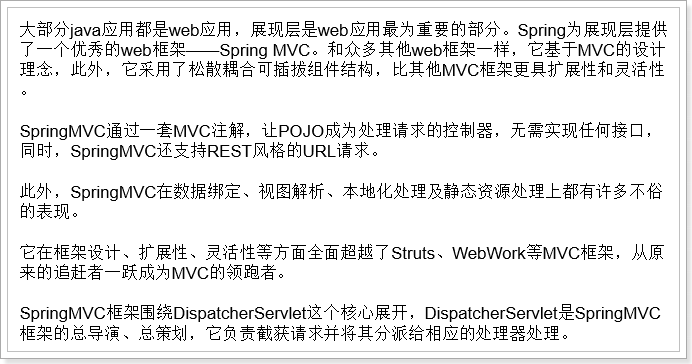
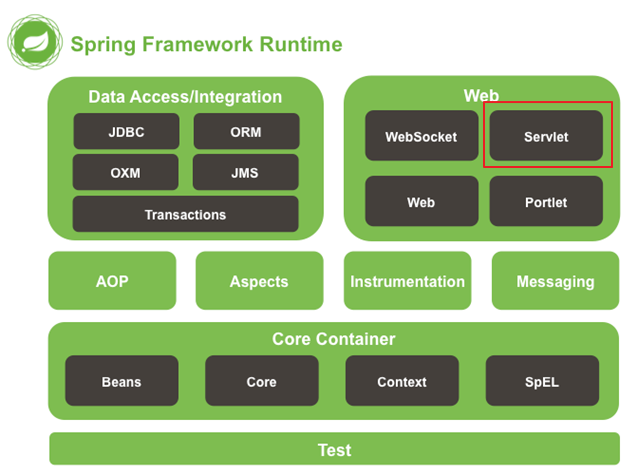
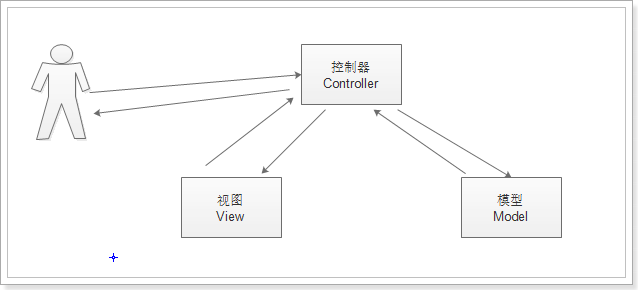
# SpringMVC简介



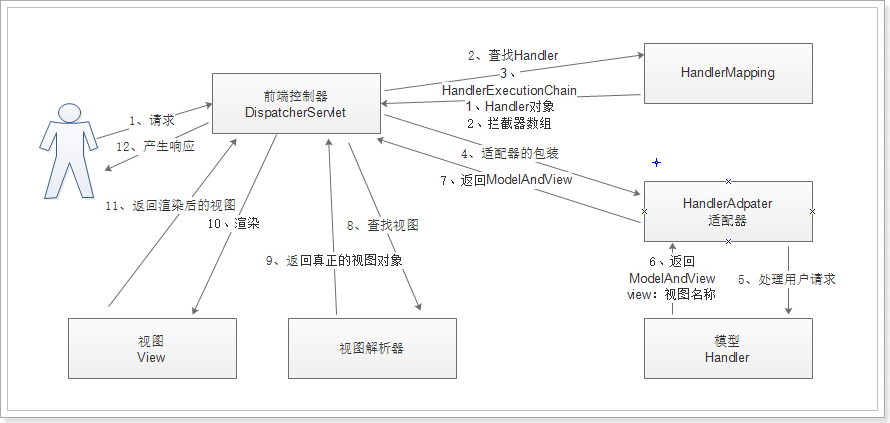
## Spring体系简介



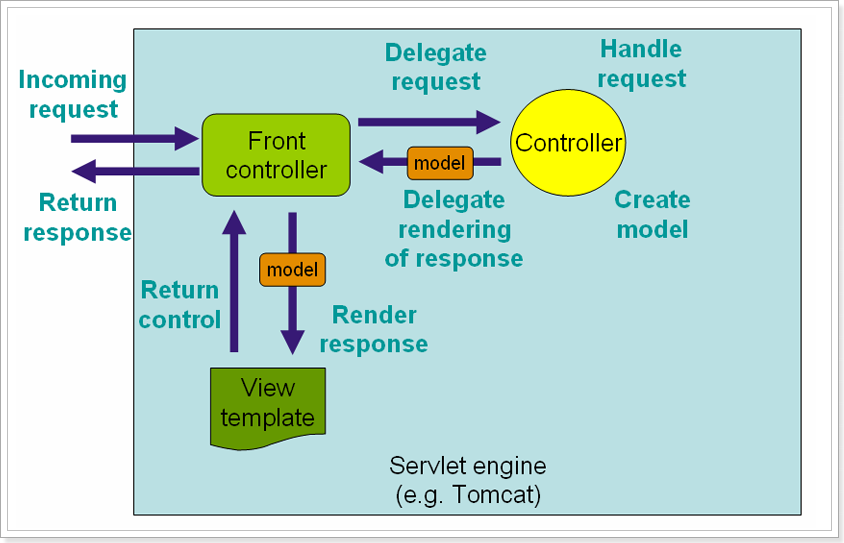
# 回顾MVC设计模式



# SpringMVC的架构

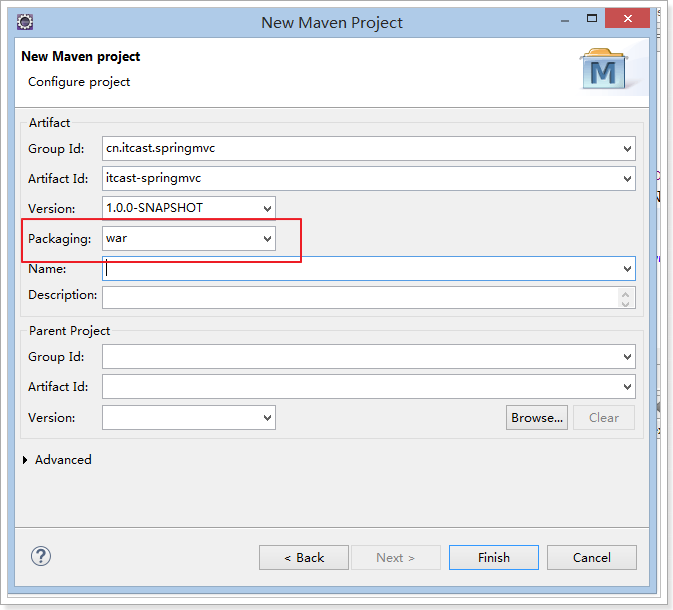


1. 用户发出请求到SpringMVC的前端控制器
2. 通过HandlerMapping查找Handler
3. 返回执行链
   1. Handler对象
   2. 拦截器数组（list）
4. 通过适配器对Handler对象进行包装，调用Handler
5. 处理用户的请求
6. 返回ModelAndView对象
   1. Model数据
   2. 视图名称，不是真正的视图对象
7. 将ModelAndView对象返回到前端控制器
8. 通过视图名称在视图解析器中查找视图对象
9. 返回真正的视图对象
10. 前端控制器拥有了模型数据和视图对象，进行视图渲染
11. 返回渲染后的视图
12. 给用户产生响应



# 第一个SpringMVC的程序

## 创建工程



## 添加依赖

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>cn.itcast.parent</groupId>

<artifactId>itcast-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>cn.itcast.springmvc</groupId>

<artifactId>itcast-springmvc</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</dependency>

