# 1.SpringMvc初体验

## 新建Dynamic Web工程。

## 导入jar包

|  |
| --- |
|  |

## 3.编写springmvc.xml的 约束

|  |
| --- |
| <?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"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  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*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx.xsd"*>  </beans> |

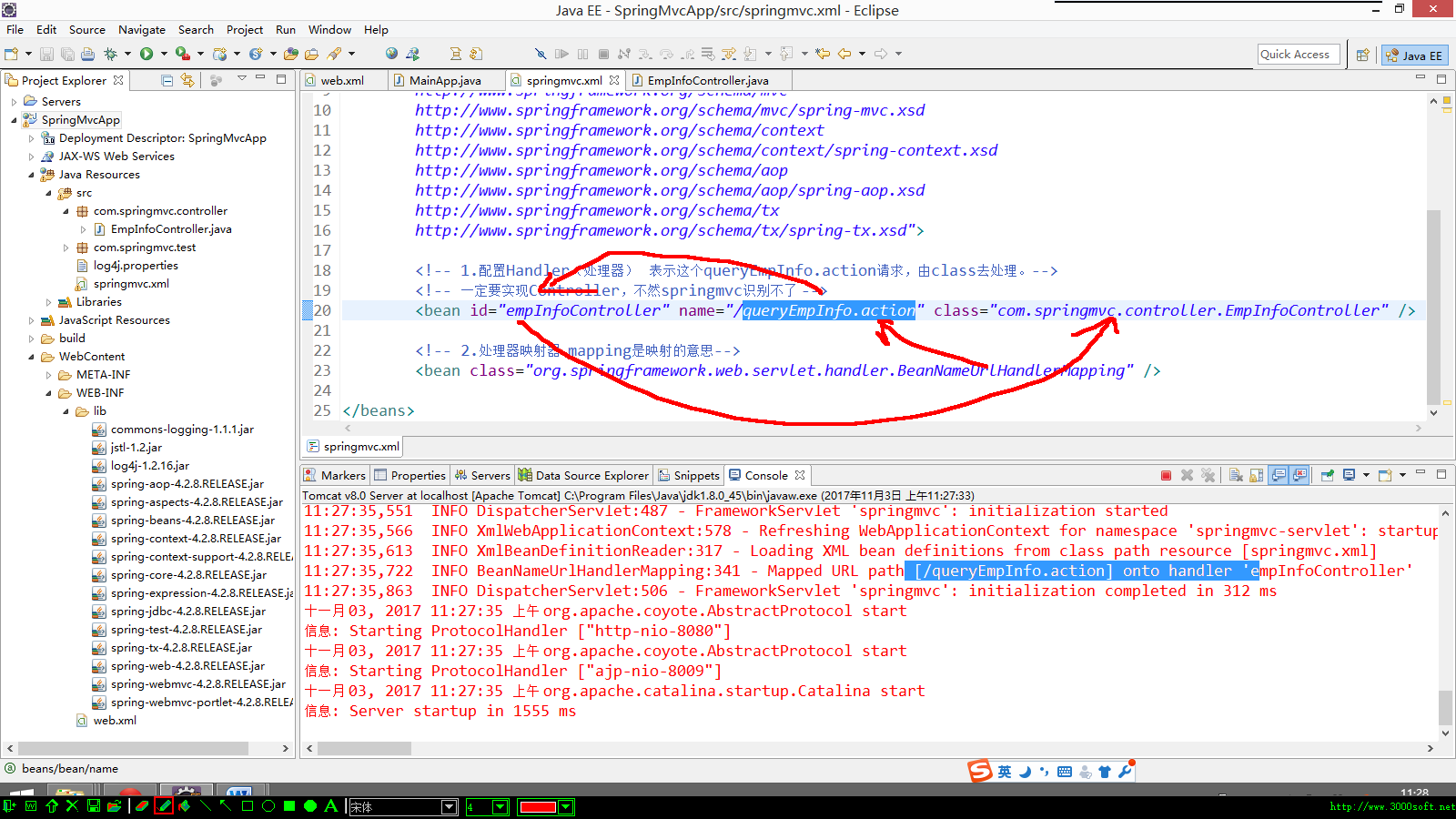
## 4.编辑web.xml文件，配置前端控制器（DispatcherServlet）并加载springmvc.xml配置。

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* id=*"WebApp\_ID"* version=*"3.0"*>  <display-name>SpringMvcApp</display-name>  <!-- 配置前端控制器 -->  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc.xml</param-value>  </init-param>  <!-- load-on-startup表示servlet随服务器启动加载 -->  <load-on-startup>1</load-on-startup>  </servlet>    <!-- 配置springmvc的映射规则 -->  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <!-- / 是匹配所有请求,也会拦截的所有的静态资源 -->  <!-- <url-pattern>/</url-pattern> -->  <!-- \*.action \*.do等等 拦截所有已action、do结尾的请求 -->  <url-pattern>\*.action</url-pattern>  </servlet-mapping>    </web-app> |

