Spring MVC 概述

9413	0

Spring MVC 框架是一个开源的 Java 平台，为开发强大的基于 Java 的 Web 应用程序提供全面的基础架构支持非常容易和非常快速。

Spring 框架最初由 Rod Johnson 撰写，并于 2003 年 6 月根据 Apache 2.0 许可证首次发布。

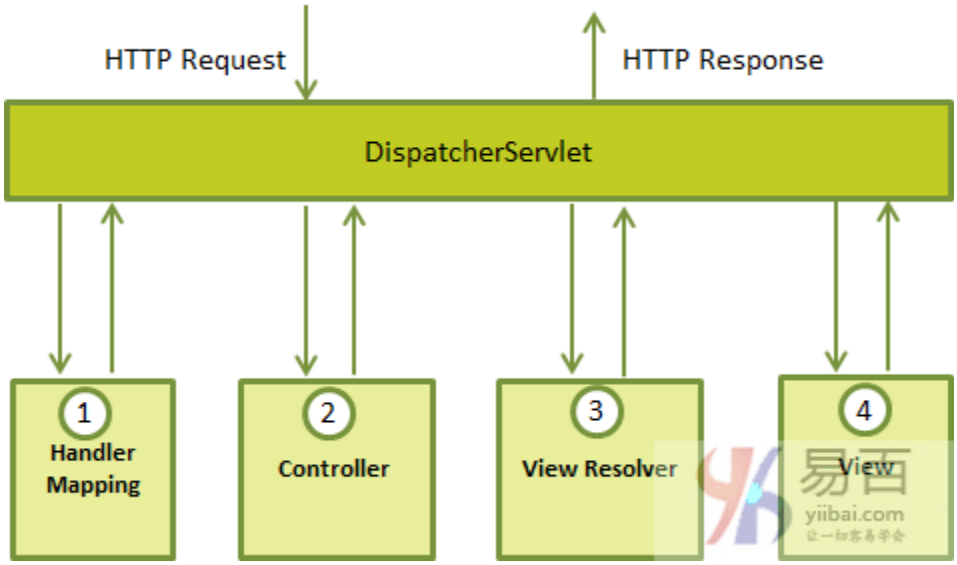
本教程是基于 2015 年 3 月发布的 Spring Framework 版本 4.1.6 编写的。

Spring web MVC 框架提供了 MVC(模型 - 视图 - 控制器)架构和用于开发灵活和松散耦合的 Web 应用程序的组件。 MVC 模式导致应用程序的不同方面(输入逻辑，业务逻辑和 UI 逻辑)分离，同时提供这些元素之间的松散耦合。

- **模型(Model)**封装了应用程序数据，通常它们将由 POJO 类组成。
- **视图(View)**负责渲染模型数据，一般来说它生成客户端浏览器可以解释 HTML 输出。
- **控制器(Controller)**负责处理用户请求并构建适当的模型，并将其传递给视图进行渲染。

DispatcherServlet 组件类

Spring Web 模型 - 视图 - 控制器(MVC)框架是围绕 DispatcherServlet 设计的，它处理所有的 HTTP 请求和响应。 Spring Web MVC DispatcherServlet 的请求处理工作流如下图所示：



以下是对应于到 DispatcherServlet 的传入 HTTP 请求的事件顺序：

- 在接收到 HTTP 请求后，DispatcherServlet 会查询 HandlerMapping 以调用相应的 Controller。
- Controller 接受请求并根据使用的 GET 或 POST 方法调用相应的服务方法。 服务方法将基于定义的业务逻辑设置模型数据，并将视图名称返回给 DispatcherServlet。
- DispatcherServlet 将从 ViewResolver 获取请求的定义视图。
- 当视图完成，DispatcherServlet 将模型数据传递到最终的视图，并在浏览器上呈现。

所有上述组件，即: HandlerMapping，Controller 和 ViewResolver 是 WebApplicationContext 的一部分，它是普通 ApplicationContext 的扩展，带有 Web 应用程序所需的一些额外功能。

必需的配置

需要通过使用 web.xml 文件中的 URL 映射来映射希望 DispatcherServlet 处理的请求。 下面是一个示例来显示 HelloWeb DispatcherServlet 示例的声明和映射：

```
<web-app id="WebApp_ID" version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <display-name>Spring MVC Application</display-name>

   <servlet>
      <servlet-name>HelloWeb</servlet-name>
      <servlet-class>
         org.springframework.web.servlet.DispatcherServlet
      </servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>HelloWeb</servlet-name>
      <url-pattern>*.jsp</url-pattern>
   </servlet-mapping>

</web-app>
XML
```

web.xml 文件将保存 Web 应用程序的 WebContent/WEB-INF 目录。在 HelloWeb DispatcherServlet 初始化时，框架将尝试从位于应用程序的 WebContent/WEB-INF 目录中

