="resize()">resize</a>
    <a href="#" onclick="getSelected()">getSelected</a>
    <a href="#" onclick="getSelections()">getSelections</a>
    <a href="#" onclick="clearSelections()">clearSelections</a>
    <a href="#" onclick="selectRow()">selectRow</a>
    <a href="#" onclick="selectRecord()">selectRecord</a>
    <a href="#" onclick="unselectRow()">unselectRow</a>
    <a href="#" onclick="mergeCells()">mergeCells</a>
</div>

<table id="test"></table>

</body>
</html>

```

datagrid 组件需要加载 url , url 需要返回 json 数据 :

datagrid\_data.json

```
{
  "total":239,
  "rows":[
    {"code":"001","name":"Name 1","addr":"Address 11","col4":"col4 data"},
    {"code":"002","name":"Name 2","addr":"Address 13","col4":"col4 data"},
    {"code":"003","name":"Name 3","addr":"Address 87","col4":"col4 data"},
    {"code":"004","name":"Name 4","addr":"Address 63","col4":"col4 data"},
    {"code":"005","name":"Name 5","addr":"Address 45","col4":"col4 data"},
    {"code":"006","name":"Name 6","addr":"Address 16","col4":"col4 data"},
    {"code":"007","name":"Name 7","addr":"Address 27","col4":"col4 data"},
    {"code":"008","name":"Name 8","addr":"Address 81","col4":"col4 data"},
    {"code":"009","name":"Name 9","addr":"Address 69","col4":"col4 data"},
    {"code":"010","name":"Name 10","addr":"Address 78","col4":"col4 data"}
  ]
}
```

上面的格式是 datagrid 要求的。

让 SpringMVC 编写一个 action 方法 , 返回 json 数据。

使用 SpringMVC 提供的 @ResponseBody 注解 , 将 java 对象转成 json 格式数据。

思路 :

使用 @ResponseBody 注解将一个 map 转成 json 数据。

Map 中填充 :

total : 数据总数

rows : List<Student>

这时转成 datagrid 要求的 json 格式。

StudentAction.java

```
package com.haoyang.springmvc.action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

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

```
import org.springframework.web.bind.annotation.ResponseBody;

import com.haoyang.springmvc.javabean.Student;

/**
 * 学生信息管理
 *
 * @author Administrator
 *
 */
@Controller
@RequestMapping("/student")
public class StudentAction {

    // 学生查询
    @RequestMapping("queryStudent")
    public String queryStudent(Model model, String type) {

        // 学生类型
        System.out.println("type = " + type);

        // 将学生信息显示到页面上
        List<Student> list = new ArrayList<Student>();

        // 模拟静态数据

        Student student1 = new Student();
        student1.setUsername("高浩阳");
        student1.setAge(23);
        student1.setAddress("BJ");

        Student student2 = new Student();
        student2.setUsername("Yingying");
        student2.setAge(24);
        student2.setAddress("DL");

        Student student3 = new Student();
        student3.setUsername("HQXY");
        student3.setAge(21);
        student3.setAddress("WS");

        list.add(student1);
        list.add(student2);
        list.add(student3);

        model.addAttribute("studentList", list);
    }
}
```

```

    // 返回逻辑视图名
    return "queryStudent";
}

// 学生查询, 返回json
@RequestMapping("queryStudentJson")
public @ResponseBody Map<String, Object> queryStudentJson(Model model) {

    // 将学生信息显示到页面上
    List<Student> list = new ArrayList<Student>();

    // 模拟静态数据

    Student student1 = new Student();
    student1.setUsername("高浩阳");
    student1.setAge(23);
    student1.setAddress("BJ");

    Student student2 = new Student();
    student2.setUsername("Yingying");
    student2.setAge(24);
    student2.setAddress("DL");

    Student student3 = new Student();
    student3.setUsername("HQXY");
    student3.setAge(21);
    student3.setAddress("WS");

    list.add(student1);
    list.add(student2);
    list.add(student3);

    model.addAttribute("studentList", list);

    //定义map
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("total", list.size());
    map.put("rows", list);

    // 返回map数据, 由于使用了@ResponseBody注解, 自动将map转成json
    return map;
}

// 学生修改
// 学生添加
}