## 5.编辑springmvc.xml文件。里面要配置处理器映射器、处理器适配器、处理器。

### 5.1配置处理器handler，也就是说你这个请求使用哪个类去处理，要手动实现controller接口。

|  |
| --- |
| <!-- 1.配置Handler（处理器） 表示这个queryEmpInfo.action请求，由class去处理。-->  <bean id=*"empInfoController"* name=*"/queryEmpInfo.action"* class=*"com.springmvc.controller.EmpInfoController"* /> |



### 5.2配置处理器映射器

|  |
| --- |
| <!-- 2.处理器映射器 -->  <bean class=*"org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"* /> |

## 6.编写Handler类，本例中是EmpInfoController.java，该类需要实现controller接口。

|  |
| --- |
| package com.springmvc.controller;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.web.servlet.ModelAndView;  import org.springframework.web.servlet.mvc.Controller;  public class EmpInfoController implements Controller{  @Override  public ModelAndView handleRequest(HttpServletRequest arg0,  HttpServletResponse arg1) throws Exception {  System.out.println("spring的非注解开发");  return null;  }  } |

# 2.SpringMvc基础讲解。

## 1.MVC是什么意思？

### Mvc是一种分层结构，是将一个软件系统拆分成三层结构：

C：Controller，作为系统的控制器来使用，不直接操作数据库或者前台界面。

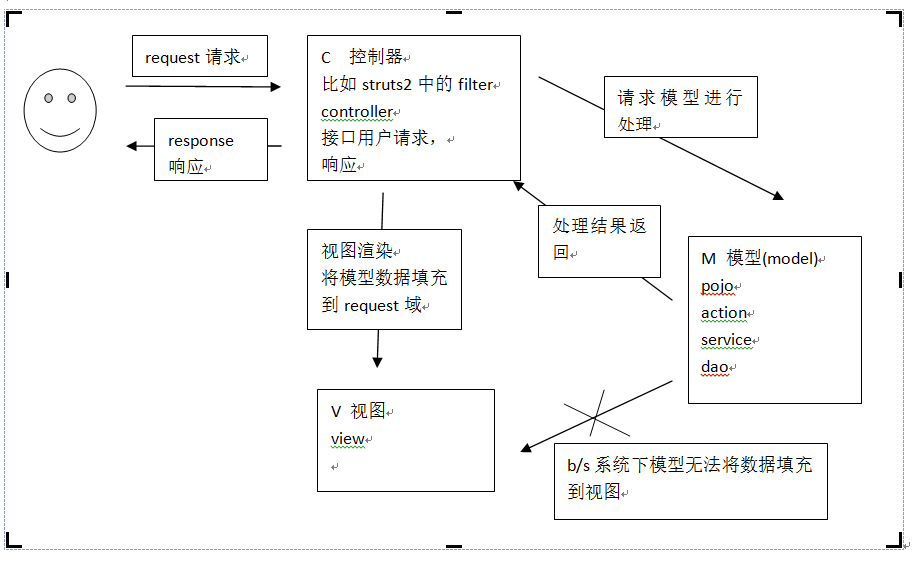
M：Model，作为系统的数据承载或者加载角色。是用来操作数据库里面的数据crud。

V：View,扮演系统的展示角色，直接和用户做交互展示。客户能直接接触的就是view。

### Mvc的好处是什么？

系统代码层次更加清晰，代码更容易维护，各司其职。

### Mvc流程图：



#### 以一个查询员工信息的需求为例分析：

Controller里面一般会通过service对象去调用dao对象，然后dao对象在将响应的数据返回给service对象，service在返回给controller对象。

#### 1.用户发起一个查询员工信息的请求到后台服务器，服务器使用Controller（控制器）接受这个请求。

#### 2.没有遵循mvc结构的时候，我们是在controller里面直接操作数据库，但现在mvc结构的话，就不会直接controller里面操作数据，而是将请求委托给model层去操作。然后将查询的结构返回给controller对象。

Model层里面有哪些东西？

Model对象pojo,也就是个javabean，用来承载数据。

Service对象。业务层，对业务做一个具体的实现。

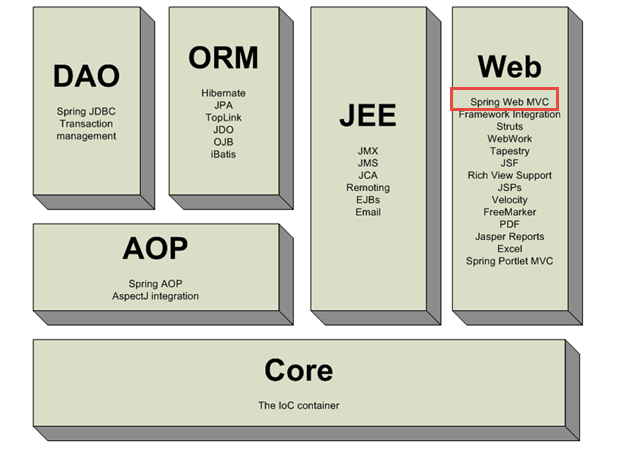
Dao对象，直接操作数据库。

#### 3.view：Controller拿到数据之后，在对数据渲染，填充到view（jsp/html/word/excel/pdf）里面。

## 2.SpringMvc是什么？

Springmvc是spring框架下的一套子框架，它必须依赖spring才能运行。是遵循mvc架构的一套框架，主要用于web系统里面的mvc框架，Springmvc和spring之间不需要中间框架来整合。

