SpringMVC教程

1. 简述springMVC优点

先说一下什么是MVC和三层架构

1、MVC：Model+View+Controller（数据模型+视图+控制器）

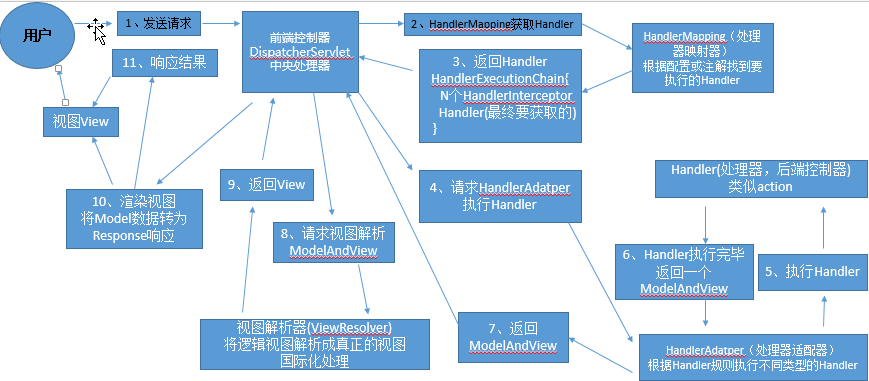
2、三层架构：Presentation tier+Application tier+Data tier(展现层+应用层+数据访问层)

3、MVC只存在三层架构的展现层，M是数据模型，是包含数据的对象，在SpringMVC有一个专门的类叫Model，用来和V之间

的数据交互、传值，V指的是视图页面如jsp等，C就是控制器（SpringMVC的注解@Controller的类）

* 1. 先来看看SpringMVC有哪些优点
     1. 简单易用，提高开发效率
     2. 性能优秀
     3. 学习成本低
     4. 很容易就能写出性能优秀的程序
     5. 灵活性高