```



@ResponseBody 注解用于将 Controller 的方法返回的 java 对象转为 json 格式数据输出到客户端页面

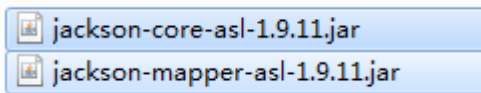
配置 json 转换器：

实现将 java 对象转成 json

依赖 jar

Springmvc 默认用 MappingJacksonHttpMessageConverter 对 json 数据进行转换，需要加入 jackson 的包

D:\BaiduYunDownload\spring mvc 公开课课件代码\spring mvc 公开课课件代码\jar 包  
\springmvc 加入 json 支持



在注解适配器中加入 messageConverters

springmvc.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.1.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">

    <!-- 使用组件扫描 -->
    <!-- 将action扫描出来，在spring容器中进行注册，自动对action在spring容器中进行配置 -->
    <context:component-scan base-package="com.haoyang.springmvc.action" />

    <!--注解映射器HandlerMapping -->
    <bean

        class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerM
```

```

apping" />

<!--注解适配器HandlerAdapter -->
<bean

    class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerA
dapter">
    <!-- json转换器 -->
    <property name="messageConverters">
        <list>
            <bean
                class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter"
            />
        </list>
    </property>
</bean>

<!-- 视图解析器ViewResolver -->
<!-- 解析jsp, 默认支持jstl -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>

