springMVC的经典示例

# SpringMVC实例的步骤

### 加入jar包或Maven添加依赖；

### 在web.xml中配置DispatcherServlet；

### 添加springmvc的配置文件，如springmvc-config.xml;

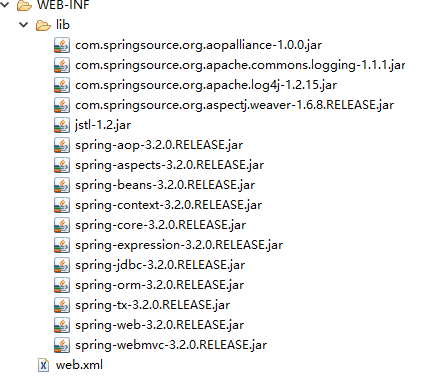
### 编写请求处理器Handler并标识为控制器@Controller;

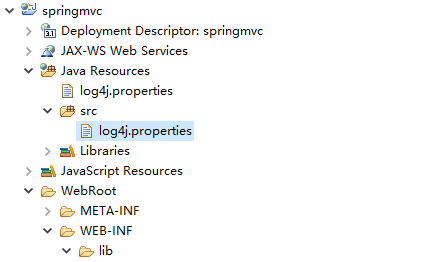
### 编写视图jsp文件。

# 普通Web工程：springmvc的第一个案例

## 创建一个web工程：

环境搭建：





## 第一个案例的分析：

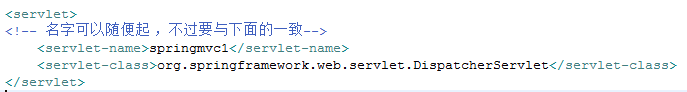
两个配置文件：

* 1. **web.xml** : Web应用的所有都在这里加载。
  2. springmvc.xml ：web应用默认在WEB-INF下寻找projectname-servlet.xml文件，但是如果不想设置这个默认名字和路径，可以在web.xml中指定：