<!-- JSP相关 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<port>80</port>

<!--

http://127.0.0.1/

-->

<path>/</path>

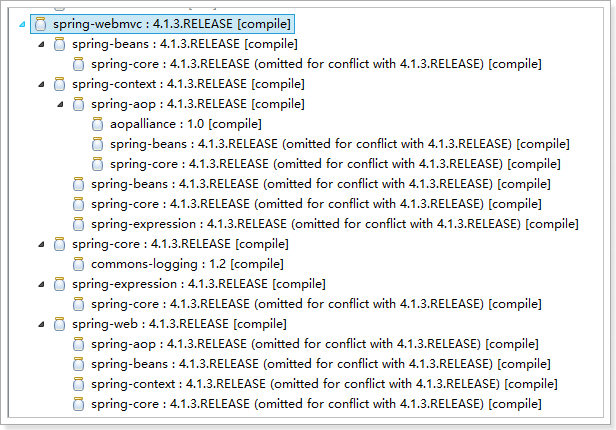
</configuration>

</plugin>

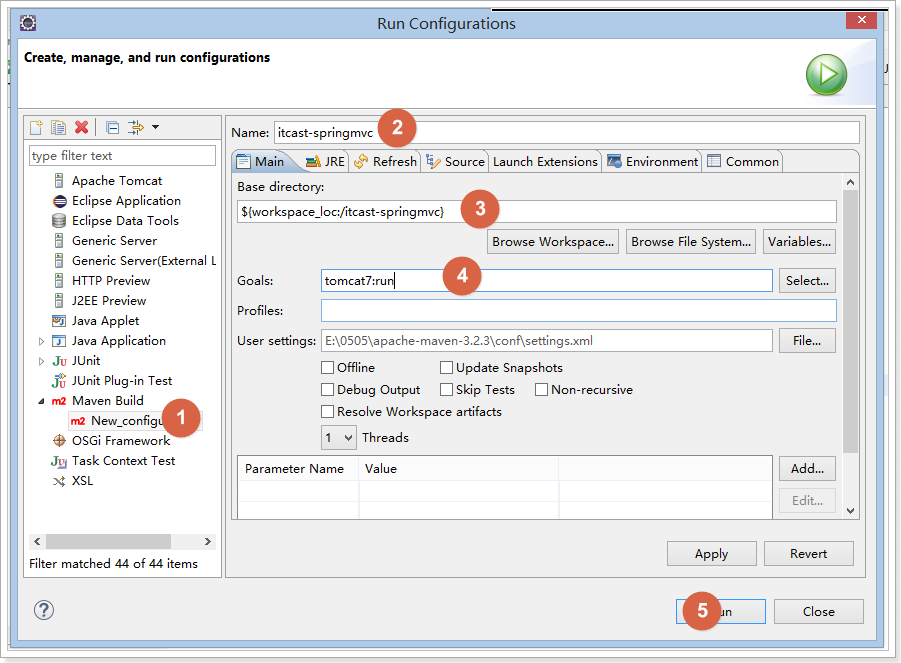
</plugins>

</build>

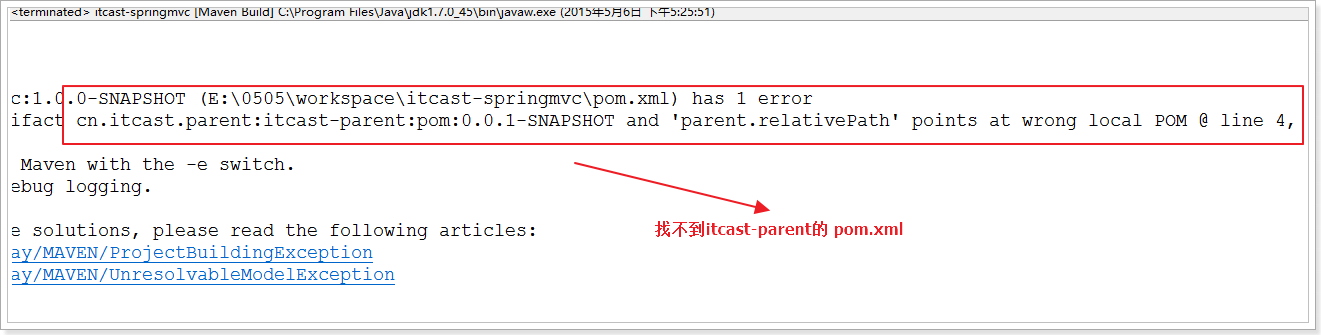
SpringMVC的依赖关系：



## 启动tomcat插件



错误：



找不到：是指在本地仓库中找不到。

解决方案：

将itcast-parent安装到本地仓库中，run as 🡺 Maven install

## 创建web.xml

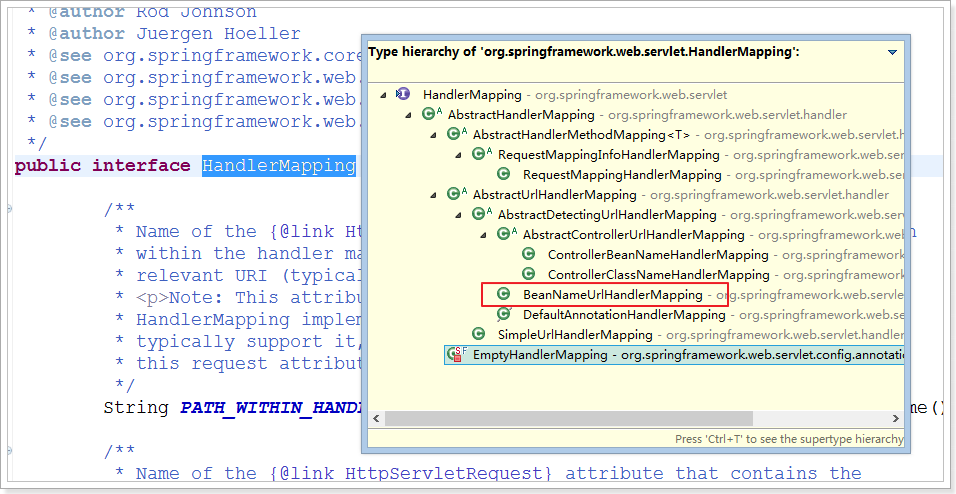
配置SpringMVC的入口：



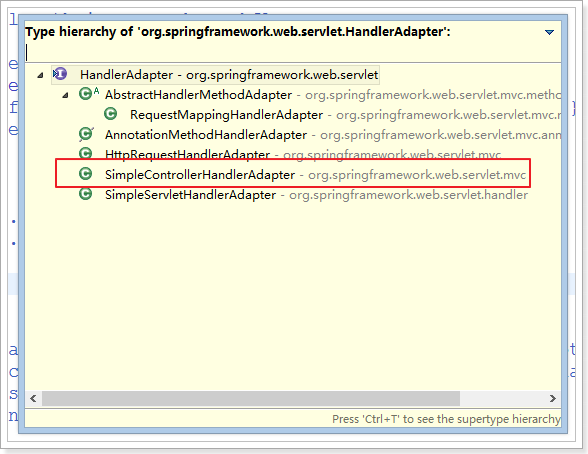
SpringMVC的读取默认的配置文件的路径：/WEB-INF/{servlet-name}-servlet.xml

