Springmvc01

SpringMVC 和 Struts2 的优劣分析

共同点：

它们都是表现层框架，都是***基于 MVC 模型编写***的。

它们的底层都***离不开原始 ServletAPI***（HttpServletRequest、HttpServletResponse...）。

它们处理***请求的机制都是一个核心控制器***。

区别：

Spring MVC 的核心控制器入口是 Servlet，而 Struts2 是 Filter

Spring MVC 是基于方法设计的，而 Struts2 是基于类，Struts2 每次执行都会创建一个动作类。所以 Spring MVC 会稍微比 Struts2 快些。

组件分析

（1）前端控制器（DispatcherServlet）

用户请求到达前端控制器，它就相当于mvc模式中的c，dispatcherServlet是整个流程控制的中心，由它调用其它组件处理用户的请求，dispatcherServlet的存在降低了组件之间的耦合性。

（2）处理器映射器（HandlerMapping）

HandlerMapping负责根据用户请求找到Handler即处理器，SpringMVC提供了不同的映射器实现不同的映射方式，例如：配置文件方式，实现接口方式，注解方式等。

返回执行链，包含两部分内容：

a) 处理器对象：Handler

b) HandlerInterceptor（拦截器）的集合

（3）处理器适配器（HandlAdapter）

通过HandlerAdapter对处理器进行执行，这是适配器模式的应用，通过扩展适配器可以对更多类型的处理器进行执行。

前端控制器通过HandlerAdapter（处理器适配器）对Handler进行适配包装

不同的适配器，查找不同的Handler，（即不同请求查找所对应Controller中处理的方法）

（4）处理器（Handler）(Contoller)

它就是我们开发中要编写的具体业务控制器。由DispatcherServlet把用户请求转发到Handler。由Handler对具体的用户请求进行处理。

（5）视图解析器（View Resolver）

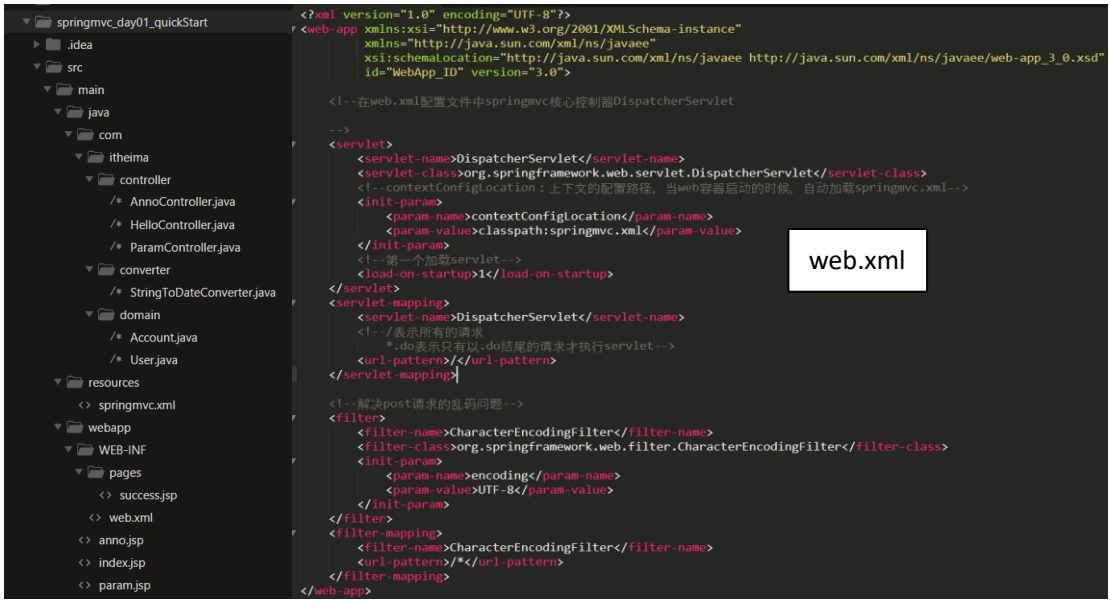
View Resolver负责将处理结果生成View视图，View Resolver首先根据逻辑视图名解析成物理视图名即具体的页面地址，再生成View视图对象，最后对View进行渲染将处理结果通过页面展示给用户。

