DispatcherServelet是前端控制器  
HandlerAdapter相当于controller,DispatcherServelet调用各种HandlerAdapter来实现任务分发给相关的业务逻辑  
HandlerInterceptor是一个接口，可以用来在Handler调用之前，之后，以及view呈现后可以做很多事情  
HandlerMapping是负责确定DispatcherServelet与controller之间映射的类，告诉DispatcherServelet，在请求到来后，由哪个controller来响应这个请求

viewResolver视图解释器。ModeAndView模型和视图，mav.addObject(“1”，“2”)表示在视图层的1位置将2传过去，mav.setViewName（”/1.jsp“）表示转到1.jsp视图层。

@RequestMapping是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径.

百度传课内容：

spring mvc框架是一个MVC框架，通过实现MVC很好的将数据、业务、展现进行分离

spring MVC 的设计围绕DispatcherServlet展开的，

由DispatcherServlet负责将请求派发到特定的handler

springmvc 框架搭建步骤：

1. 创建工程，拷贝springmvc相关的jar，并把jar拷贝到lib

2. 配置web.xml,配置前段控制器DispatcherServlet

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

3. 在web-inf 下创建springmvc配置文件，命名规则：[servletname]-servlet.xml

文件名必须为springmvc-servlet.xml

4. 配置HandlerMapping 根据benaname找到对应Conntroller（ 可以省略）

<bean

class="org.springframework.web.servlet.mvc.support.ControllerBeanNameHandlerMapping"></bean>

5. 创建JSP页面，需要发出请求的页面

6. 创建HelloController 继承AbstractController重写handleRequestInternal

//返回数据和页面

ModelAndView mav = new ModelAndView("index");

mav.addObject("helloword", " hello "+hello);

7. <!-- 配置视图解析器 -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<!-- 前缀 -->

<property name="prefix" value="/view/"></property>

<!-- 后缀 -->

<property name="suffix" value=".jsp"></property>

</bean>

SpringMVC参数传递：

请求参数

1. 通过设置HttpServletRequest request 来获取数据

2. 直接设置参数add(@RequestParam(value="number")String userNumber,Integer passWord),方法上注解为@RequestMapping("/add2.do")

3. 通过对象:

表单的name属性的值必须和类的属性名称一致

/\*\*

\* 时间属性编辑器

\*/

@InitBinder

public void initBinder(ServletRequestDataBinder bin){

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

CustomDateEditor cust = new CustomDateEditor(sdf, true);

bin.registerCustomEditor(Date.class,cust);

}

乱码过滤器：

<filter>

<filter-name>encodingFilter</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>

