第 3 章
Spring MVC简介

3.1 Spring MVC 概述

Spring MVC 是一种基于 MVC 设计模式的使用请求-响应模型的轻量级 Web 框架。所谓 MVC 是一种 Web 开发领域的设计模式，MVC 是 Model、View 与 Controller 三个英文单词的缩写，指的是对 Web 应用程序中的资源按功能划分为以下三大部分。

View：视图，是用户进行操作的可视化的界面，可以是 HTML、JSP、XML 等。用户可以在视图上看到服务端传来的数据，或者在视图上录入数据以传递给服务端处理。

Model：模型，用于处理业务逻辑、封装、传输业务数据。

Controller：控制器，是程序的调度中心，控制程序的流转，接收客户端的请求，判断该调用哪个服务端程序来处理，处理完毕后把获得的模型数据显示到视图，返回给用户。

通过以上分工，将使程序更加简单、高效。著名的 Struts2 和 Spring MVC 都是 MVC 框架。

3.1.1 Spring MVC 的优点

Spring MVC 跟 Struts2 相比，具有更好的安全性和可靠性，运行速度更快。目前，Spring MVC 已成为 Java Web 开发的一款利器，越来越受到 Java 开发者的喜欢。其主要优点如下。

(1) 角色划分清晰。核心控制器(DispatcherServlet)、处理器映射器(HandlerMapping)、处理器适配器(HandlerAdapter)、视图解析器(ViewResolver)、处理器控制器(Controller)、验证器(Validator)、命令对象(即 Command 请求参数绑定到的对象)、表单对象(即 Form Object 提供给表单展示和提交到的对象)。

(2) 分工明确，扩展灵活。作为 Spring 的一部分，易与 Spring 其他框架集成。

(3) 可适配性好。通过 HandlerAdapter 就可以支持任意一个类作为处理器。

(4) 支持数据验证、数据格式化、数据绑定机制。

(5) 提供功能强大的 JSP 标签库，使数据在视图中的展示或者获取更加丰富与灵活。

(6) RESTful 风格的支持，简单的文件上传、下载功能。

(7) 注解的零配置支持等。

3.1.2　Spring MVC 的运行原理

Spring MVC 的工作流程如图 3-1 所示。

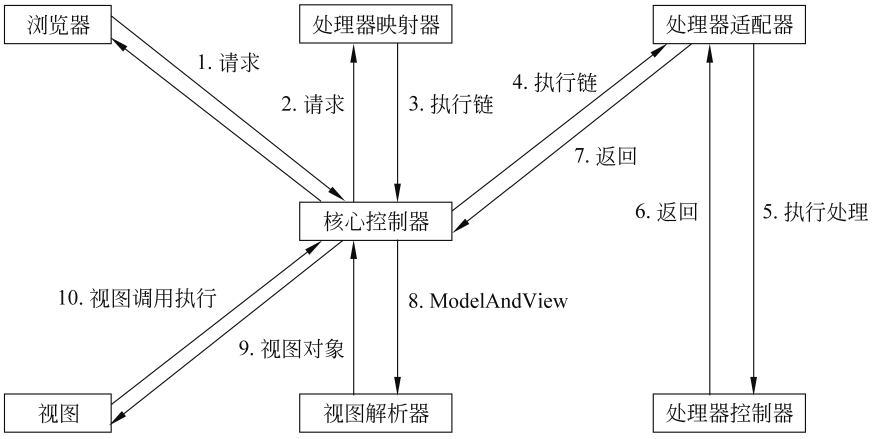

图 3-1　Spring MVC 的工作流程

(1) 浏览器向服务端提交请求,请求会被核心控制器 DispatcherServlet 拦截。

(2) 核心控制器将请求转给处理器映射器 HandlerMapping。

(3) 处理器映射器 HandlerMapping 会根据请求找到处理该请求的具体的处理器,并将其封装为处理器执行链后返回给核心控制器 DispatcherServlet。

(4) 核心控制器根据处理器执行链中的处理器,找到能够执行该处理器的处理器适配器 HandlerAdapter。

(5) 处理器适配器 HandlerAdapter 调用执行处理器控制器 Controller。

(6) 处理器控制器 Controller 将处理结果及要跳转的视图封装到一个对象 ModelAndView 中,并将其返回给处理器适配器 HandlerAdapter。

