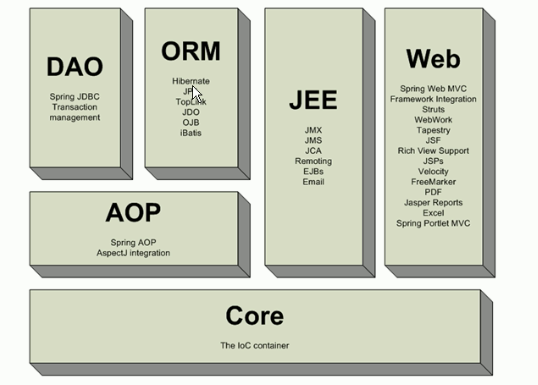
1. **1 Springmvc框架**

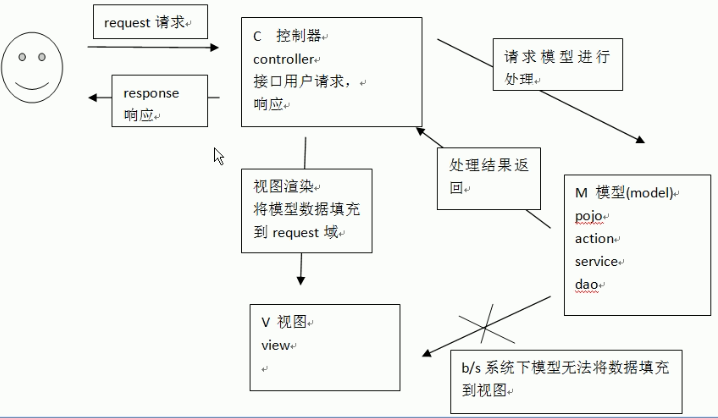




**1.2什么是mvc**

mvc是一种设计模式。

Mvc在**b/s**系统下的应用：eg:单例，多例模式



b/s系统下模型无法将数据填充到视图。

**1.3 SpringMVC框架**

Struts2中的filter就是前端控制器，action是后端控制器

第一步：发起请求到前端控制器{DispatcherServlet}}。

第二步：前端控制器请求HandlerMapping查找Handler.

可以根据xml配置，注解进行查找。

第三步：处理器映射器向前端控制器返回Handler.

第四步：前端控制器调用处理器映射器去执行Handler.

第五步：处理器适配器去执行Handler。

第六步：Handler执行完成给适配器返回ModleAndView.

第七步：处理器适配器向前端控制器返回ModleAndView

ModleAndView是springmvc框架的一个底层对象，包括Modle和view。

第八步：前端控制器请求视图解析器去进行视图解析。

根据逻辑视图名解析成真正的试图。

第九步：视图解析器向前端控制器返回View.

第十步：前端控制器进行视图渲染。

视图渲染将模型数据{在ModelAndView对象中}填充到request域。

第十一步：response前端控制器向用户响应结果。

重要组件：

1. 前端控制器：DispatcherServlet（不需要程序员开发）

作用：接受请求，响应结果，相当于转发器,中央处理器。

有了DispatcherServlet，减少了其他组件之间的耦合度。

1. 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。

1. 处理适配器HandlerAdapter。

作用：按照特定规则(HandlerAdapter要求的规则)去执行HandlerAdapter

注意：编写Handler时按照HandlerAdapter的要求去做，这样适配器才可以正常去执行Handler.

4.处理器Handler(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

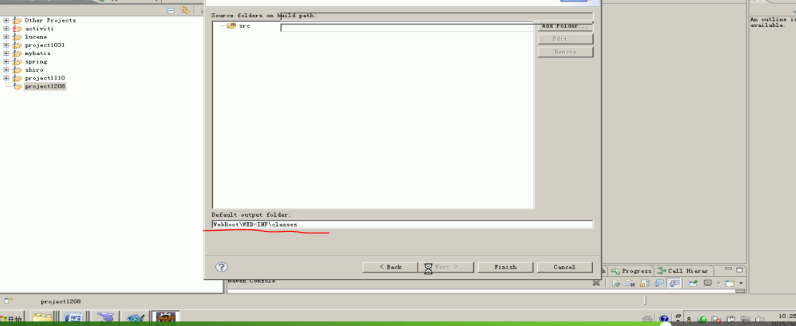
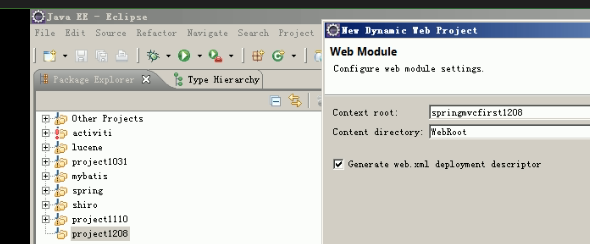
5.视图解析器View resolver（不需要程序员开发）