## 前端控制器：拦截。

**在web.xml文件中配置**：





## 在springmvc.xml中配置处理器映射器：

（1）这个是默认的**处理器映射器**，不设置默认也是这个。

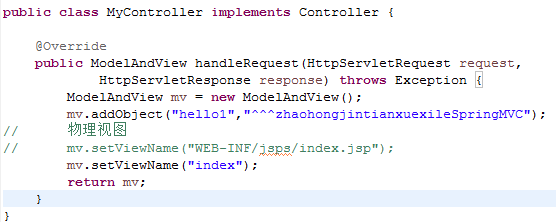


（2） 在springmvc.xml中创建bean，就是一个映射，由name映射到Controller。

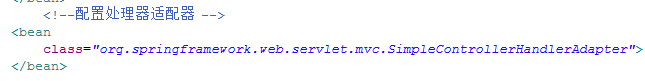




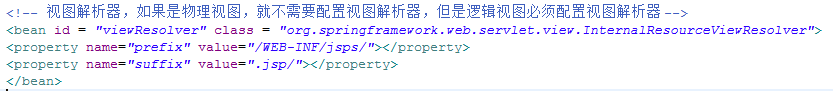
## 创建Controller：



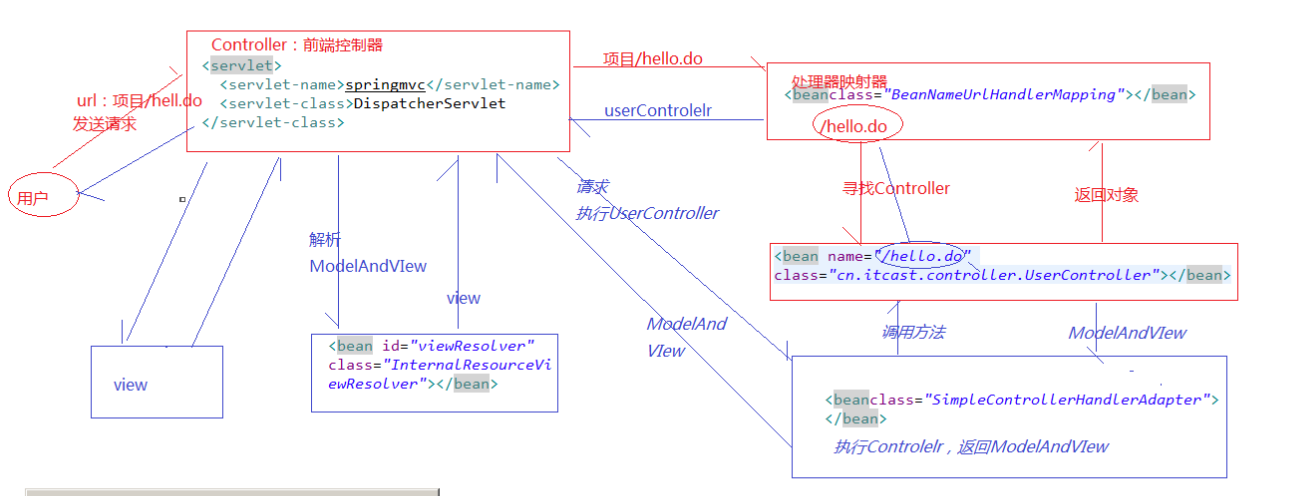
## 在springmvc.xml中配置处理器适配器：（适配器设计模式）



## 在springmvc.xml中配置视图解析器：



## 代码层面的执行流程：

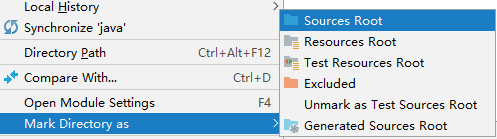


# 利用IDEA创建的Maven-Web工程

## 创建Maven-webapp工程

由于IDEA没有java目录，新建java目录，然后右击选择Mark Directory as==>**Source Root**。

否则无法在java内部创建package、Class文件。



## pom文件，添加依赖

主要是spring的依赖：core、context、beans、expression、web、webmvc等；还有common-logging；javax-servlet等；

这里为了方便版本管理，使用<properties>指定版本：

<**properties**>  
 <**spring.version**>4.1.7.RELEASE</**spring.version**>  
 <**servlet.version**>3.1.0</**servlet.version**>   
</**properties**>

<**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-core</**artifactId**>  
 <**version**>**${spring.version}**</**version**>  
</**dependency**>

## 配置web.xml文件

最关键配置DispatcherServlet:通过<servlet>和<servlet-mapping>配置

注意：**contextConfigLocation参数名是固定的。**

*<!-- 定义Spring MVC的前端控制器 -->*<**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**>  
 *<!--DispatcherServlet的启动级别-->* <**load-on-startup**>1</**load-on-startup**>  
</**servlet**>  
*<!-- 让Spring MVC的前端控制器拦截所有请求 -->*<**servlet-mapping**>  
 <**servlet-name**>springmvc</**servlet-name**>  
 <**url-pattern**>/</**url-pattern**>  
</**servlet-mapping**>

对于其他的filter、listener、welcome-file-list、error-page等，可以自由扩展，见web.xml讲解。

## 配置springmvc-config.xml文件

**spring-servlet.xml**文件的名字可以随意起。Spring框架默认使用的配置文件名称为**applicationContext.xml**。

重点配置：注解的扫描包。

还有：处理器映射器、处理器适配器(有默认)、视图解析器。

*<!-- spring可以自动去扫描base-pack下面的包或者子包下面的java文件，  
 如果扫描到有Spring的相关注解的类，则把这些类注册为Spring的bean -->*<**context:component-scan base-package="org.springmvc.controller"**/>  
<**context:annotation-config**/>  
*<!-- 配置处理映射器 -->*<**bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"**/>  
*<!-- 配置处理器适配器-->*<**bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"**/>  
*<!-- 视图解析器 -->*<**bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <**property name="prefix" value="/WEB-INF/content/"**></**property**>  
 <**property name="suffix" value=".jsp"**></**property**>  
</**bean**>

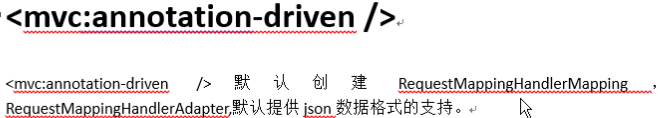
## 写Controller的Handler方法

*/\*\*  
 \* HelloController是一个基于注解的控制器,  
 \* 可以同时处理多个请求动作，并且无须实现任何接口。  
 \* org.springframework.stereotype.Controller注解用于指示该类是一个控制器  
 \*/*@Controller  
**public class** HelloController{  
 */\*\*  
 \* org.springframework.web.bind.annotation.RequestMapping注解  
 \* 用来映射请求的的URL和请求的方法等。本例用来映射"/hello"  
 \* hello只是一个普通方法。  
 \* 该方法返回一个包含视图路径或视图路径和模型的ModelAndView对象。  
 \* \*/* @RequestMapping(value=**"/hello"**)  
 **public** ModelAndView hello(){  
 *// 创建准备返回的ModelAndView对象，该对象通常包含了返回视图的路径、模型的名称以及模型对象* ModelAndView mv = **new** ModelAndView();  
 */\*添加模型数据 可以是任意的POJO对象\*/* mv.addObject(**"message"**, **"Hello World!"**);  
 *// 设置逻辑视图名，视图解析器会根据该名字解析到具体的视图页面* mv.setViewName(**"welcome"**);  
 *// 返回ModelAndView对象。* **return** mv;  
 }  
 @ResponseBody  
 @RequestMapping(**"/getBody"**)  
 **public** String getBody(){  
 **return "Body:@ResponseBody"**;  
 }  
}

# <mvc:annotation-driven /> 说明:

**<mvc:annotation-driven />**是一种**简写形式**，可以让初学者快速成应用**默认的配置方案**，会默认注册 **DefaultAnnotationHandleMapping**以及**AnnotionMethodHandleAdapter** 这两个 Bean, 这两个 Bean ，前者对应**类**级别， 后者对应到**方法**级别；

上面的 DefaultAnnotationHandlerMapping和AnnotationMethodHandlerAdapter 是 Spring 为 @Controller 分发请求所必需的。





不需要再**单独配置处理器映射器、处理器适配器**，还有对JSON的支持。