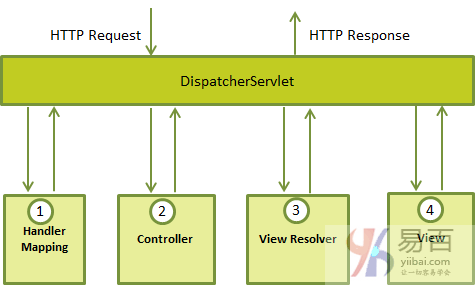
## **DispatcherServlet组件类**

****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](https://github.com/Controller" \o "@Controller" \t "http://www.yiibai.com/spring_mvc/_blank)和[@RequestMapping](https://github.com/RequestMapping" \o "@RequestMapping" \t "http://www.yiibai.com/spring_mvc/_blank)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](https://github.com/Controller" \o "@Controller" \t "http://www.yiibai.com/spring_mvc/_blank)注释指示特定类充当控制器的角色。[@RequestMapping](https://github.com/RequestMapping" \o "@RequestMapping" \t "http://www.yiibai.com/spring_mvc/_blank)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](https://github.com/Controller" \o "@Controller" \t "http://www.yiibai.com/spring_mvc/_blank)注释将类定义为Spring MVC控制器。这里[@RequestMapping](https://github.com/RequestMapping" \o "@RequestMapping" \t "http://www.yiibai.com/spring_mvc/_blank)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](https://github.com/RequestMapping" \o "@RequestMapping" \t "http://www.yiibai.com/spring_mvc/_blank)(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](https://github.com/RequestMapping" \o "@RequestMapping" \t "http://www.yiibai.com/spring_mvc/_blank)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。