## 配置SpringMVC文件

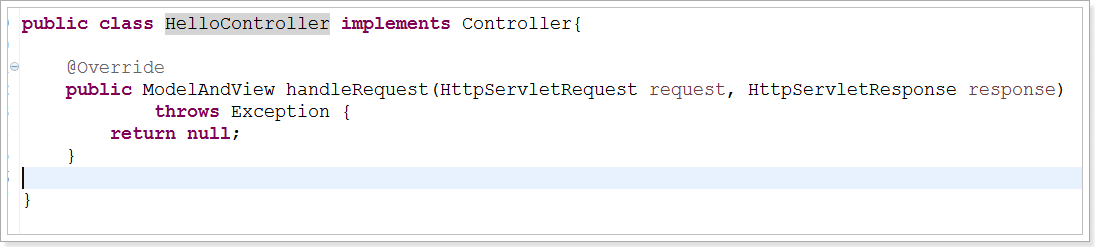
### HandlerMapping



### HandlerAdapter

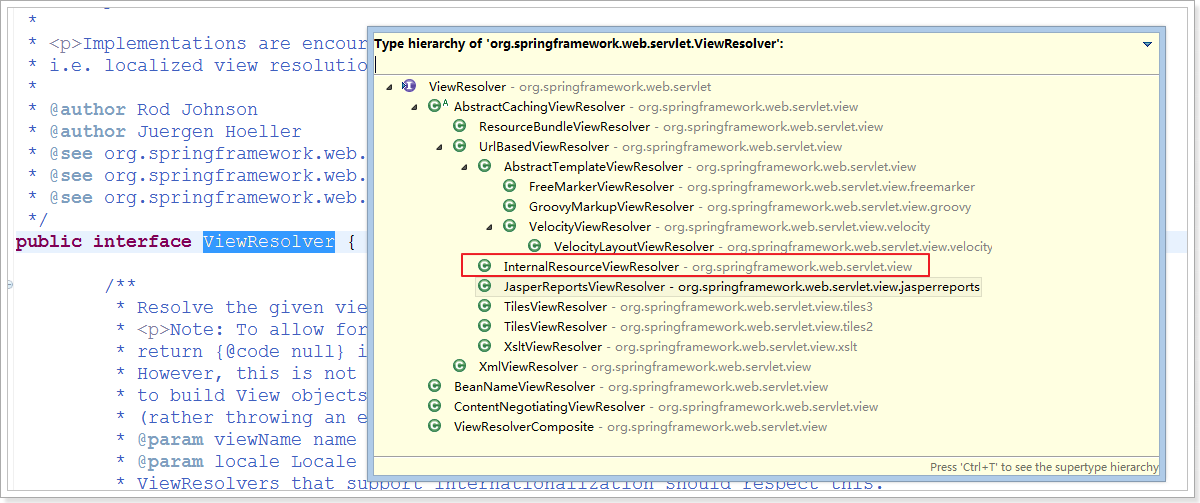


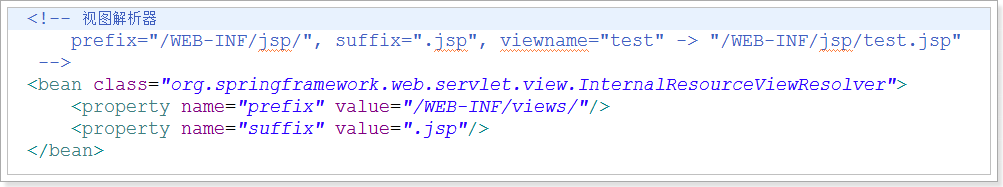
### 定义处理器



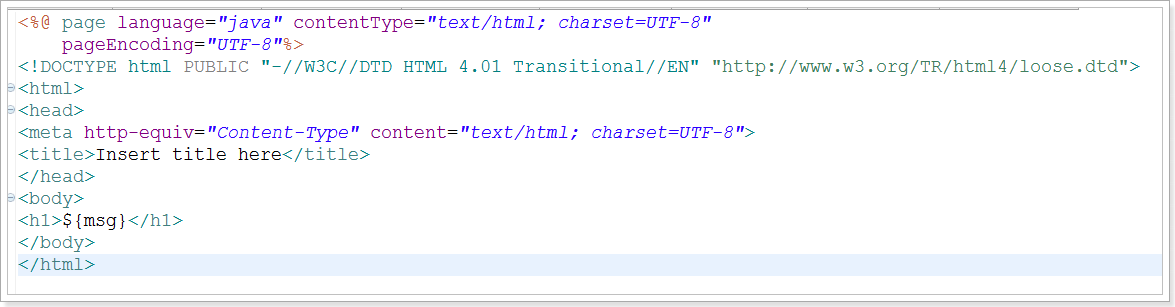


### ViewResolver



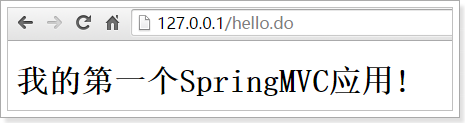


### 创建hello.jsp

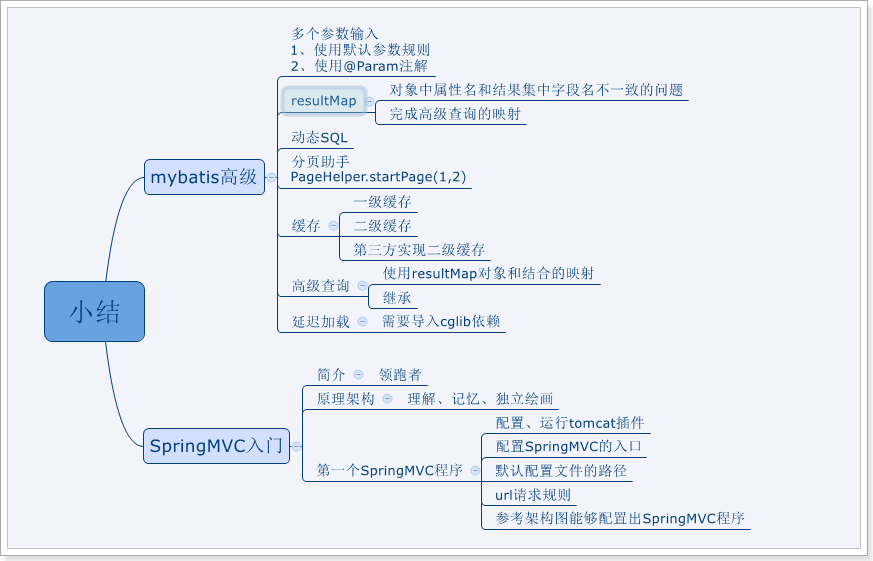


## 测试

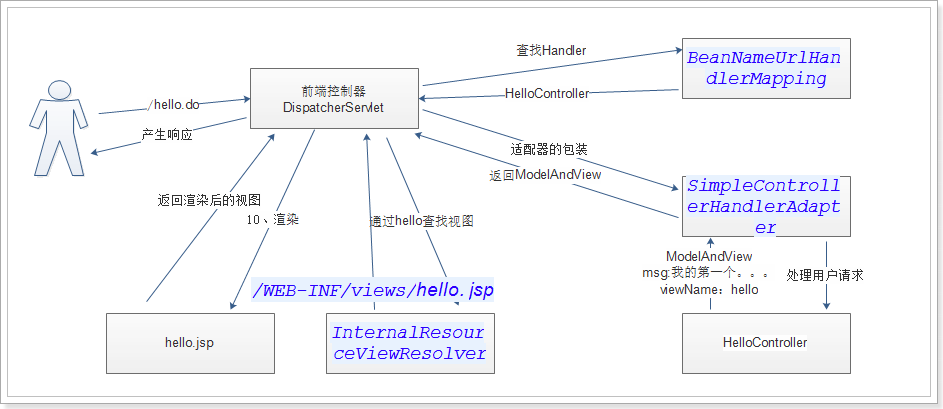
<http://127.0.0.1/hello.do>



# 小结



# 第一个SpringMVC应用的流程

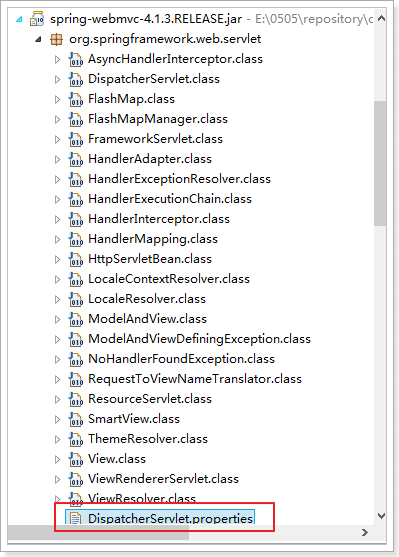


# 优化SpringMVC的配置

处理器映射器和适配器是可以不配置的。



问题：这是为什么？ 一定存在默认配置。



org.springframework.web.servlet.LocaleResolver=org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver

org.springframework.web.servlet.ThemeResolver=org.springframework.web.servlet.theme.FixedThemeResolver

org.springframework.web.servlet.HandlerMapping=org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping,\

org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping

org.springframework.web.servlet.HandlerAdapter=org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter,\

org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter,\

org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter

org.springframework.web.servlet.HandlerExceptionResolver=org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerExceptionResolver,\

org.springframework.web.servlet.mvc.annotation.ResponseStatusExceptionResolver,\

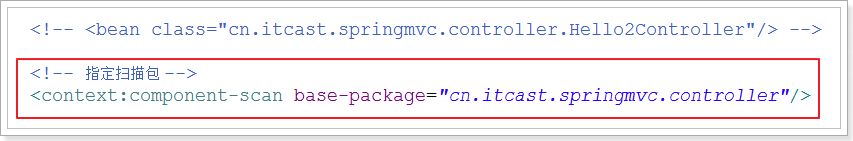
org.springframework.web.servlet.mvc.support.DefaultHandlerExceptionResolver

org.springframework.web.servlet.RequestToViewNameTranslator=org.springframework.web.servlet.view.DefaultRequestToViewNameTranslator

org.springframework.web.servlet.ViewResolver=org.springframework.web.servlet.view.InternalResourceViewResolver

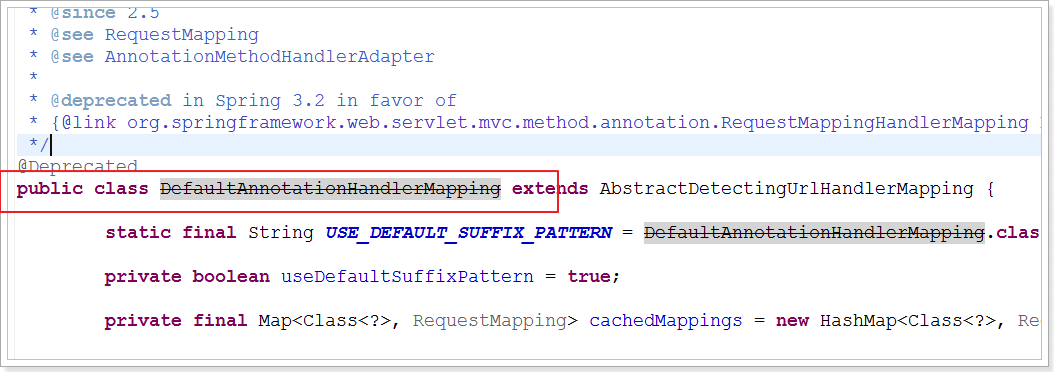
org.springframework.web.servlet.FlashMapManager=org.springframework.web.servlet.support.SessionFlashMapManager