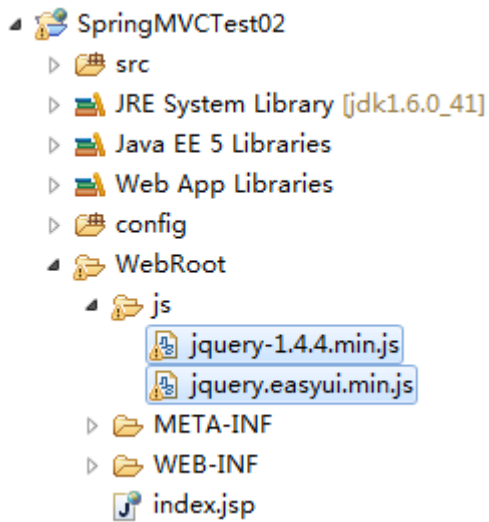
</beans>

```

开始写页面

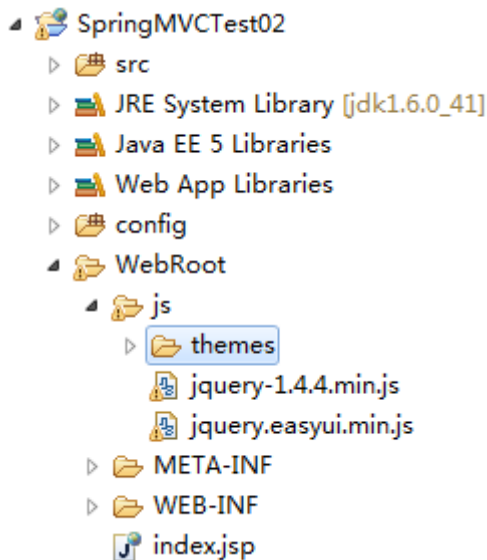
将js 文件拷贝到工程中

D:\BaiduYunDownload\spring mvc 公开课课件代码\spring mvc 公开课课件代码\资料\jquery  
easyui\jquery-easyui-1.2.2



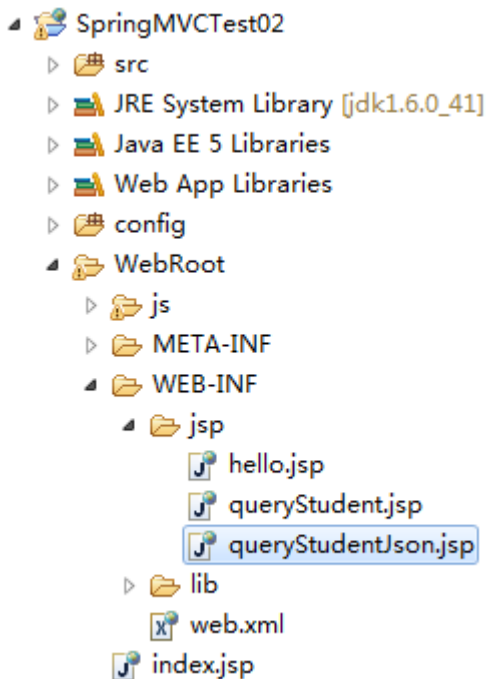
主题也拷贝进去

D:\BaiduYunDownload\spring mvc 公开课课件代码\spring mvc 公开课课件代码\资料\jquery easyui\jquery-easyui-1.2.2



创建一个jsp 页面

queryStudentJson.jsp



### queryStudentJson.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>学生列表</title>
<link rel="stylesheet" type="text/css"
href="${pageContext.request.contextPath}/js/themes/default/easyui.css">
<link rel="stylesheet" type="text/css"
href="${pageContext.request.contextPath}/js/themes/icon.css">