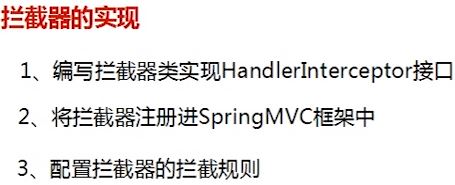
</filter>

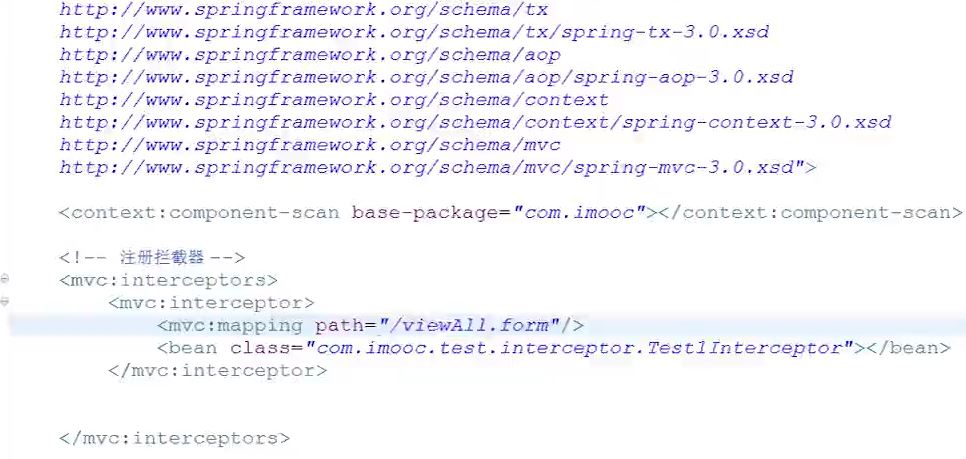
<filter-mapping>

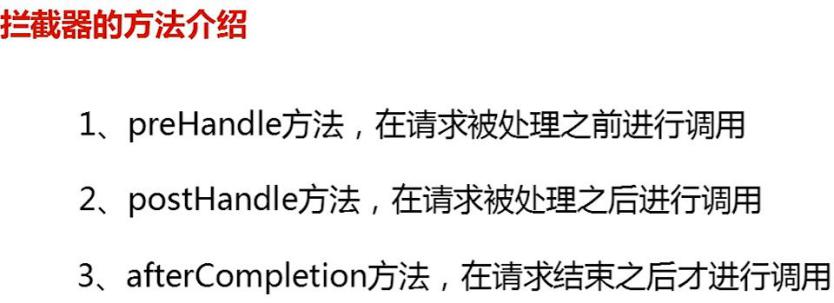
<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

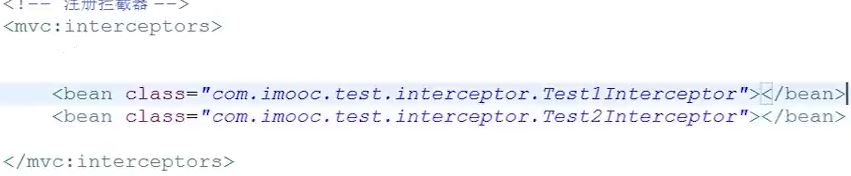


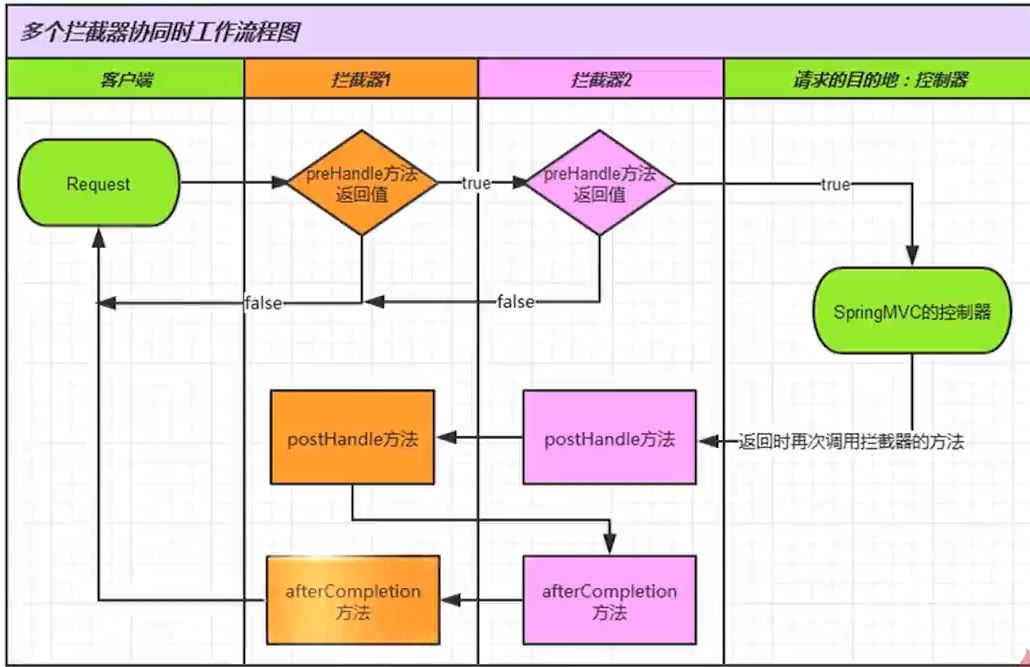


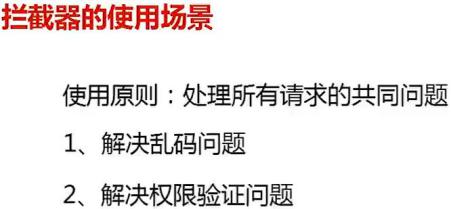




HttpServletRequest包含了所有请求的内容，HttpServletResponse包含了所有响应的内容Object 表示被拦截的请求的目标对象。通过ModelAndView参数来修改显示的视图或发往视图的方法。





拦截器取代过滤器解决乱码问题