# 第一个注解程序

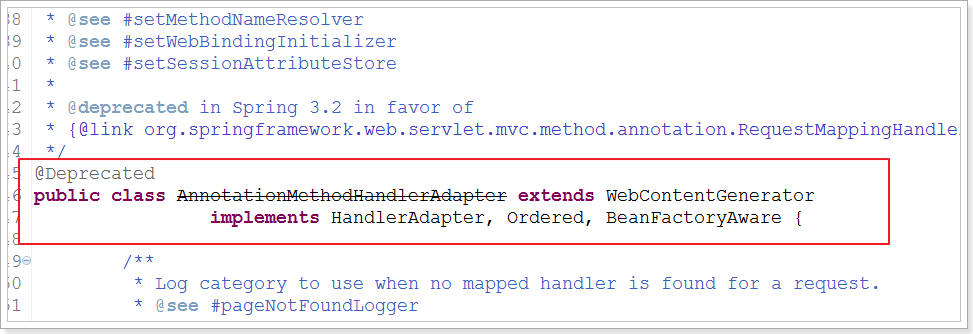
1. 创建Hello2Controller  
   
2. 将Controller加入到SpringMVC的容器中  
   
3. 测试  
   

## 发现默认配置的问题

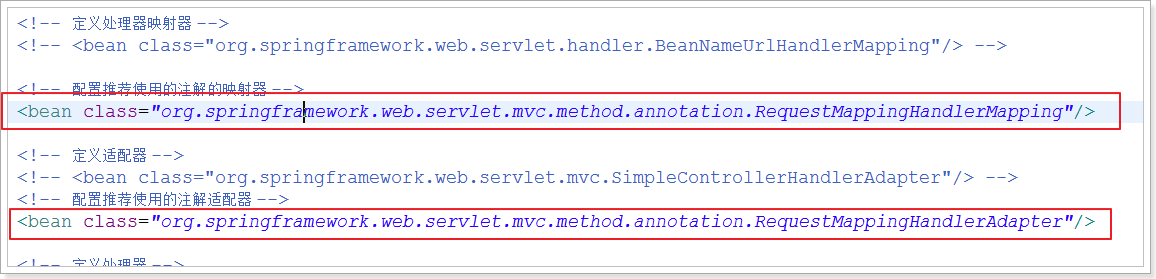
默认注解映射器以及过期不推荐使用：



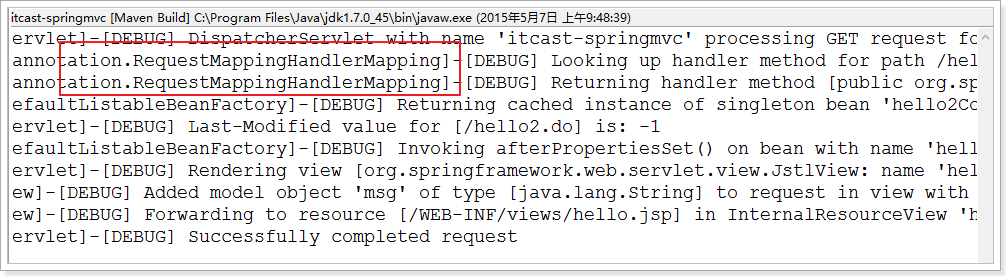
默认的适配器也过期了：



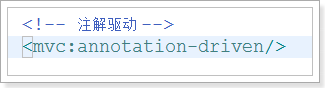
推荐的配置：



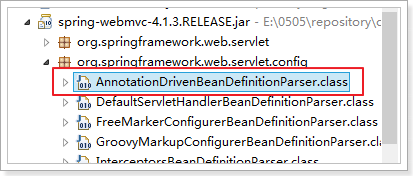
测试：

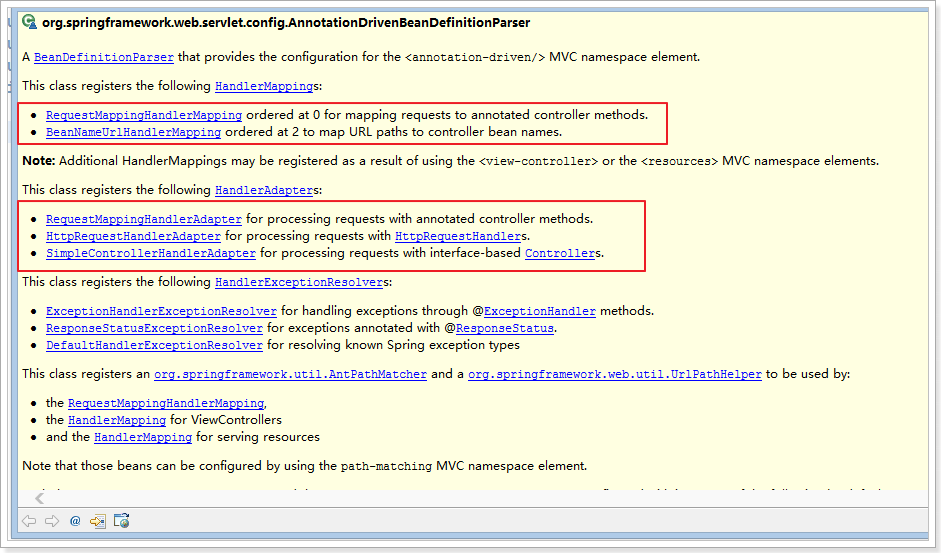


## 配置注解驱动使用推荐的注解配置



问题：为什么使用注解驱动就可以使用推荐的配置？





注解驱动是默认配置的升级版！

# @RequestMapping





请求地址：<http://127.0.0.1/test/show.do>

Value的两种写法：



## @RequestMapping映射

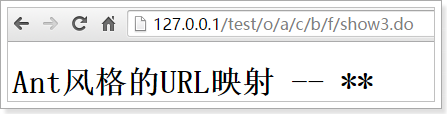


### 标准URL映射

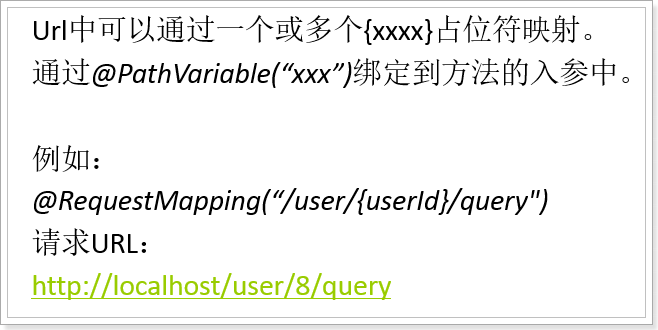


