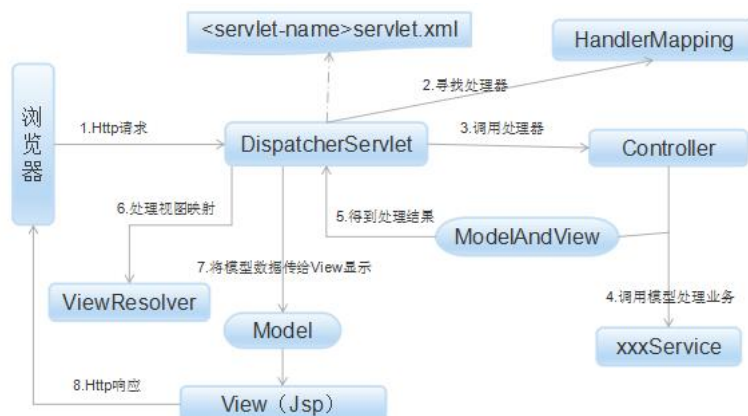


首先，springmvc是基于spring的，是spring提供的一个强大而灵活的web框架，借助于注解，使得控制器的开发和测试更简单

springmvc的组成：DispatcherServlet，处理器映射，控制器，视图解析器，视图
两个核心：处理器映射，视图解析器



SpringMVC接口解释

（1）DispatcherServlet接口：

Spring提供的前端控制器，所有的请求都有经过它来统一分发。在DispatcherServlet将请求分发给Spring Controller之前，需要借助于Spring提供的HandlerMapping定位到具体的Controller。

（2）HandlerMapping接口：

能够完成客户请求到Controller映射。

（3）Controller接口：

需要为并发用户处理上述请求，因此实现Controller接口时，必须保证线程安全并且可重用。

Controller将处理用户请求，这和Struts Action扮演的角色是一致的。一旦Controller处理完用户请求，则返回ModelAndView对象给DispatcherServlet前端控制器，ModelAndView中包含了模型（Model）和视图（View）。

从宏观角度考虑，DispatcherServlet是整个Web应用的控制器；从微观考虑，Controller是单个Http请求处理过程中的控制器，而ModelAndView是Http请求过程中返回的模型（Model）和视图（View）。

（4）ViewResolver接口：

Spring提供的视图解析器（ViewResolver）在Web应用中查找View对象，从而将相应结果渲染给客户。

简单的一个springmvc的例子

springmvc的配置文件

springmvc-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans-3.0.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context-3.0.xsd" >
    <!-- 对web包中的所有类进行扫描，以完成Beans注册 -->
    <context:component-scan base-package="com.jlb" />

    <!-- 这两个类用来自动基于Spring MVC的注解功能，将控制器与方法映射加入容器中 -->
    <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMapping" />
    <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />

    <!-- 这个类用于Spring MVC视图解析 -->
    <bean id="viewResolver"
          class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

web.xml

设置springmvc配置文件的路径

```

<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/springmvc-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

```

添加处理中文传过来乱码的问题

```

<filter>
  <filter-name>characterEncodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>characterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

```

控制器

```

@Controller
public class UserController {

    @RequestMapping("")
    public String Create(Model model) {
        return "create";
    }

    @RequestMapping("/save")
    public String Save(@ModelAttribute("form") User user, Model model) { // user:视图层传给控制层的表单对象;
        System.out.println(user.getName());
        model.addAttribute("user", user);
        return "detail";
    }
}

```

顺便提一下，用model传值的话，只有跳转的那个页面有效，貌似作用域有点和request相同，可以研究研究


持久化类


```

public class User implements Serializable {
    /**
     * @author zjn
     */
    private static final long serialVersionUID = 1L;
    private Integer id; // id
    private String name; // name
    private String pwd; // pwd
    private Integer age; // age
    private Date createTime; // createTime
}

```

两个页面

 create.jsp
925B

 detail.jsp
509B