作用：进行视图解析，根据逻辑视图名解析真正的试图（view）。

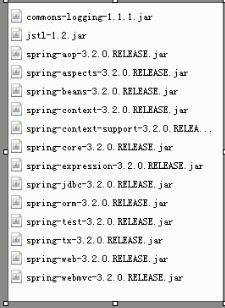
6.试图：View（需要程序员开发jsp）

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

**2.1环境配置**

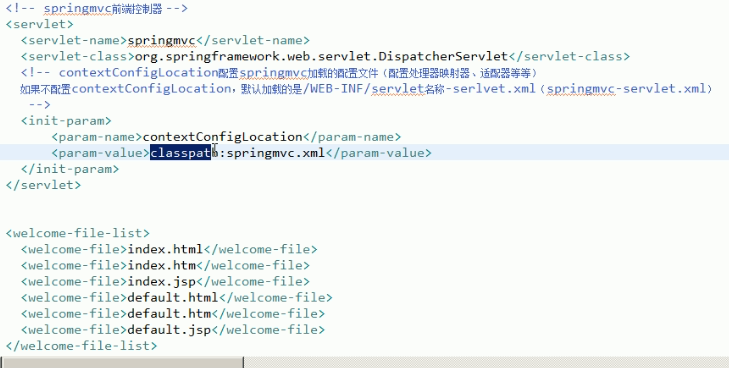
01 2

该处改为这样的话可以将eclipse中的程序导入到myeclipse中使用。



**2.2配置前端控制器**

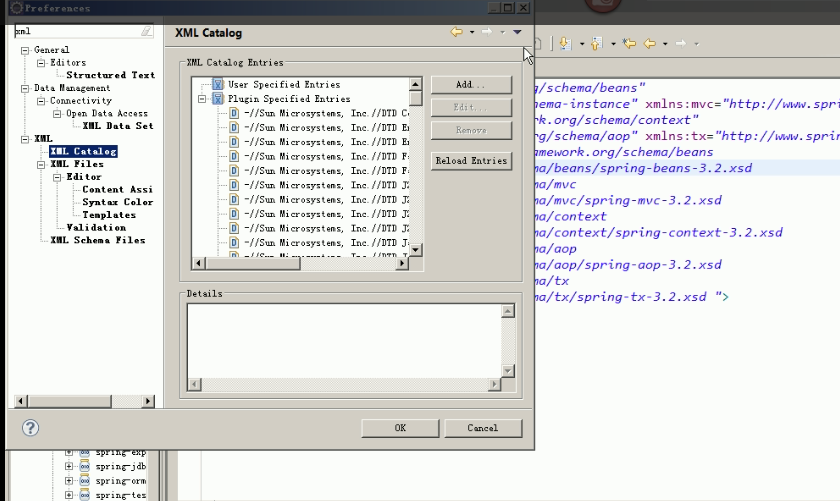
在web.xml中配置前端控制器





2.3配置处理器映射器

在classpath下的springmvc.xml中配置处理器映射器





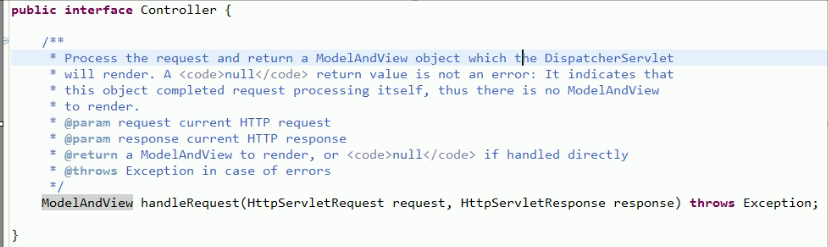




配置映射器时顺便配置好handler

2.4配置处理器适配器

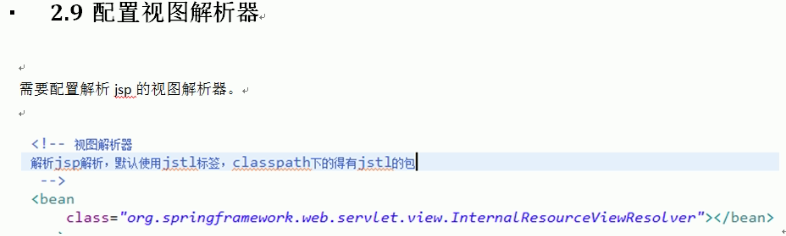




<!-- 处理器适配器 所有处理器适配器都实现 HandlerAdapter接口 -->

<bean class=*"org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"*/>

2.5配置视图解析器



<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

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-3.2.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.2.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-3.2.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-3.2.xsd "*>

<!-- 配置Handler -->

<bean id=*"itemsController1"* name=*"/queryItems\_test.action"* class=*"cn.itcast.ssm.controller.ItemsController1"* />

<!-- 配置另外一个Handler -->

<bean id=*"itemsController2"* class=*"cn.itcast.ssm.controller.ItemsController2"* />

<!-- 对于注解的Handler可以单个配置

实际开发中建议使用组件扫描

-->

<!-- <bean class="cn.itcast.ssm.controller.ItemsController3" /> -->

<!-- 可以扫描controller、service、...