的名为[servlet-name]-servlet.xml 的文件加载应用程序上下文。在这个示例中，使用的文件将是 HelloWeb-servlet.xml。

接下来，<servlet-mapping>标记指示哪些 URL 将由 DispatcherServlet 处理。 这里所有以.jsp 结尾的 HTTP 请求都将由 HelloWeb DispatcherServlet 处理。

如果不想使用默认文件名为[servlet-name]-servlet.xml 和默认位置为 WebContent/WEB-INF，可以通过在 web.xml 文件中添加 servlet 侦听器 ContextLoaderListener 来自定义此文件名和位置 如下：

```
<web-app...>

<!-------- DispatcherServlet definition goes here----->
....
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>/WEB-INF/HelloWeb-servlet.xml</param-value>
</context-param>

<listener>
   <listener-class>
      org.springframework.web.context.ContextLoaderListener
   </listener-class>
</listener>
</web-app>
Java
```

现在来看看 HelloWeb-servlet.xml 文件的必需配置，放在 Web 应用程序的 WebContent/WEB-INF 目录中：

```
<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">

   <context:component-scan base-package="com.yiibai" />

   <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="prefix" value="/WEB-INF/jsp/" />
      <property name="suffix" value=".jsp" />
```

```
   </bean>

</beans>
```

Java

以下是有关 HelloWeb-servlet.xml 文件的重点说明：

- [servlet-name]-servlet.xml 文件将用于创建定义的 bean，它会覆盖在全局范围中使用相同名称定义的任何 bean 的定义。
- <context: component-scan ...>标签将用于激活 Spring MVC 注释扫描功能，允许使用@Controller 和@RequestMapping 等注释。
- InternalResourceViewResolver 将定义用于解析视图名称的规则。根据上面定义的规则，hello 的逻辑视图将委托给位于/WEB-INF/jsp/hello.jsp 这个视图来实现。

下一节将演示如何创建实际组件。即：控制器，模型和视图。

定义控制器

DispatcherServlet 将请求委派给控制器以执行特定于其的功能。 @Controller 注释指示特定类充当控制器的角色。@RequestMapping 注释用于将 URL 映射到整个类或特定处理程序方法。

```
@Controller
@RequestMapping("/hello")
public class HelloController{

   @RequestMapping(method = RequestMethod.GET)
   public String printHello(ModelMap model) {
      model.addAttribute("message", "Hello Spring MVC Framework!");
      return "hello";
   }

}
```

Java

@Controller 注释将类定义为 Spring MVC 控制器。这里@RequestMapping 的第一个用法表示此控制器上的所有处理方法都与/hello 路径相关。 下一个注释@RequestMapping(method = RequestMethod.GET)用于声明 printHello()方法作为控制器的默认服务方法来处理 HTTP GET 请求。可以定义另一个方法来处理同一 URL 的任何 POST 请求。

可以以另一种形式在上面的控制器中编写，在@RequestMapping 中添加其他属性，如下所示：

```
@Controller
public class HelloController{
```

```
   @RequestMapping(value = "/hello", method = RequestMethod.GET)
   public String printHello(ModelMap model) {
      model.addAttribute("message", "Hello Spring MVC Framework!");
      return "hello";
   }

}
Java
```

value 属性指示处理程序方法映射到的 URL，method 属性定义处理 HTTP GET 请求的服务方法。关于以上定义的控制器，需要注意以下几点：

- 在服务方法中定义所需的业务逻辑。可以根据需要在此方法内调用其他方法。
- 基于定义的业务逻辑，将在此方法中创建一个模型。可以设置不同的模型属性，这些属性将被视图访问以呈现最终结果。此示例创建且有属性“message”的模型。
- 定义的服务方法可以返回一个 String，它包含要用于渲染模型的视图的名称。此示例将“hello”返回为逻辑视图名称。

创建 JSP 视图

Spring MVC 支持许多类型的视图用于不同的表示技术。包括 - JSP，HTML，PDF，Excel 工作表，XML，Velocity 模板，XSLT，JSON，Atom 和 RSS 源，JasperReports 等。但最常见的是使用 JSPL 编写的 JSP 模板，这里使用的是 JSP 模板，并在/WEB-INF/hello/hello.jsp 中写一个简单的 hello 视图：

```
<html>
   <head>
   <title>Hello Spring MVC</title>
   </head>
   <body>
   <h2>${message}</h2>
   </body>
</html>
Java
```

这里${message}是在 Controller 中设置的属性。可以在视图中显示多个属性。