二、核心类与接口



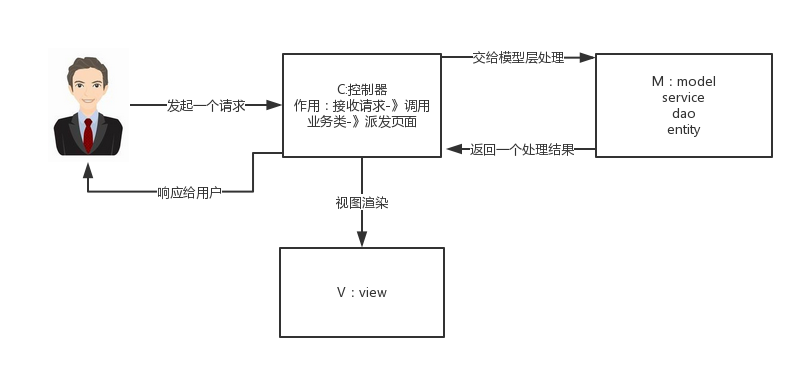
上图是springMVC的工作原理

* + 1. 用户请求发送到前端控制器DispatcherServlet
    2. DispatcherServlet收到请求调用HandlerMapping处理映射器
    3. 处理映射器找到具体的处理器(根据xml或者注解查找)，生成处理器对象及处理器拦截器，(如果有则生成)，一并返回给DispatcherServlet
    4. DispatcherServlet调用HandlerAdapter处理器适配器
    5. HandlerAdapter调用具体的适配器（Controller，也叫后端控制器）
    6. Controller执行完成返回ModelAndView
    7. HandlerAdapter将Controller执行结果ModelAndView返回给DispatcherServlet
    8. DispatcherServlet将ModelAndView传给ViewResolver视图解析器
    9. ViewResolver解析后返回具体的View
    10. DispatcherServlet根据View进行渲染视图（将模型数据填充到视图中）
    11. DispatcherServlet响应客户端
  1. 框架提供实现的组件：
     1. DispatcherServlet：作为前端控制器，整个流程控制中心，控制其他组件执行，统一调度，降低组件之间的耦合性，提高每个组件的扩展性，接收请求，响应请求，相当于MVC模式中的C
     2. HandlerMapping：通过扩展处理器映射器实现不同的映射方式，如：配置文件方式、实现接口方式、注解方式等。根据url查找Handler
     3. HandlerAdapter：通过扩展处理器适配器，支持更多类型的处理器，按照指定规则执行Handler
     4. ViewResolver：通过扩展视图解析器，支持更多类型的视图解析，如jsp、freemarker、pdf、excel等。进行视图解析，根据逻辑视图解析成真正的视图(View)，ViewResolver负责将处理结果生成View视图，ViewResolver首先根据逻辑视图解析名成物理视图名(具体的页面最地址)，再生成View视图对象，最后对View进行视图渲染，将处理结果通过页面展示给客户，SpingMVC提供了很多View视图类型：JstlView、freemarker、pdfView等，一般情况下需要通过页面标签或者页面模板技术将模型数据通过页面展示给客户
     5. Handler：编写Handler要按照HandlerAdapter的规则去做，这样利于适配器正确执行Handler，Handler是继DispatcherServlet前端控制器的后端控制器，在DispatcherServlet的控制下Handler对具体的用户请求进行处理
     6. 视图View：View是一个接口，实现类支持不同的view类型
  2. 核心架构的具体流程：
     1. 首先用户发送请求--🡪DispatcherServlet前端控制器收到请求时不自己处理，而是委托其他的解析器处理，，作为统一访问点，进行全局的流程控制
     2. DispatcherServlet--🡪HandlerMapping会把请求映射为HandlerExceptionChain对象(包含一个Handler处理器(页面控制器)对象，多个HandlerInterceptor拦截器)对象，通过这种策略很容易添加新的映射策略
     3. DispatcherServlet--🡪HandlerAdapter会把处理器包装成适配器，支持多种类型的处理器，即适配器设计模式的使用，从而支持多种类型的处理器
     4. HandlerAdapter--🡪处理器功能处理方法的调用，HandlerAdapter会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理，并返回一个ModelAndView对象（包含模型数据、逻辑视图名）
     5. ModelAndView的逻辑视图名--🡪ViewResolver把逻辑视图名解析为具体的View通过这种策略模式很容易更换其他视图技术
     6. View--🡪渲染，View会根据传进来的Model模型数据进行渲染，此处的Model实际是一个Map数据结构，因此很容易支持其他视图技术
     7. 返回控制权给DispatcherServlet然后返回给用户