## 3.Spring结构图：



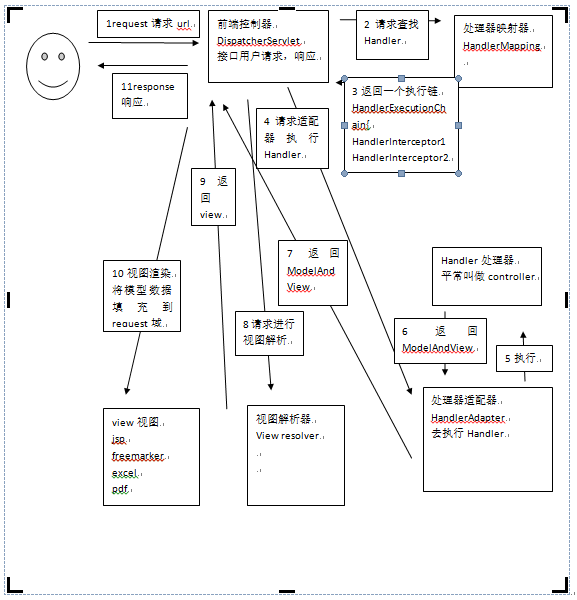
### Core：IOC功能，主要spring的核心功能。

### AOP: Spring Aop /AspectJ

### WEB:SpringMVC

### DAO:SpringJdbc/Spring Data

## 4.SpringMvc的运行原理：



**SpringMvc的流程说明：**

#### 以一个查询员工信息的需求为例分析：

Controller里面一般会通过service对象去调用dao对象，然后dao对象在将响应的数据返回给service对象，service在返回给controller对象。

#### 第一步:用户发起一个请求，通过springmvc的配置，请求会被DispathcerServlet（中央控制器）拦截。因为我们已经在web.xml文件里面配置servlet映射，而这个servlet我们制定是DispatcherServlet，所以url-patten里面配置规范请求都会被DispatcherServlet拦截下来。

|  |
| --- |
| <!-- 配置前端控制器 -->  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc.xml</param-value>  </init-param>  <!-- load-on-startup表示servlet随服务器启动而加载 -->  <load-on-startup>1</load-on-startup>  </servlet>    <!-- 配置springmvc的映射规则 -->  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <!-- / 是匹配所有请求,也会拦截的所有的静态资源 -->  <!-- <url-pattern>/</url-pattern> -->  <!-- \*.action \*.do等等 拦截所有已action、do结尾的请求 -->  <url-pattern>\*.action</url-pattern>  </servlet-mapping> |

#### 第二步：请求来了之后，DispatcherServlet就会指定一个类去处理，那么指定谁去处理？所以这个地方就需要一个匹配的规则，不然DispatcherServlet怎么去找到相应的处理类（Handler），也就是我们平时叫Controller类。

#### 第三步：既然是要映射规则，所以需要配置，Springmvc里面就通过HandlerMapping处理器映射器。DispatcherServlet调用这个处理器之后，处理器映射器就会返回一个执行链给DispatcherServlet。

