# 一、基础

## 1. SpringMVC概念

MVC 模式导致了应用程序的不同方面(输入逻辑、业务逻辑和 UI 逻辑)的分离，同时提供了在这些元素之间的松散耦合。

1. 模型封装了应用程序数据，并且通常它们由 POJO 组成。

POJO：简单的 Java 对象，普通的JavaBeans，不允许有业务方法

1. 视图主要用于呈现模型数据，并且通常它生成客户端的浏览器可以解释的 HTML 输出。
2. 控制器主要用于处理用户请求，并且构建合适的模型并将其传递到视图呈现。

## 2. DispatcherServlet

* Spring Web 模型-视图-控制（MVC）框架是围绕 DispatcherServlet 设计的，DispatcherServlet 用来处理所有的 HTTP 请求和响应。

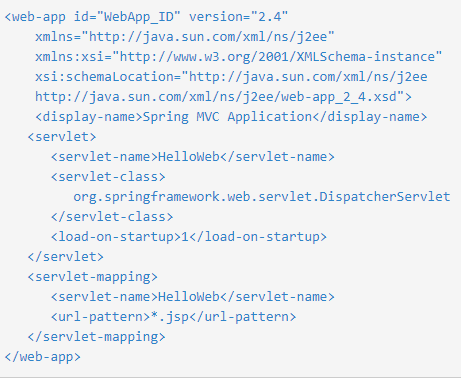


* 1. DispatcherServlet 传入 HTTP 请求的事件序列：

1. 收到一个 HTTP 请求后，DispatcherServlet 根据 HandlerMapping 来选择并且调用适当的控制器。
2. 控制器接受请求，并基于使用的 GET 或 POST 方法来调用适当的 service 方法。Service 方法将设置基于定义的业务逻辑的模型数据，并返回视图名称到 DispatcherServlet 中。
3. DispatcherServlet 会从 ViewResolver 获取帮助，为请求检取定义视图。
4. 一旦确定视图，DispatcherServlet 将把模型数据传递给视图，最后呈现在浏览器中
5. HandlerMapping、Controller 和 ViewResolver 是 WebApplicationContext 的一部分，而 WebApplicationContext 是带有一些对 web 应用程序必要的额外特性的 ApplicationContext 的扩展。

2.2 需求的配置 -- 配置servlet

(1) 需要映射 DispatcherServlet 处理的请求，通过使用在 web.xml 文件中的一个 URL 映射。（映射HelloWeb的DispatcherServlet）



(2) 工作原理： -- 默认的文件名

web.xml 文件将被保留在你的应用程序的 WebContent/WEB-INF 目录下。好的，在初始化 HelloWeb DispatcherServlet 时，该框架将尝试加载位于该应用程序的 WebContent/WEB-INF 目录中文件名为 [servlet-name]-servlet.xml 的应用程序内容。在这种情况下，我们的文件将是 HelloWeb-servlet.xml。

接下来，<servlet-mapping> 标签表明哪些 URLs 将被 DispatcherServlet 处理。这里所有以 .jsp 结束的 HTTP 请求将由 HelloWeb DispatcherServle t处理。

(3) 如果你不想使用默认文件名 [servlet-name]-servlet.xml 和默认位置 WebContent/WEB-INF，你可以通过在 web.xml 文件中添加 servlet 监听器 ContextLoaderListener 自定义该文件的名称和位置，



2.3 HelloWeb-servlet.xml – 相关配置(WEB-INF目录下)



* 配置要点：

1. [servlet-name]-servlet.xml 文件将用于创建 bean 定义，重新定义在全局范围内具有相同名称的任何已定义的 bean。
2. <context:component-scan...> 标签将用于激活 Spring MVC 注释扫描功能，该功能允许使用注释，如 @Controller 和 @RequestMapping 等等。
3. InternalResourceViewResolver 将使用定义的规则来解决视图名称。按照上述定义的规则，一个名称为 hello 的逻辑视图将发送给位于 /WEB-INF/jsp/hello.jsp 中实现的视图。
   1. 定义控制器 - DispatcherServlet 发送请求到控制器中执行特定的功能。
4. 注释：

@Controller 注释表明一个特定类是一个控制器的作用。

@RequestMapping 注释用于映射 URL 到整个类或一个特定的处理方法。



@Controller 注释定义该类作为一个 Spring MVC 控制器。在这里，第一次使用的 @RequestMapping 表明在该控制器中处理的所有方法都是相对于 /hello 路径的。下一个注释 @RequestMapping(method = RequestMethod.GET) 用于声明 printHello() 方法作为控制器的默认 service 方法来处理 HTTP GET 请求。你可以在相同的 URL 中定义其他方法来处理任何 POST 请求。