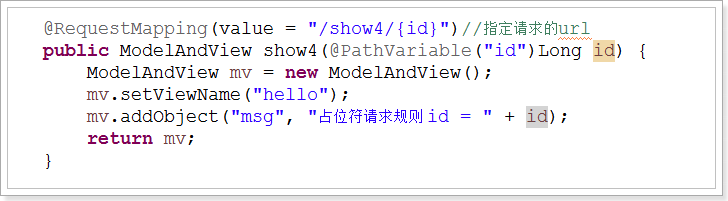
### Ant风格的URL映射



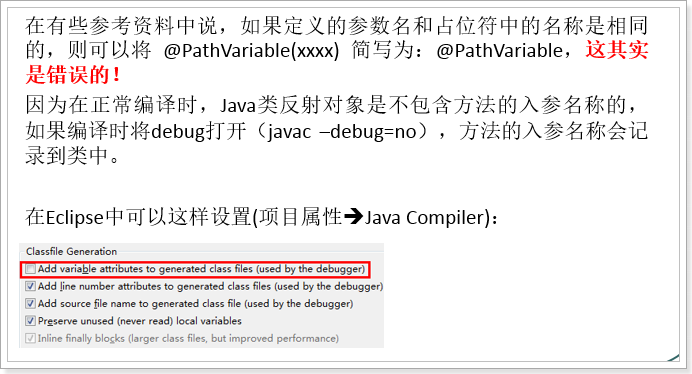


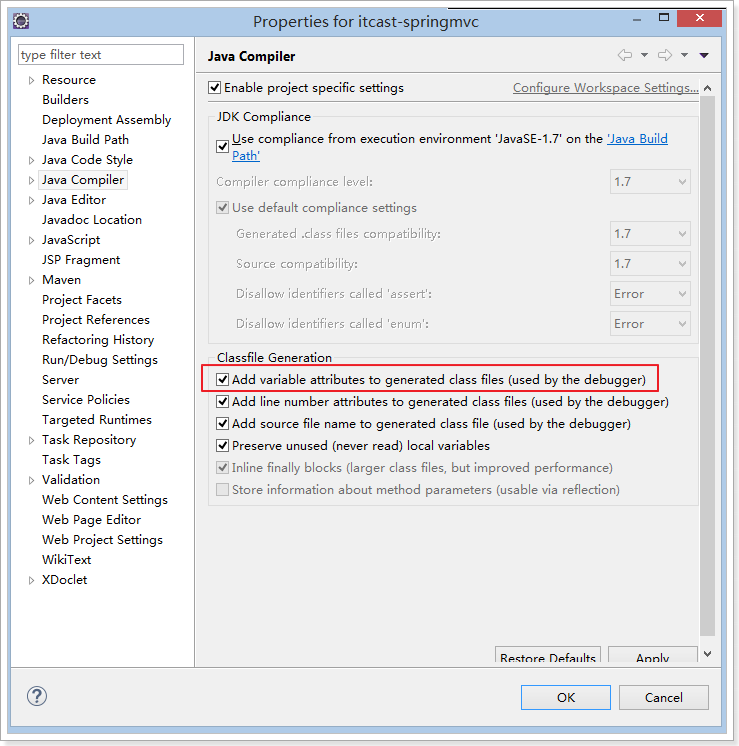
### 占位符映射





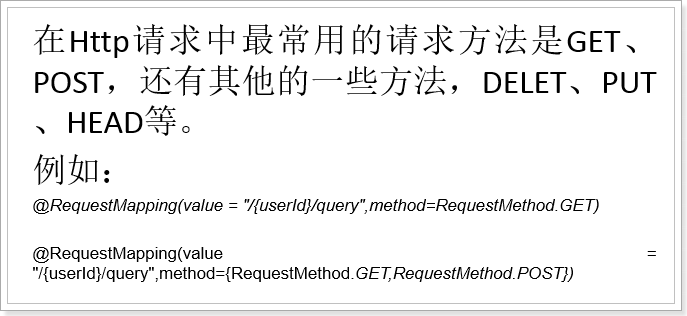
### @PathVariable小误区



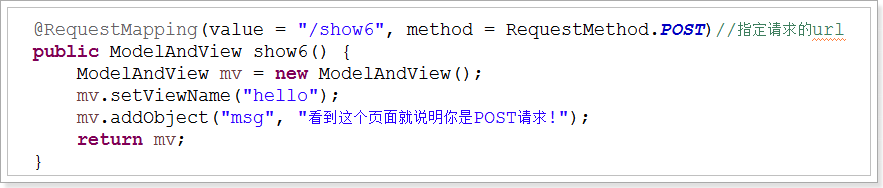


结论：无论参数名和占位符名是否一致，都必须执行参数名。

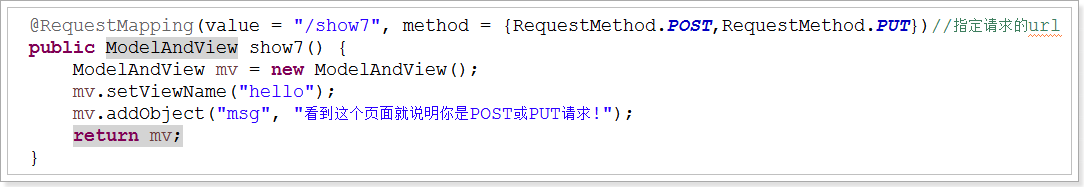
### 限制请求方法



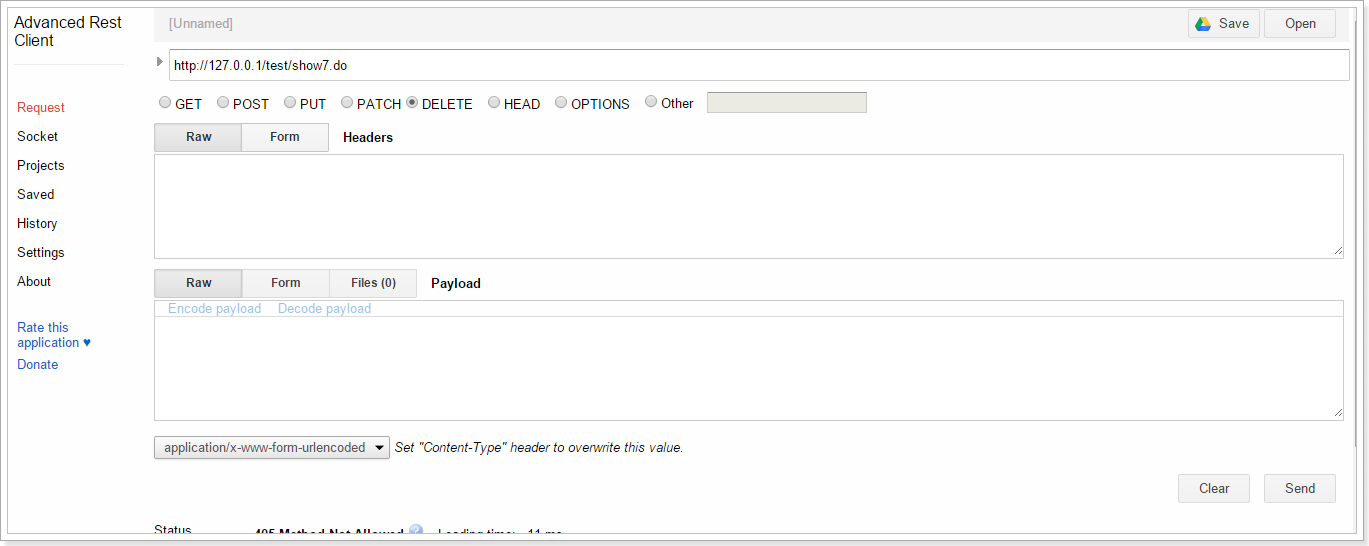
限制POST请求：



限制POST或PUT请求：



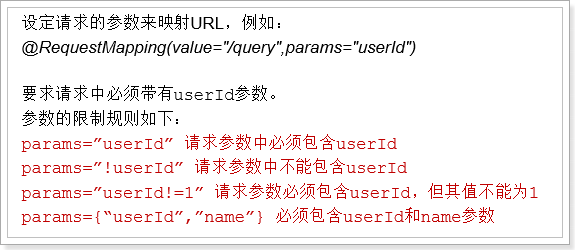
测试：

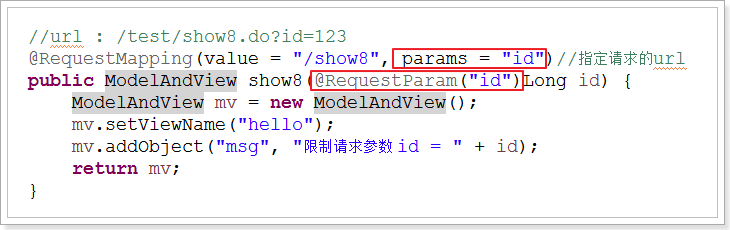


405状态：

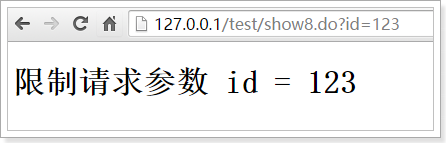


### 限定参数映射





测试：



# 处理方法的数据绑定

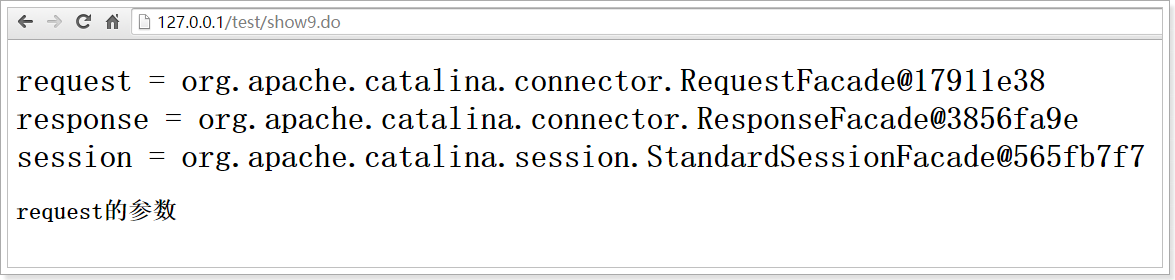
数据绑定： SpringMVC将接受到数据绑定到入参的参数中。