#### 第四步：前端控制器拿到这个执行链之后，然后在去适配。所以这个需要处理器适配器来进行handler的适配。DispatcherServlet就委托处理器适配器去执行handler，也就是执行我们的controller类。

#### 第五步：handler类就是执行请求，执行完成之后返回一个结果，这个结果就是springmvc的一个底层对象，叫做ModelAndView。

#### 第六步：handler类将处理的结果ModelAndView对象返回处理器适配器。

#### 第七步：处理器适配器得到handler返回的ModelAndView对象，然后将该对象返回给DispatcherServlet.

#### 第八步：DispatcherServlet拿到处理器适配器返回的 ModelAndView对象之后，再对ModelAndView解析，调用的springmvc的一个底层对象，叫做ViewResource。

#### 第九步：视图解析器解析完成之后，返回给DispatcherServelt一个视图（view）。

#### 第十步：DispatcherServlet拿到视图之后，再进行视图渲染，将数据填充到请求域。

#### 十一步：DispatcherServlet在最终的视图响应给用户。

## 5.SpringMvc组件：

### 1、前端控制器DispatcherServlet（不需要程序员开发）

作用接收请求，响应结果，相当于转发器，中央处理器。

有了DispatcherServlet减少了其它组件之间的耦合度。

DispatcherServlet配置在web.xml文件里面，作用拦截请求，担任中央控制的调度角色。

### 2、处理器映射器HandlerMapping(不需要程序员开发)

作用：根据请求的url查找Handler，解析请求和handler 的映射规则。

### 3、处理器适配器HandlerAdapter

作用：按照特定规则（HandlerAdapter要求的规则）去执行Handler，担任DsipathcerServlet和Handler执行过程中的中介。

### 4、处理器Handler，也即是Controller类(需要程序员开发)。

注意：编写Handler时按照HandlerAdapter的要求去做，这样适配器才可以去正确执行Handler

### 5、视图解析器View resolver(不需要程序员开发)

作用：进行视图解析，根据逻辑视图名解析成真正的视图（view）

### 6、视图View(需要程序员开发jsp)

View是一个接口，实现类支持不同的View类型（jsp、freemarker、pdf...）

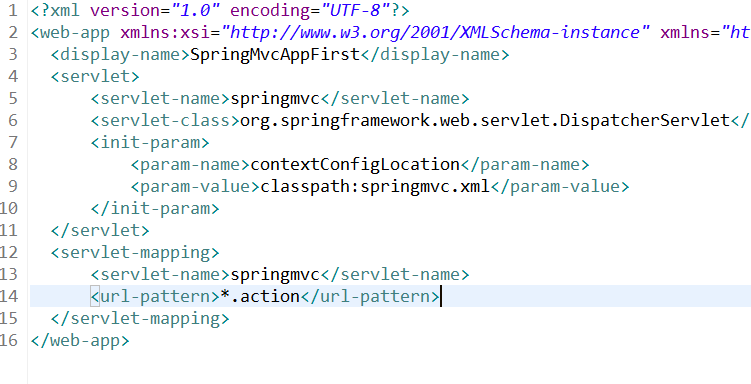
# 3.SpringMvc入门程序。

## 1.新建动态web程序，添加web.xml文件。

## 2.导入jar包



## 3.开始配置web.xml文件。在web.xml文件里面加载springmvc的配置文件。



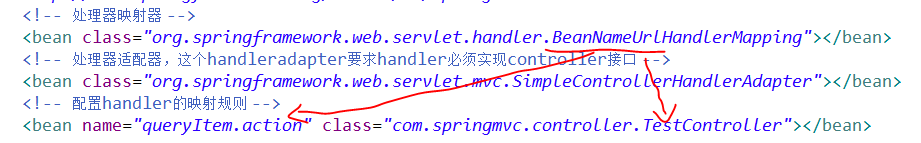
## 4.配置处理器映射器 我们这个地方使用的xml配置的方法，先不讲注解开发，但是重点要掌握注解开发，在springmvc.xml文件里面配置处理器映射器。



## 5.配置处理器适配器。



## 6.编写handler类，也就是平常controller类。



|  |
| --- |
| **package** com.springmvc.controller;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **import** org.springframework.web.servlet.ModelAndView;  **import** org.springframework.web.servlet.mvc.Controller;  **public** **class** TestController **implements** Controller {  @Override  **public** ModelAndView handleRequest(HttpServletRequest arg0,  HttpServletResponse arg1) **throws** Exception {  System.***out***.println("springmvc的配置开发！！");  **return** **null**;  }  } |