<script type="text/javascript"
src="${pageContext.request.contextPath}/js/jquery-1.4.4.min.js"></script>
<script type="text/javascript"
src="${pageContext.request.contextPath}/js/jquery.easyui.min.js"></script>

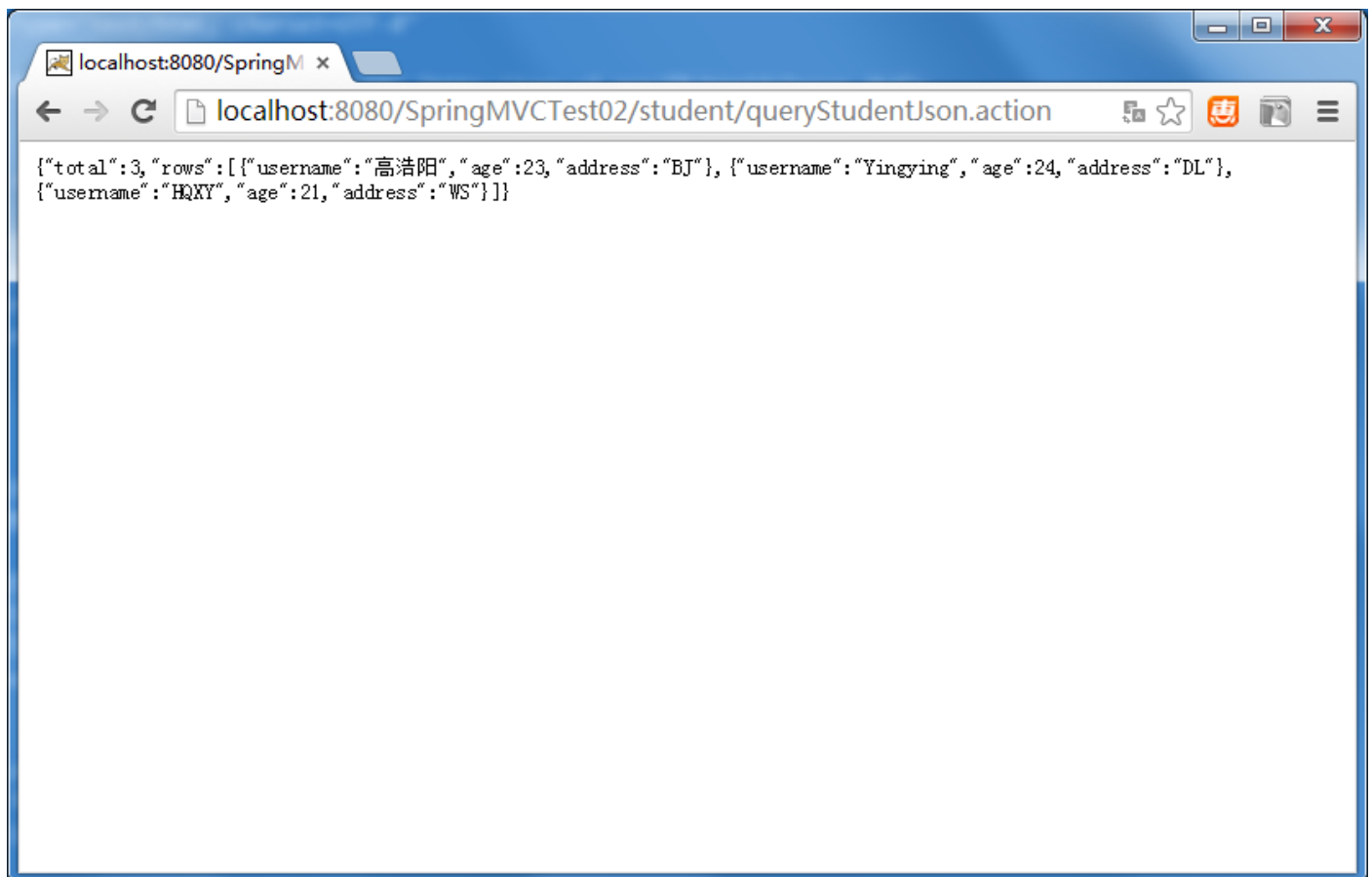
<script type="text/javascript">
$(function(){
    $('#test').datagrid({
        title:'学生列表',//数据列表标题
        width:600,//数据列表的宽度
        height:350,//列表的高度
        nowrap: false,//单元格中的数据不换行，如果为true表示不换行，不换行情况下数据加载性能
高，如果为false就是换行，换行数据加载性能不高
        striped:true,//条纹显示效果
        collapsible:true,//是否可以折叠
        url:'${pageContext.request.contextPath}/student/queryStudentJson.action',//加
```