## 绑定Servlet内置对象

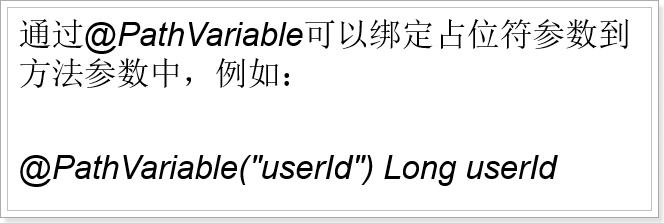




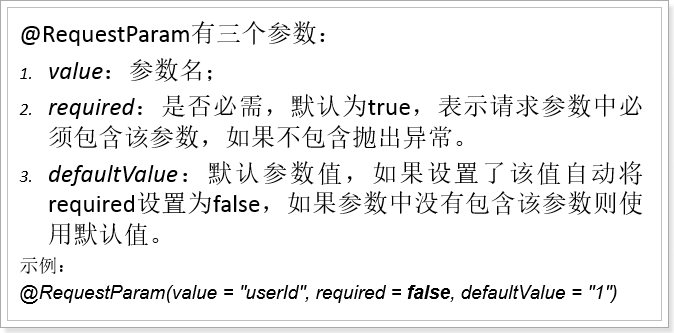
测试：

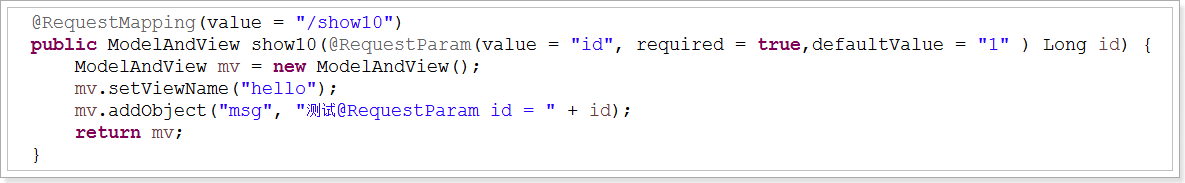


## @PathVariable



## @RequestParam

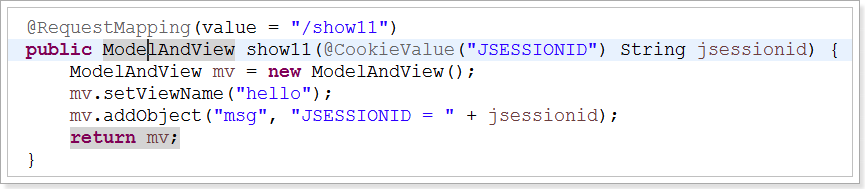


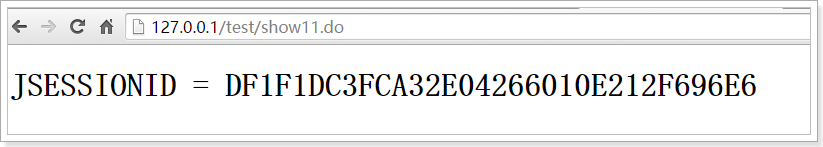




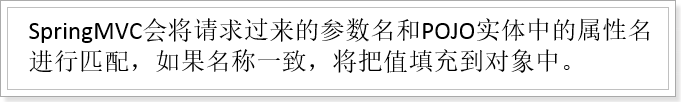
## @ CookieValue

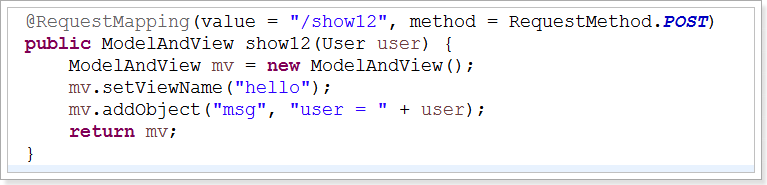


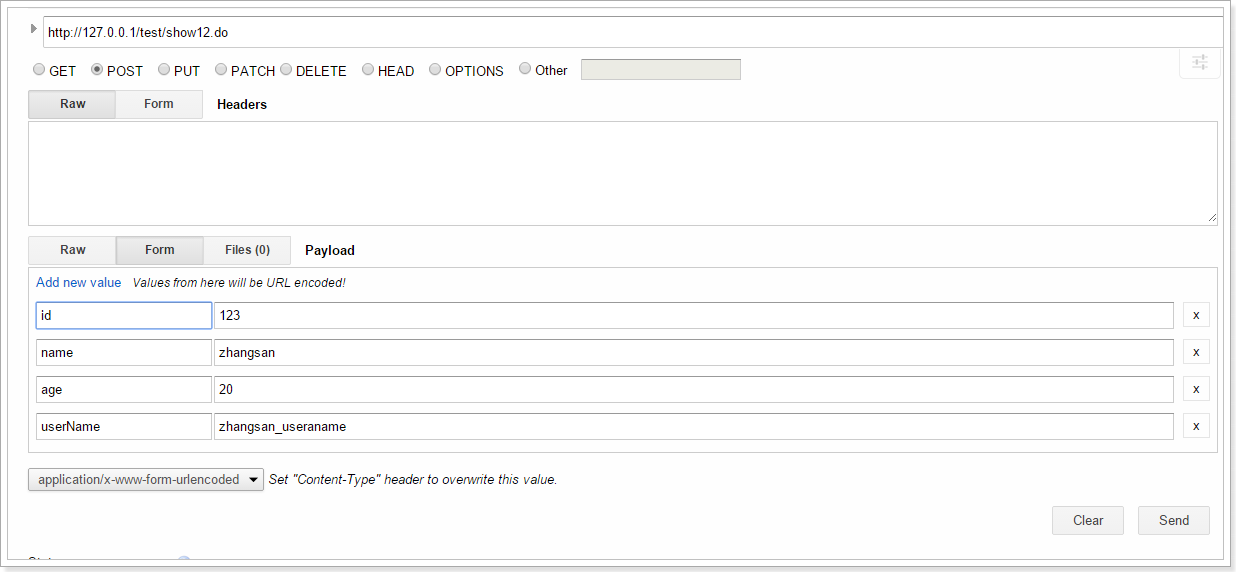




## POJO对象绑定参数

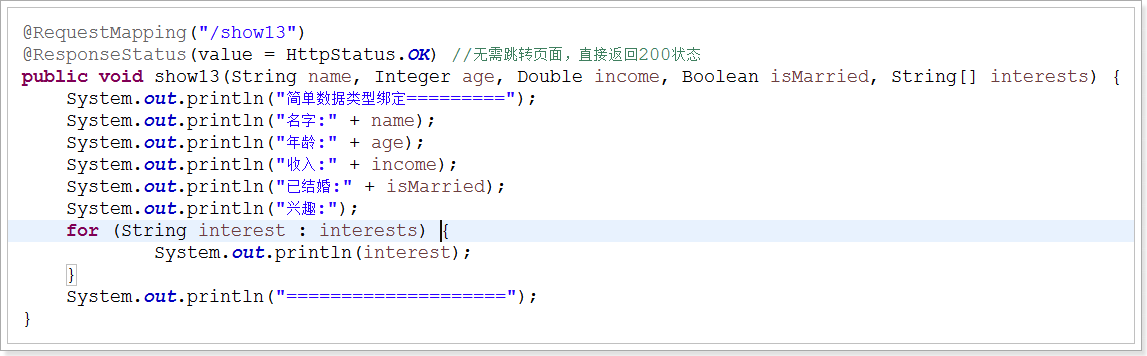






## 基本数据类型绑定

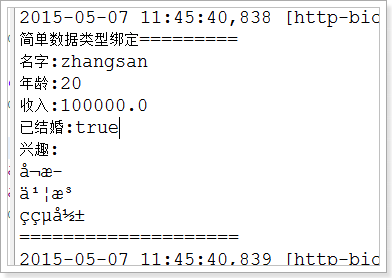






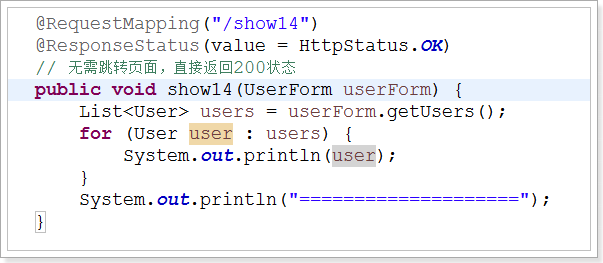
测试：

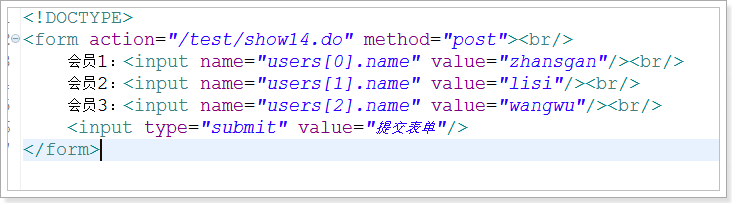




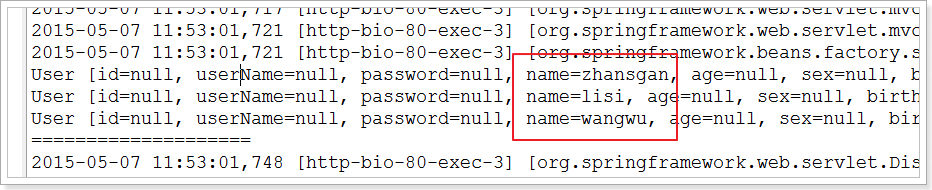