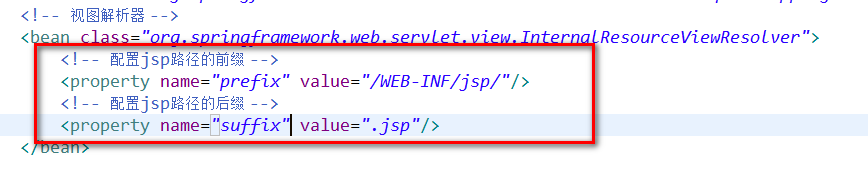
## 6.返回ModelAndView对象。



## 7.配置视图解析器，在springmvc的配置里面配置。

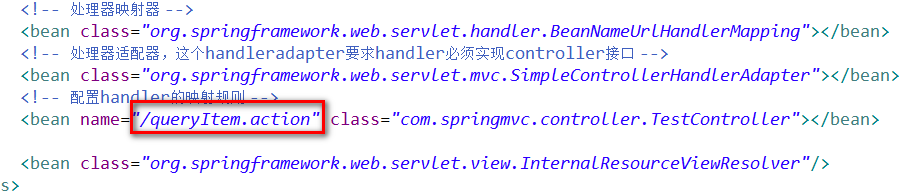


在视图解析器里面配置jsp文件的位置信息。



## 8.将项目配置在tomcat中运行。

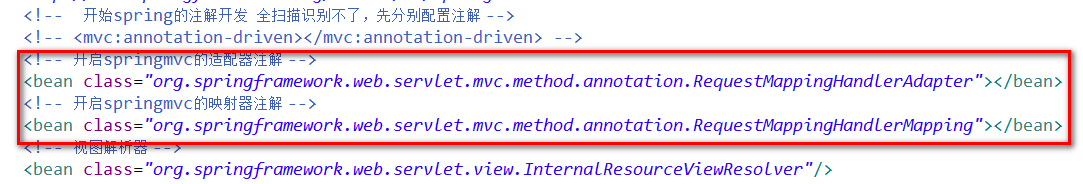
访问的链接是：



<http://localhost:8080/SpringMvcAppFirst/queryItem.action>

# 4.SpringMvc的注解开发。

## 1.配置SpringMvc的配置文件，开发注解功能。



注解说明：

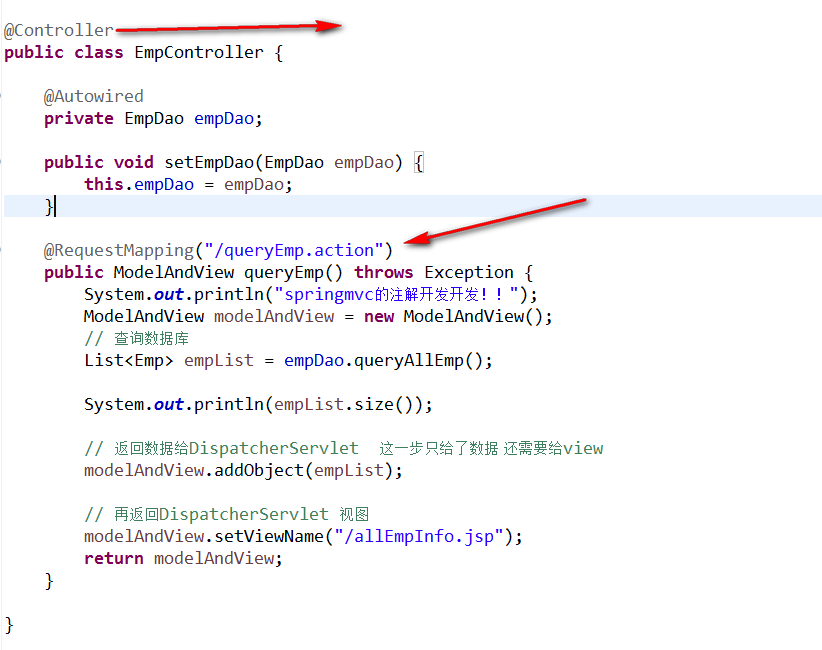
### 在spring3.1之前使用org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping注解映射器。

### 在spring3.1之后使用org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping注解映射器。

### 在spring3.1之前使用org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter注解适配器。

### 在spring3.1之后使org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter注解适配器。

## 2.开发注解的handler，也就是平时说的Controller类。



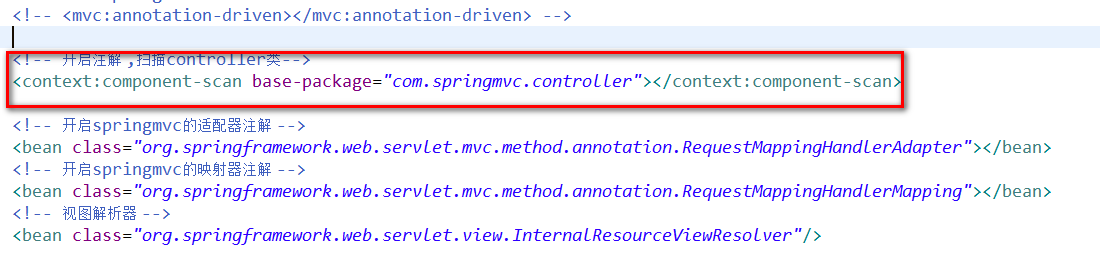
注解说明：

### @Controller 注解，说明此类是controller，需要在spring的配置文件开启注解扫描

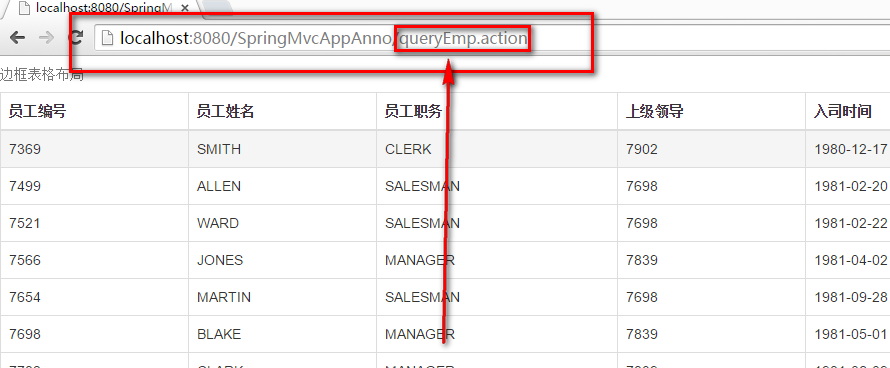
@RequestMapping实现 对queryEmp方法和url进行映射，一个方法对应一个url