这里让扫描controller，指定controller的包

-->

<context:component-scan base-package=*"cn.itcast.ssm.controller"*></context:component-scan>

<!-- 处理器映射器 将bean的name作为url进行查找 ，需要在配置Handler时指定beanname（就是url）

所有的映射器都实现 HandlerMapping接口。

-->

<bean

class=*"org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"* />

<!--简单url映射 -->

<bean class=*"org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"*>

<property name=*"mappings"*>

<props>

<!-- 对itemsController1进行url映射，url是/queryItems1.action -->

<prop key=*"/queryItems1.action"*>itemsController1</prop>

<prop key=*"/queryItems2.action"*>itemsController1</prop>

<prop key=*"/queryItems3.action"*>itemsController2</prop>

</props>

</property>

</bean>

<!--注解映射器 -->

<bean class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"*/>

<!--注解适配器 -->

<bean class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"*/>

<!-- 使用 mvc:annotation-driven代替上边注解映射器和注解适配器配置

mvc:annotation-driven默认加载很多的参数绑定方法，

比如json转换解析器就默认加载了，如果使用mvc:annotation-driven不用配置上边的RequestMappingHandlerMapping和RequestMappingHandlerAdapter

实际开发时使用mvc:annotation-driven

-->

<!-- <mvc:annotation-driven></mvc:annotation-driven> -->

<!-- 处理器适配器 所有处理器适配器都实现 HandlerAdapter接口 -->

<bean

class=*"org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"* />

<!-- 另一个非注解的适配器 -->

<bean class=*"org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter"*/>

<!-- 视图解析器

解析jsp解析，默认使用jstl标签，classpath下的得有jstl的包

-->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<!-- 配置jsp路径的前缀 -->

<property name=*"prefix"* value=*"/WEB-INF/jsp/"*/>

<!-- 配置jsp路径的后缀 -->

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

</bean>

</beans>

**常见的两种404错误**

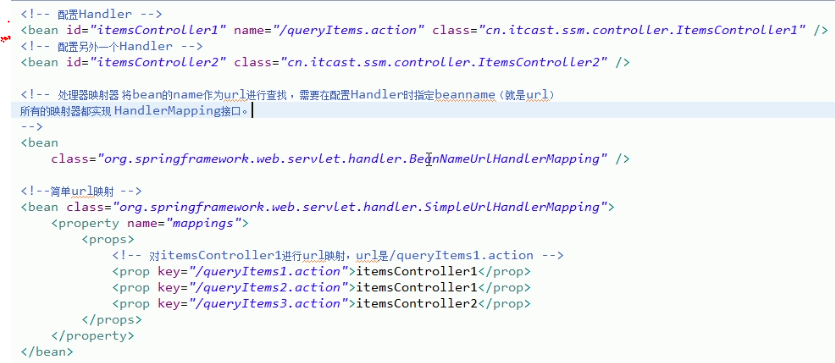
1

**3非注解的处理器与映射器**

3.1映射器

所有的映射器都实现 HandlerMapping接口。





多个映射器可以并存,前端控制器判断url能让那些映射器映射，就让正确的映射器处理。

3.2适配器





映射器就是根据url来找handler

适配器就是按照适配器所要求的规则去执行handler.

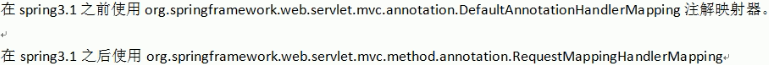
## DispatcherServlet.properties

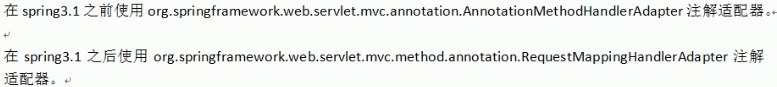
## 

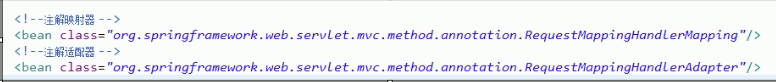
前端控制器从上边的文件中加载处理器映射器、适配器、视图解析器等组件。

如果不在springmvc中配置，就默认加载了。

**4.注解的处理器与映射器**







<!--注解映射器 -->

<bean class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"*/>

<!--注解适配器 -->

<bean class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"*/>

<!-- 使用 mvc:annotation-driven代替上边注解映射器和注解适配器配置

mvc:annotation-driven默认加载很多的参数绑定方法，

比如json转换解析器就默认加载了，如果使用mvc:annotation-driven不用配置上边的RequestMappingHandlerMapping和RequestMappingHandlerAdapter

实际开发时使用mvc:annotation-driven

-->

<!-- <mvc:annotation-driven></mvc:annotation-driven> -->

## 开发注解handler