（6）视图（View）

SpringMVC框架提供了很多的View视图类型的支持，包括：jstlView、freemarkerView、pdfView等。我们最常用的视图就是jsp。

一般情况下需要通过页面标签或页面模版技术将模型数据通过页面展示给用户，需要由程序员根据业务需求开发具体的页面。

Springmvc 入门




```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       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.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">
    <!--1：配置组件的扫描-->
    <context:component-scan base-package="com.itheima"></context:component-scan>
    <!--2：开启springmvc的注解驱动，自动开启处理器映射器和处理器适配器-->
    <mvc:annotation-driven conversion-service="conversionServiceFactory"></mvc:annotation-driven>
    <!--3：视图解析器（后续做项目的时候）    index.jsp->Controller类->/WEB-INF/pages/success.jsp-->
    <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>

    <!--配置类型转换器-->
    <bean id="conversionServiceFactory" class="org.springframework.context.support.ConversionServiceFactoryBean">
        <property name="converters">
            <set>
                <bean class="com.itheima.converter.StringToDateConverter"></bean>
            </set>
        </property>
    </bean>
</beans>
```

springmvc.xm

```java
@Controller
@RequestMapping(value = "/user")
public class HelloController {
    // String的返回值，可以执行视图解析器
    @RequestMapping(value = "/hello")
    public ModelAndView sayHello(){
        System.out.println("执行HelloController类中的sayHello的方法！");
        ModelAndView mv = new ModelAndView();
        mv.setViewName("success");
        return mv;
        // 等同于return "success"
        //return "success"; // 表示视图解析器，前缀和后缀中间的内容
    }
    /**
     * 类上：

     请求URL的第一级访问目录。

     方法上：

     请求URL的第二级访问目录。

     属性：
     value：用于指定请求的URL。它和path属性的作用是一样的。
     method：用于指定请求的方式。一般用在Restful风格的编程（GET、POST、PUT、DELETE）
     params：用于指定限制请求参数的条件。它支持简单的表达式。要求请求参数的key和value必须和配置的一模一样。
     例如：
     params = {"accountName"}，表示请求参数必须有accountName
     params = {"accountName=zhangsan"}，表示请求参数中accountName的值必须是zhangsan。
     params = {"accountName!zhangsan"}，表示请求参数中accountName的值不能是zhangsan。

     headers：用于指定限制请求消息头的条件。
     * @return
     */
    @RequestMapping(value = "/testRequestMapping",method = RequestMethod.GET,params = {"name=zhangsan"},headers = {"Accept"})
    public String testRequestMapping(){
        System.out.println("执行HelloController类中的testRequestMapping的方法！");
        return "success"; // 表示视图解析器，前缀和后缀中间的内容
    }
}
```

在控制器中使用原生的 ServletAPI 对象

【拓展】还支持原生的ServletAPI有：

- HttpServletRequest
- HttpServletResponse
- HttpSession
- java.security.Principal
- Locale
- InputStream
- OutputStream
- Reader
- Writer

常用注解

1：@RequestParam注解（重点）：请求参数，用于指定参数名称

2：@RequestBody注解（重点）：请求体，用于将json数据封装成java类

3：@PathVariable注解（重点）：用于获取restful风格的请求参数

4：@RequestHeader注解（了解）：获取请求头

5：@CookieValue注解（了解）：获取Cookie的值

6：@ModelAttribute注解（了解）：用于在跳转请求方法之前，封装实体

7：@SessionAttributes注解（了解）：将请求数据存放到Sessioin