### 一般建议将url和方法写成一样

## 3.在springmvc.xml的配置文件中开启注解功能。



## 4.部署调试



<http://localhost:8080/SpringMvcAppAnno/queryEmp.action>

# 5.Spring、SpringJdbc、SpringMvc整合思路

## 1. SpringJdbc操作数据库。db.properties

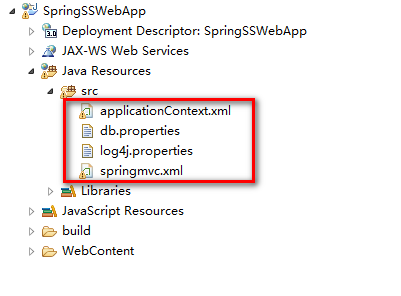
Dao/Service JdbcTemplate

## 2. Spring实例化bean和加载配置文件。applicationContext.xml

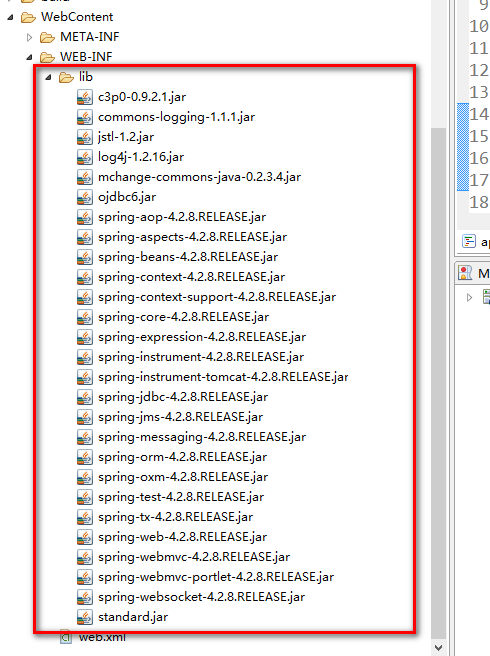
## 3. SpringMvc处理和响应请求web springMvc.xml

## 4.整合过程：

### 第一步：先建立好各自的配置文件。db.propertis/applicationContent.xml/springMvc/log4j.xml,这四个文件都是放在src类路径下面。



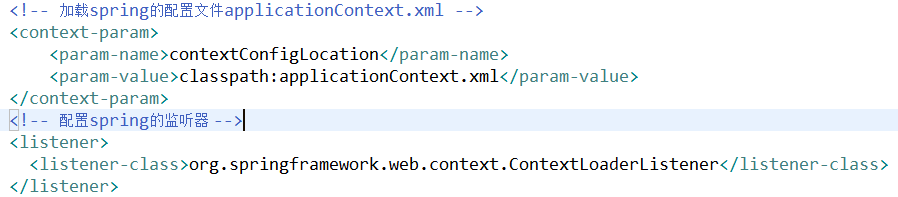
### 第二步：添加各自的jar包



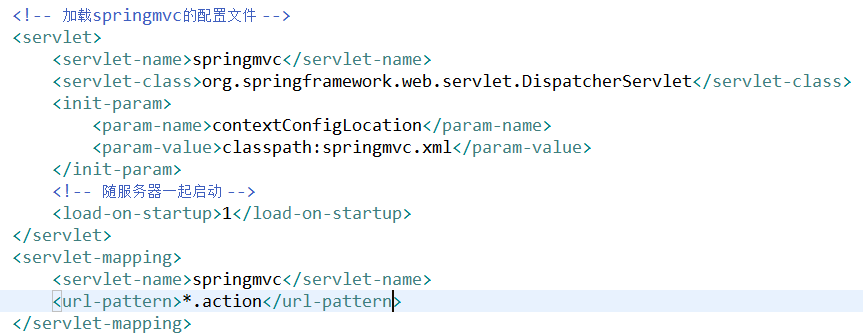
Jar放在web-inf下面的lib文件夹里面。

### 第三步：在web.xml里面启动springmvc框架的配置文件springmvc.xml和spring框架的配置文件applicationContext.xml，配置随服务器启动。

#### 1.加载spring的配置文件。



#### 2.加载springmvc的配置文件。



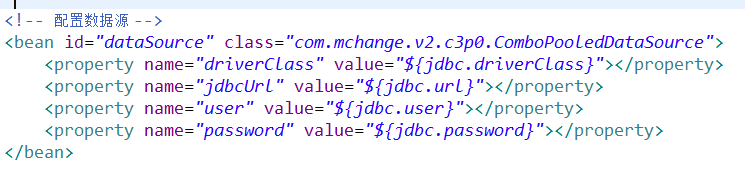
### 第四步：整合spring和springJdbc,使用c3po数据池。

#### 在spring的配置文件applicationContext.xml里面整合。

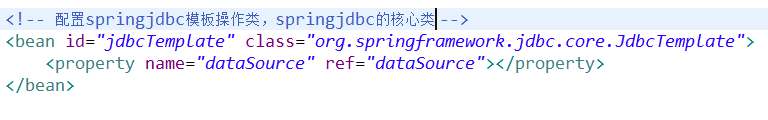
#### 1.加载数据库信息配置文件db.properties



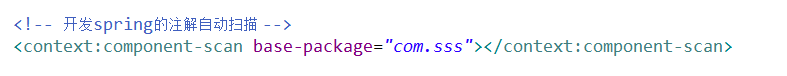
#### 2.配置数据源



#### 3.整合spirngjdbc核心操作类。



#### 4.在spring的配置文件里面开启注解扫描。

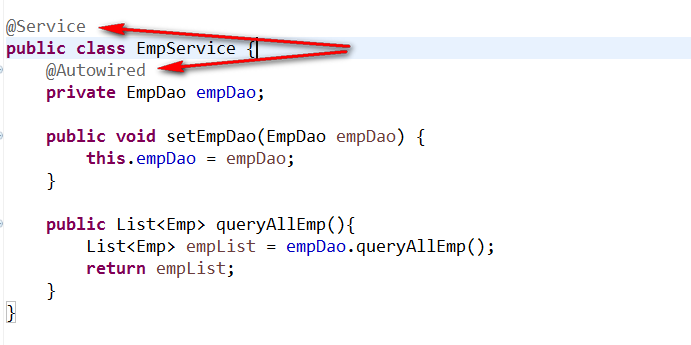


#### 5.测试spring和springjdbc的整合

**编写Dao类，直接操作数据库**



**编写Service类，业务逻辑处理。**



#### 6.编写测试类，输出结果。



#### 结论：Spring和springjdbc的整合已经完成，测试类通过。

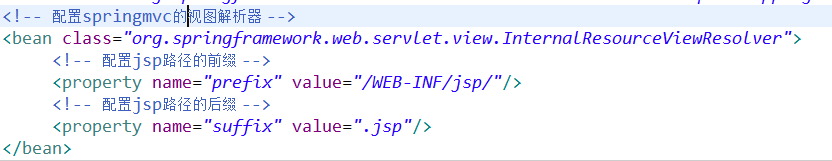
### 第五步：spring、 springmvc无缝整合。编辑springmvc的配置文件springmvc.xml文件。