(7) 处理器适配器 HandlerAdapter 直接将结果返回给核心控制器。

(8) 核心控制器调用视图解析器 ViewResolver,将 ModelAndView 中的视图名称封装为视图对象 View。

(9) 视图解析器 ViewResolver 将封装了的视图对象 View 返回给核心控制器 DispatcherServlet,到此一个流程结束。

(10) 核心控制器 DispatcherServlet 调用视图对象 View,让其自己进行数据填充,形成响应对象。

(11) 核心控制器把填充好数据的 View 响应给浏览器。

3.2　创建一个 Spring MVC 程序

项目案例：用户通过浏览器提交一个请求,服务端处理器在接收到这个请求后,给出一条欢迎信息：Hello Spring MVC,在响应页面中显示该信息,采用传统的配置式开发

方式。

(1) 在 Eclipse 中新建一个 Dynamic Web Project 项目。

(2) 导入 Spring 的 jar 包。项目结构如图 3-2 所示。

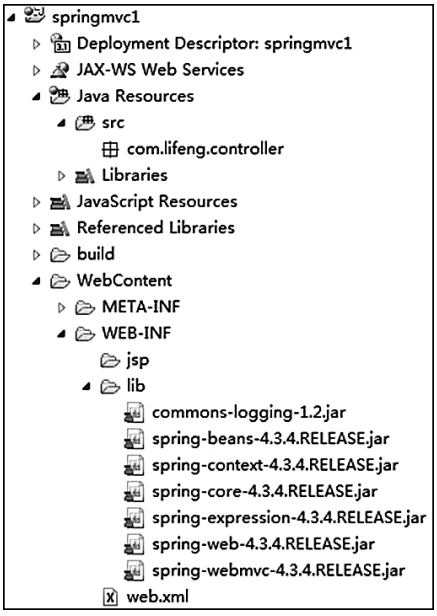

图 3-2 项目结构

(3) 在 web.xml 上配置 DispatcherServlet 核心控制器,在项目 WebContent/WEB-INF 目录下的 web.xml 文件关键配置如下:

```
<servlet>
    <servlet-name>Spring MVC</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:Spring MVC.xml</param-value>
    </init-param>
</servlet>
<servlet-mapping>
<servlet-name>Spring MVC</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

(4) 编写一个 Controller 类。

```
public class FirstController implements Controller{
    @Override
    public ModelAndView handleRequest(HttpServletRequest arg0, HttpServletResponse
```

```
arg1) throws Exception {
        ModelAndView mv=new ModelAndView();
        mv.addObject("hello", "Hello Spring MVC!!!");
        mv.setViewName("welcome");
        return mv;
    }
}
```

这个方法的作用是创建一个 ModelAndView 对象，在该对象中添加 hello 字符串对象，值为“Hello Spring MVC!!!”，并在 ModelAndView 对象中设置返回的逻辑视图为 welcome，这里逻辑视图 welcome 本身还不是完整的视图路径，将在下述有关步骤（视图解析器）进一步解释为完整路径。此方法意味着程序将携带 hello 对象数据跳转到名为 welcome 的逻辑视图。

（5）配置 HandlerMapping 处理器映射器。在 src 下新建一个 xml 文件，命名为 Spring MVC.xml，输入如下关键内容。

```
<!-- 配置处理器映射器-->
<bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping">
</bean>
```

上面代码的意思是创建一种类型为 BeanNameUrlHandlerMapping 的处理器映射器，即定义一种“请求/响应”映射规则，客户端的 URL 请求如果跟某一个 bean 的 name 属性匹配，则由该 bean 的 class 属性指定的控制器 Controller 类进行响应处理。

（6）配置处理器适配器。配置完处理器映射器后，接着在 Spring MVC.xml 中插入如下内容（插入位置在如上代码的＜/bean＞下方，节点＜/beans＞的上方）。

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

该代码的意思是创建一种处理器适配器，类型为 SimpleControllerHandlerAdapter，用于对上述指定的控制器 Controller 类的 handleRequest()方法的调用与执行。

（7）配置自定义控制器。继续在 Spring MVC.xml 中输入如下内容。

```
<bean name="/hello.do" class="com.fqy.controller.FirstController">
</bean>
```

该代码自定义了一个 bean，定义了一种具体的“请求/响应”映射关系，表示假如客户端发出的 url 请求的是/hello.do，则指定由服务端的 com.fqy.controller.FirstController 程序来处理，通过这行代码确定了一条具体的“请求/响应”映射关系，一种一对一的对应关系，即 name 属性表示的客户端请求（如 URL 路径），对应 class 属性表示的服务端的响应程序。

注意：这个 bean 的配置要有效，前提是第(5)步要配置类型为 BeanNameUrlHandlerMapping 的映射器，以及第(6)步的适配器。第(5)步相当于制定了一个规则，如乒乓球比赛的“男

女混合双打”，而第(7)步则是这个规则下的具体条文，如指派张无忌与赵敏搭配去参加“男女混合双打”。

(8) 配置视图解析器。视图解析器用来解释控制器返回的逻辑视图的真实路径，这样更方便，易于扩展。在 Spring MVC.xml 中输入如下代码。