**在将SpringMVC之前我们先来看一下什么是MVC模式**

MVC：MVC是一种设计模式

MVC的原理图：



**分析：**

M-Model 模型（完成业务逻辑：有javaBean构成，service+dao+entity）

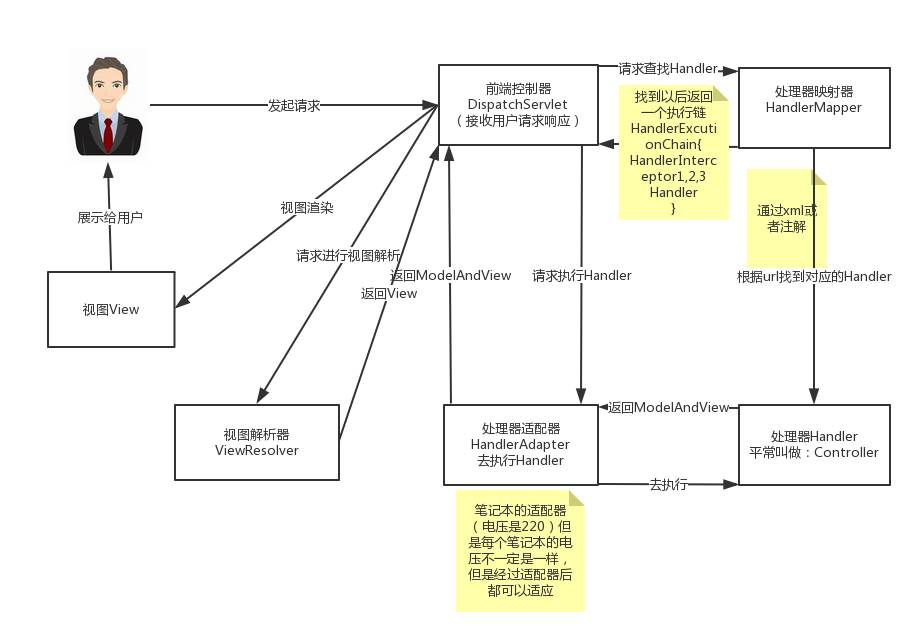
V-View 视图（做界面的展示  jsp，html……）

C-Controller 控制器（接收请求—>调用模型—>根据结果派发页面）

**springMVC是什么：**

springMVC是一个MVC的开源框架，springMVC=struts2+spring，springMVC就相当于是Struts2加上sring的整合，但是这里有一个疑惑就是，springMVC和spring是什么样的关系呢？这个在百度百科上有一个很好的解释：意思是说，springMVC是spring的一个后续产品，其实就是spring在原有基础上，又提供了web应用的MVC模块，可以简单的把springMVC理解为是spring的一个模块（类似AOP，IOC这样的模块），网络上经常会说springMVC和spring无缝集成，其实springMVC就是spring的一个子模块，所以根本不需要同spring进行整合。

**SpringMVC的原理图：**

****

**看到这个图大家可能会有很多的疑惑，现在我们来看一下这个图的步骤：（可以对比MVC的原理图进行理解）**

第一步:用户发起请求到前端控制器（DispatcherServlet）

第二步：前端控制器请求处理器映射器（HandlerMappering）去查找处理器（Handle）：通过xml配置或者注解进行查找

第三步：找到以后处理器映射器（HandlerMappering）像前端控制器返回执行链（HandlerExecutionChain）

第四步：前端控制器（DispatcherServlet）调用处理器适配器（HandlerAdapter）去执行处理器（Handler）

第五步：处理器适配器去执行Handler

第六步：Handler执行完给处理器适配器返回ModelAndView

第七步：处理器适配器向前端控制器返回ModelAndView

第八步：前端控制器请求视图解析器（ViewResolver）去进行视图解析

第九步：视图解析器像前端控制器返回View

第十步：前端控制器对视图进行渲染

第十一步：前端控制器向用户响应结果

**看到这些步骤我相信大家很感觉非常的乱，这是正常的，但是这里主要是要大家理解springMVC中的几个组件：**

前端控制器（DispatcherServlet）：接收请求，响应结果，相当于电脑的CPU。

处理器映射器（HandlerMapping）：根据URL去查找处理器

处理器（Handler）：（需要程序员去写代码处理逻辑的）

处理器适配器（HandlerAdapter）：会把处理器包装成适配器，这样就可以支持多种类型的处理器，类比笔记本的适配器（适配器模式的应用）

视图解析器（ViewResovler）：进行视图解析，多返回的字符串，进行处理，可以解析成对应的页面

建立一个springMVC项目，这里以maven项目举例

**Pom.xml**

<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"**>  
 <**parent**>  
 <**artifactId**>ideaproject</**artifactId**>  
 <**groupId**>com.lsh</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**parent**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
 <**artifactId**>springmvcdemo1</**artifactId**>  
 <**packaging**>war</**packaging**>  
 <**name**>springmvcdemo1 Maven Webapp</**name**>  
 <**url**>http://maven.apache.org</**url**>

<**build**>  
 *<!--编译的时候xml没有编译进去-->* <**resources**>  
 <**resource**>  
 <**directory**>src/main/java</**directory**>  
 <**includes**>  
 <**include**>\*\*/\*.xml</**include**>  
 </**includes**>  
 <**filtering**>true</**filtering**>  
 </**resource**>  
 <**resource**>  
 <**directory**>src/main/resources</**directory**>  
 <**includes**>  
 <**include**>\*\*/\*.xml</**include**>  
 <**include**>\*\*/\*.properties</**include**>  
 </**includes**>  
 <**filtering**>true</**filtering**>  
 </**resource**>  
 </**resources**>  
 *<!--这是主动设置jdk compiler-->* <**plugins**>  
 <**plugin**>  
 <**groupId**>org.apache.maven.plugins</**groupId**>  
 <**artifactId**>maven-compiler-plugin</**artifactId**>  
 <**version**>3.5.1</**version**>  
 <**configuration**>  
 <**source**>1.8</**source**>  
 <**target**>1.8</**target**>  
 </**configuration**>  
 </**plugin**>  
 </**plugins**>  
</**build**>