#### 1.配置注解适配器RequestMappingHandlerAdater



#### 2.配置注解映射器RequestMappingHandlerMapping

#### 3.配置视图解析器



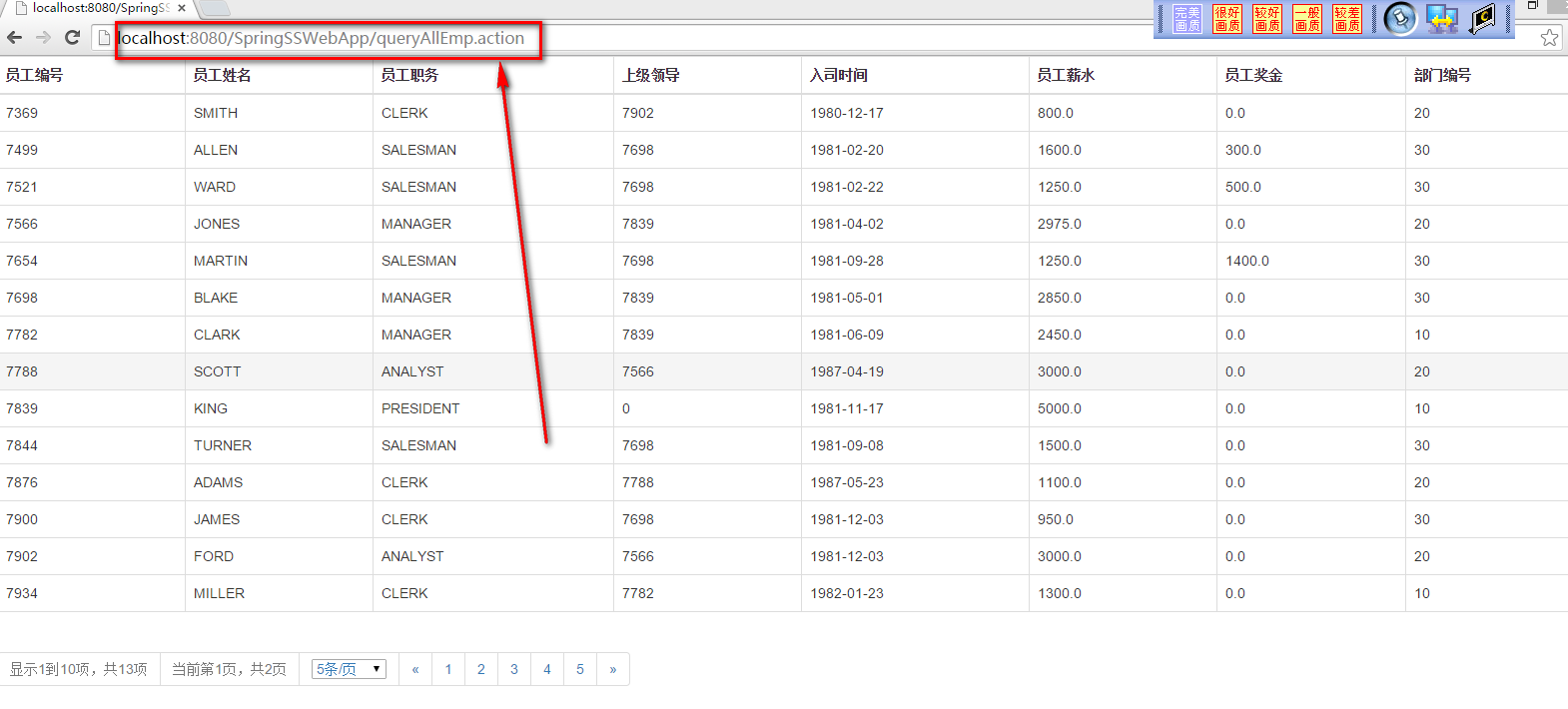
#### 4.配置springmvc的controller包扫描。



#### 5.编码Cotroller类测试



#### 6.测试结果：



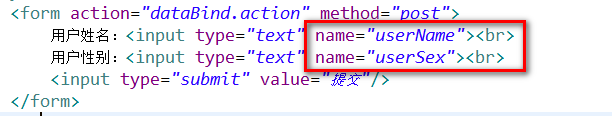
### 结论：springjdbc、spring、springmvc整合完毕！

# 6.SpringMvc参数绑定

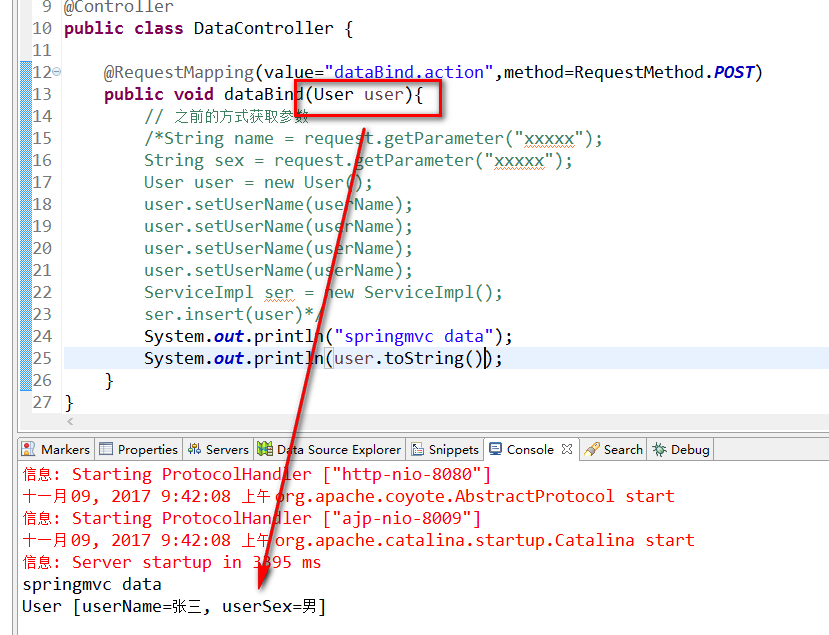