```
<!-- 配置视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!--逻辑视图前缀-->
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <!--逻辑视图后缀,匹配模式:前缀+逻辑视图+后缀,形成完整路径名-->
        <property name="suffix" value=".jsp"></property>
    </bean>
```

上面代码的意思是控制器 Controller 返回的逻辑视图，需要加上前缀“/WEB-INF/jsp/”>和后缀“.jsp”，最后拼接成完整的视图路径。如本例中，Controller 返回的视图为 welcome，视图解析器将为它加上前缀、后缀，最终构成完整路径为“/WEB-INF/jsp/welcome.jsp”。视图解析器不是非要不可，如果没有视图解析器，则 Controller 返回的视图必须打上完整路径的视图名称，Controller 类应修改为如下所示：

```
public class FirstController implements Controller{
    @Override
    public ModelAndView handleRequest(HttpServletRequest arg0, HttpServletResponse
arg1) throws Exception {
        ModelAndView mv=new ModelAndView();
        mv.addObject("hello", "Hello Spring MVC!!!");
        mv.setViewName("/WEB-INF/jsp/welcome.jsp");
        return mv;
    }
}
```

到此，完整的 Spring MVC.xml 内容结构如下所示。

```
<!-- 1.配置处理器映射器,确定一种请求-响应的映射规则-->
    <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping">
</bean>
    <!-- 2.配置处理器适配器,配置对处理器的 handleRequest()方法的调用 -->
    <bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"
></bean>
    <!-- 3.配置自定义控制器,一条具体的映射关系,name 对应客户端 url 请求,class 对应
服务器端响应程序 -->
    <bean name="/hello.do" class="com.fqy.controller.FirstController">
</bean>
    <!-- 4.配置视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!--逻辑视图前缀-->
```

```
    <property name="prefix" value="/WEB-INF/jsp/"></property>
    <!--逻辑视图后缀,匹配模式:前缀+逻辑视图+后缀,形成完整路径名-->
    <property name="suffix" value=".jsp"></property>
</bean>
```

提示：上述类型的处理器映射器是默认的，同样，上述类型的处理器适配器也是默认的，两者均可省略，因此，Spring MVC.xml 又可以简化为

```
<!-- 1.配置自定义控制器,一条具体的映射关系,name 对应客户端 url 请求,class 对应服务器端响应程序 -->
    <bean name="/hello.do" class="com.fqy.controller.FirstController"></bean>
    <!-- 2.配置视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!--逻辑视图前缀-->
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <!--逻辑视图后缀,匹配模式:前缀+逻辑视图+后缀,形成完整路径名-->
        <property name="suffix" value=".jsp"></property>
    </bean>
```

(9) 定义一个响应页面。在目录 WebContent/WEB-INF/jsp 下新建一个 jsp 文件 welcome.jsp，打开 welcome.jsp 文件，在<body></body>之间输入如下代码：

```
<h1>${hello }</h1>
```

(10) 运行项目，在浏览器中输入 http://localhost:8080/springmvc1/hello.do，出现如图 3-3 所示结果，最终完整的项目结构如图 3-4 所示。

图 3-3　Spring MVC 项目运行结果

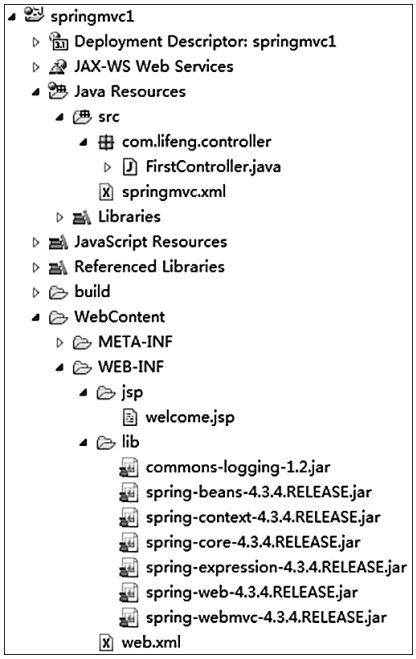

图 3-4　完整的项目结构