1. 属性



1. 值(value)属性表明 URL 映射到哪个处理方法，方法(method)属性定义了 service 方法来处理 HTTP GET 请求。
2. 你将在一个 service 方法中定义需要的业务逻辑。你可以根据每次需求在这个方法中调用其他方法。
3. 基于定义的业务逻辑，你将在这个方法中创建一个模型。你可以设置不同的模型属性，这些属性将被视图访问并显示最终的结果。这个示例创建了一个带有属性 “message” 的模型。
4. **一个定义的 service 方法可以返回一个包含视图名称的字符串用于呈现该模型**。这个示例返回 “hello” 作为逻辑视图的名称。
   1. 创建 JSP 视图 –



${message} 是我们在控制器内部设置的属性。

* 访问流程

启动tomcat后，第一个访问web.xml

spring会先找到DispatcherServlet，找到servlet-name：HelloWeb

继续找到HelloWeb-Servlet.xml

根据url 找到控制器名称/hello，控制器return视图名称hello.jsp

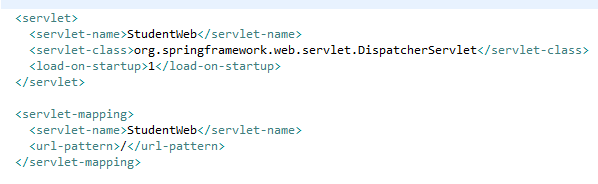
访问的时候为：localhost:8080/项目名/控制器虚拟文件夹

localhost:8080/Tspmvc/hello

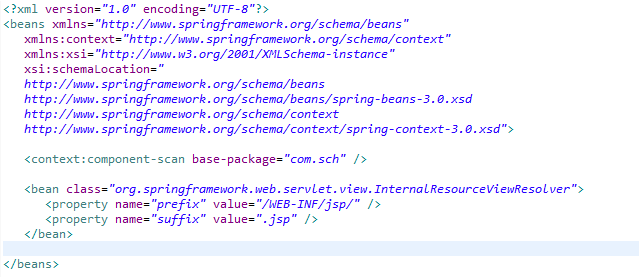
# 二、SpringMVC实例

## 表单提交实例 – 利用<form:form>标签

1. web.xml配置



1. StudentWeb-servlet.xml配置文件



1. StudentController控制器 – 提供访问地址以及转发跳转jsp地址



1. SpringMVC jsp页面 – 带有<form:form>标签 – 注意添加标签！！！！

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>



1. 执行流程

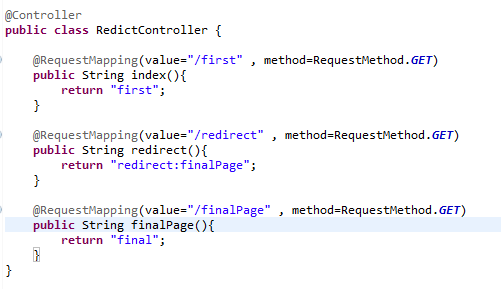
web.xml文件提供读取servlet的 地址；

StudentWeb-servlet.xml文件提供注解的servlet文件地址以及jsp的文件地址；

根据浏览器地址访问Controller，StudentController文件提供url访问地址以及转发页面；

jsp页面负责视图，以及访问不同控制器。

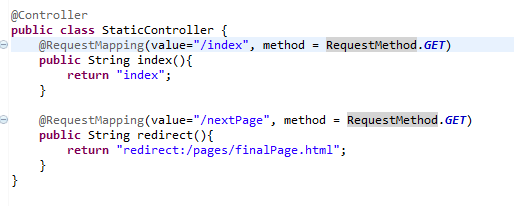
## 重定向例子 – Redirect



* redirect：finalPage 跳转页面

## 跳转到静态页面

1. Controller具体代码：



1. servlet.xml具体代码：<mvc>标签

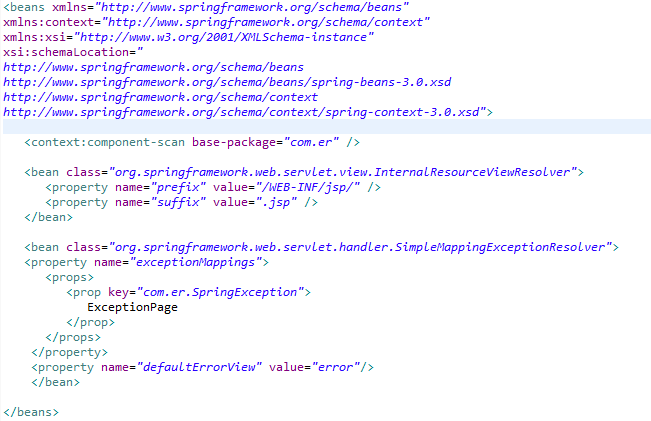


* 配置导入mvc以及其他location，添加<mvc>标签

在这里，<mvc:resources..../> 标签被用来映射静态页面。 mapping 属性必须是一个指定一个 http 请求的 URL 模式的 Ant 模式。 location 属性必须指定一个或者多个具有包含图片，样式表，JavaScript 和其他静态内容的静态页面的资源目录位置。多个资源位置可以使用逗号分隔这些值的列表来被指定。

## 异常处理例子

1. servlet.xml配置



1. Controller代码