## 1.pojo类型：就是一个javabean，也就是model对象



前台界面：



Controller接收：

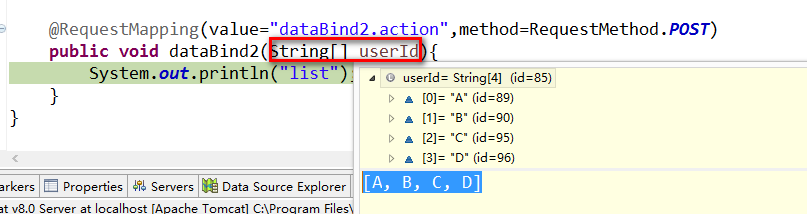


## 2.list类型参数绑定

前台界面：



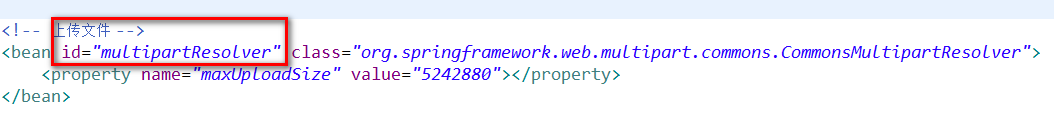
后台接收：



# 7. 数据回显

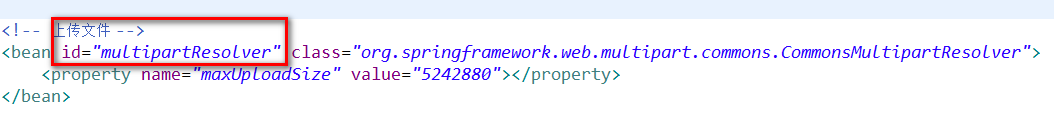
# 8.文件上传：上传图片为例。

## 1 在springmvc的配置文件中，加入文件上传解析类CommonsMultipartResolver



注意：此处记得加上id属性。

## 2.form表单enctype属性，只要是上传文件，就要将enctype属性配为”multipart/data-form”



## 3.别忘记文件上传的jar包