载数据的连接，引连接请求过来是json数据

```
        columns:[[
            {field:'username',title:'姓名',width:120},
            {field:'age',title:'年龄',width:120},
            {field:'address',title:'地址',width:120}
        ]],
        pagination:true,//是否显示分页
        rownumbers:true,//是否显示行号
        toolbar:[{//工具栏
            id:'btnadd',
            text:'添加',
            iconCls:'icon-add',
            handler:function(){
                $('#btnsave').linkbutton('enable');
                alert('add');
            }
        }]
    });
});
</script>
</head>
<body>
<table id="test"></table>
</body>
</html>
```

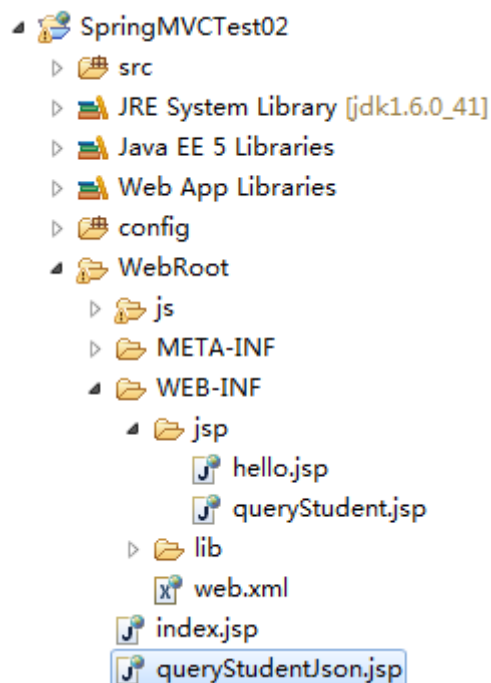
部署启动

访问 <http://localhost:8080/SpringMVCTest02/student/queryStudentJson.action>



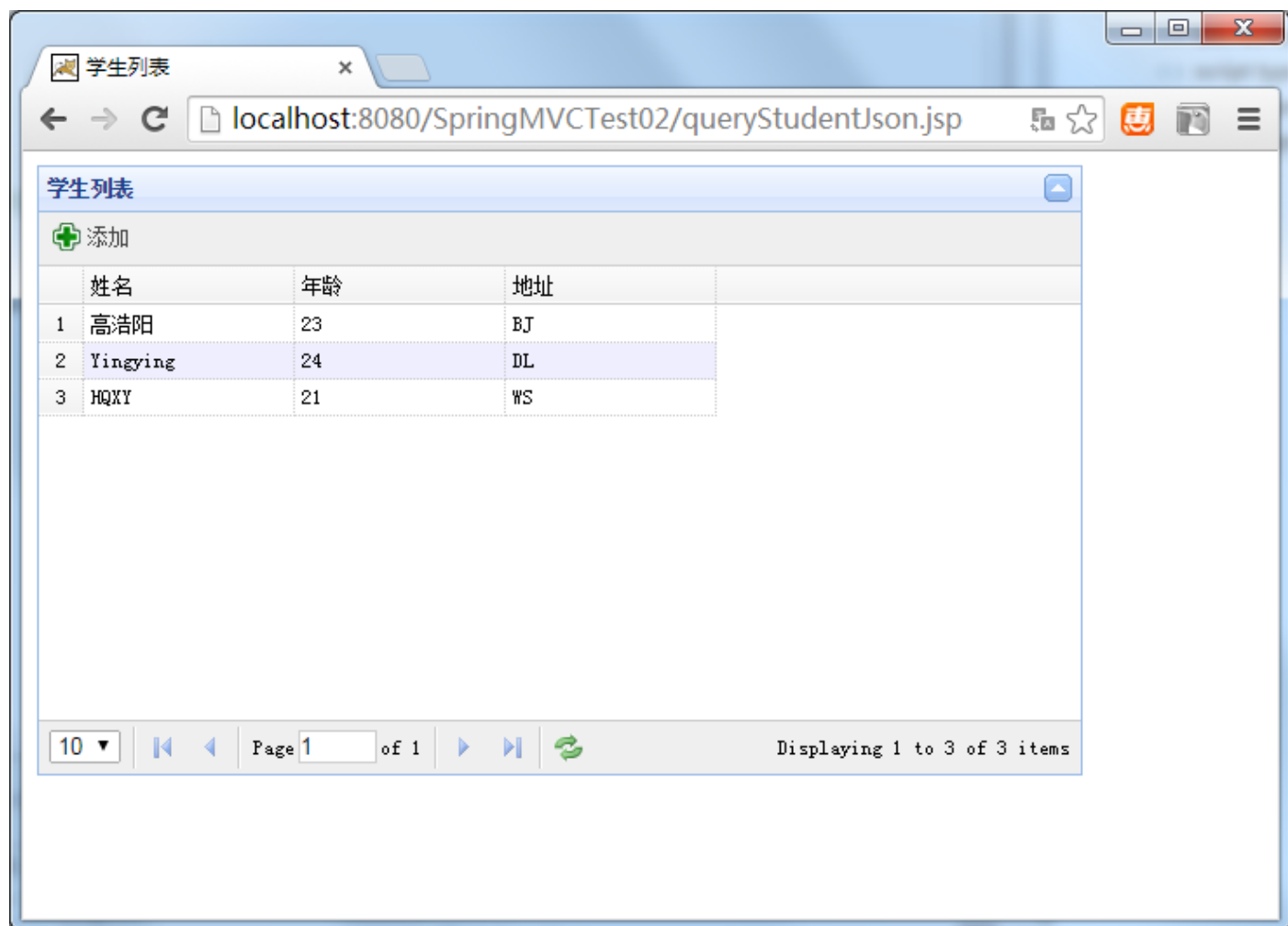
以上为测试 json 数据

将 queryStudentJson.jsp 移动出来才能看到效果



直接访问 <http://localhost:8080/SpringMVCTest02/queryStudentJson.jsp>

测试 datagrid 效果



【END】

2014/11/28