<**dependencies**>

<**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-web</**artifactId**>  
 <**version**>5.0.4.RELEASE</**version**>  
</**dependency**>

<**dependency**>  
 <**groupId**>javax.servlet</**groupId**>  
 <**artifactId**>jstl</**artifactId**>  
 <**version**>1.2</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-webmvc</**artifactId**>  
 <**version**>5.0.4.RELEASE</**version**>  
</**dependency**>

<**dependency**>  
 <**groupId**>javax.servlet</**groupId**>  
 <**artifactId**>javax.servlet-api</**artifactId**>  
 <**version**>3.1.0</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>javax.servlet.jsp</**groupId**>  
 <**artifactId**>jsp-api</**artifactId**>  
 <**version**>2.2</**version**>  
</**dependency**>

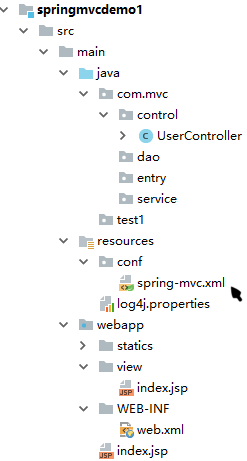
</**dependencies**>

**下面是web.xml**

*<?***xml version="1.0" encoding="UTF-8"***?>*<**web-app version="3.0"  
 xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
 http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"**>

*<!--配置springMVC 的 DispatcherServlet-->*<**servlet**>  
 <**servlet-name**>springMVC</**servlet-name**>  
 <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  
 *<!--加载配置文件-->* <**init-param**>  
 <**param-name**>contextConfigLocation</**param-name**>  
 <**param-value**>classpath:conf/spring-mvc.xml</**param-value**>  
 </**init-param**>  
 <**load-on-startup**>1</**load-on-startup**>  
 <**async-supported**>true</**async-supported**>  
</**servlet**>  
<**servlet-mapping**>  
 <**servlet-name**>springMVC</**servlet-name**>  
 <**url-pattern**>/</**url-pattern**>  
</**servlet-mapping**>

**下面是目录结构**



**下面是spring-mvc.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:p="http://www.springframework.org/schema/p"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  
 http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.2.xsd  
 http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.2.xsd  
 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd  
 http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd  
"**>

*<!--自动扫描包 使SpringMVC认为@Controller注解的是控制器-->*<**context:component-scan base-package="com.mvc"**>

如果没有下面这一句则会出现扫不到控制层，访问的资源会报错  
 <**context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"** />  
</**context:component-scan**>

*<!--  
 静态资源映射  
 把项目的静态资源放到statics文件夹下  
-->  
<!--<mvc:resources mapping="/js/\*\*" location="/statics/js/" />-->*

*<!--  
 对模型视图名称的解析，即为模型视图名称添加前后缀，文件夹最后一个斜杠（/）不能少 这里使用jsp  
 默认的视图解析器 在上边的解析错误时使用（默认使用html）  
-->*<**bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 *<!--<property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />-->* <**property name="prefix" value="/view/"** />  
 <**property name="suffix" value=".jsp"** />  
</**bean**>

其他的和spring的配置无异了

下面是controller层

**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestMethod;  
  
@Controller  
*//@RequestMapping("/u")***public class** UserController {  
*// 添加一个日志器  
// private static final Logger LOGGER= LoggerFactory.getLogger(UserController.class);* @RequestMapping(value = **"/toindex"**)*//映射一个action* **public** String toindex(){  
*// LOGGER.info("首页~~~");//输出日志文件* System.***out***.println(**"jinlai"**);  
 **return "index"**;*//返回index.jsp* }  
}