## 集合List绑定



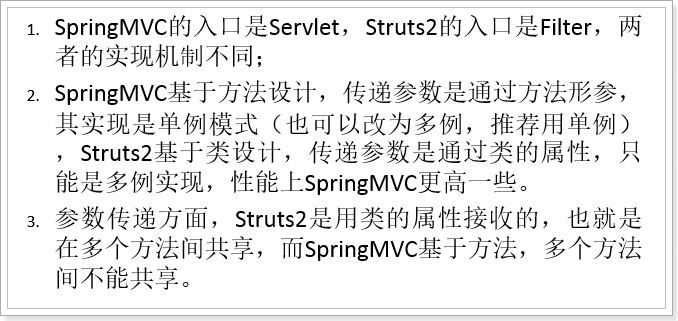




测试：

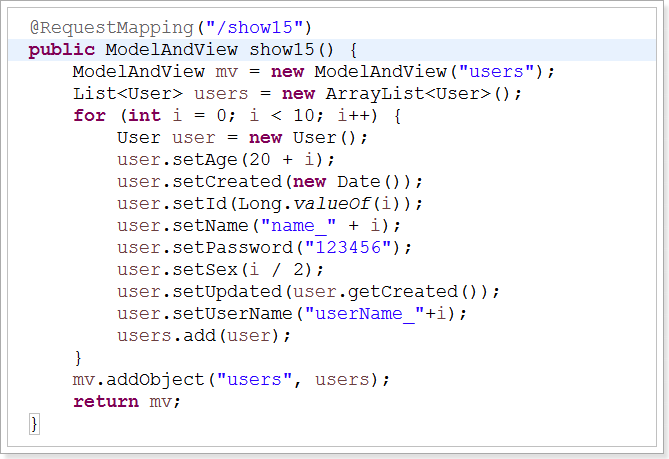


# SpringMVC与Struts2的区别

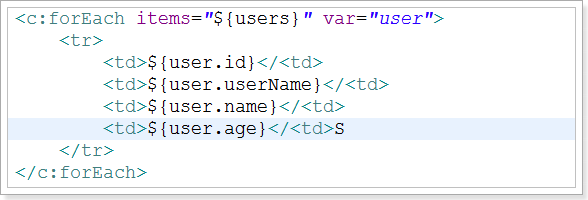


# JSP和JSTL视图解析器

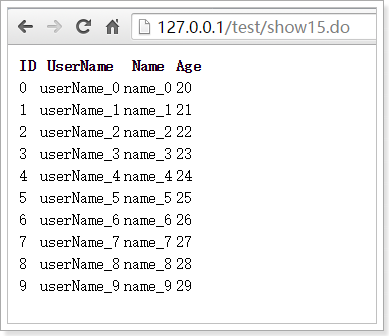








测试：



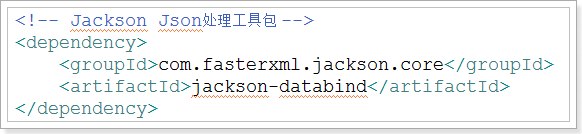
# 使用@ResponseBody输出JSON



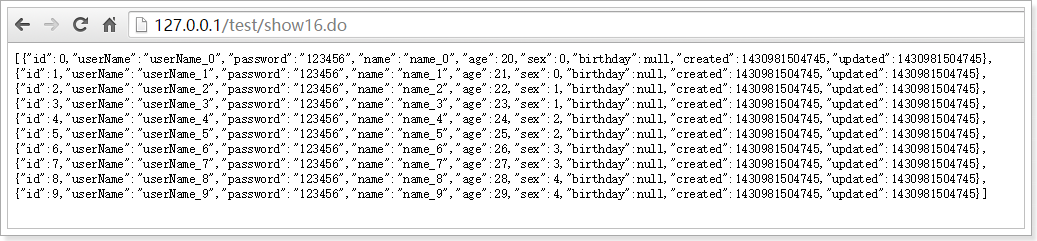
实现：



要有Jackson的支持：

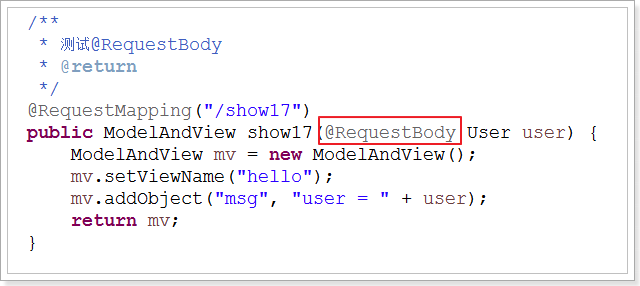


测试：

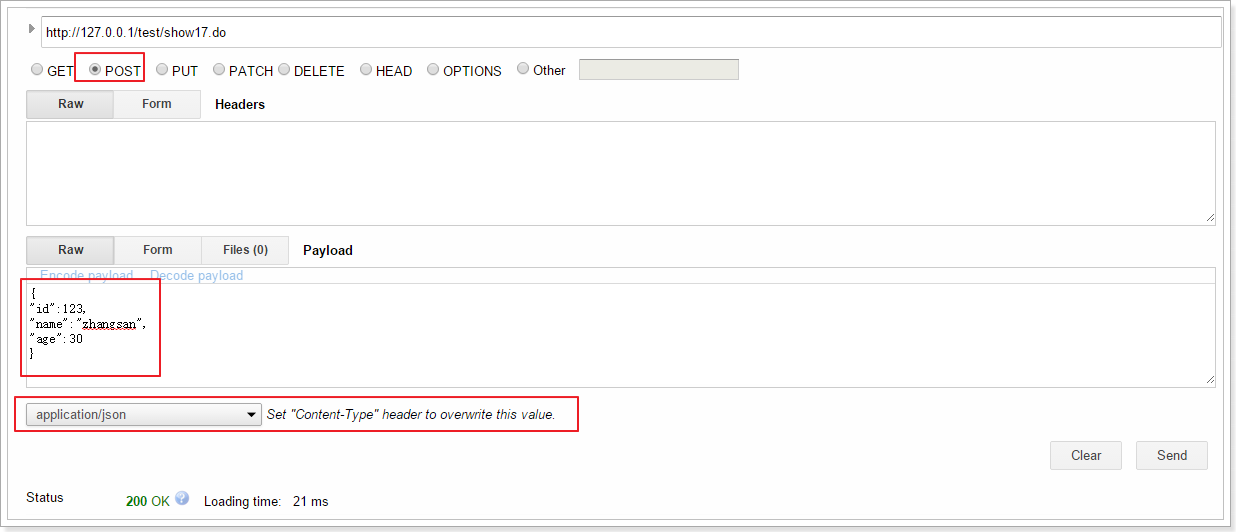


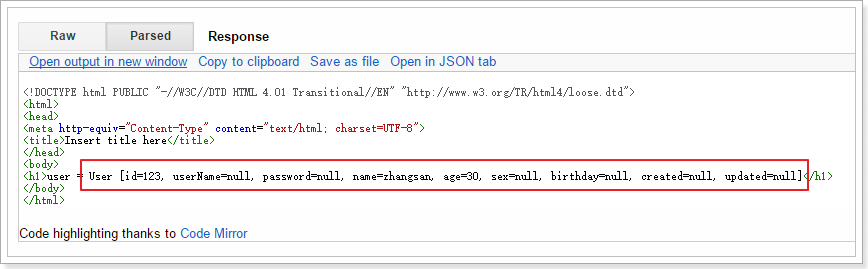
## @RequestBody

作用：将提交的json数据转换为java对象。



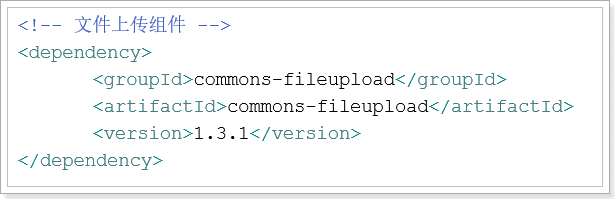
测试：



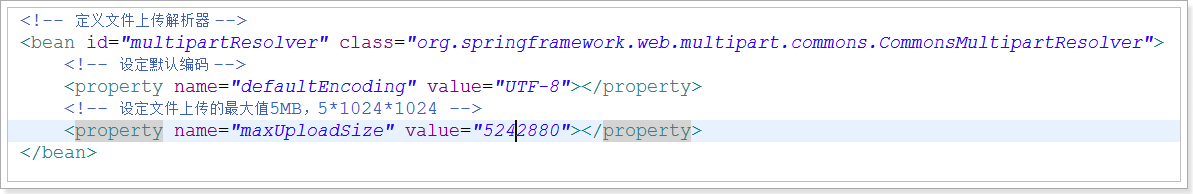


# 文件上传

## 加入上传组件依赖



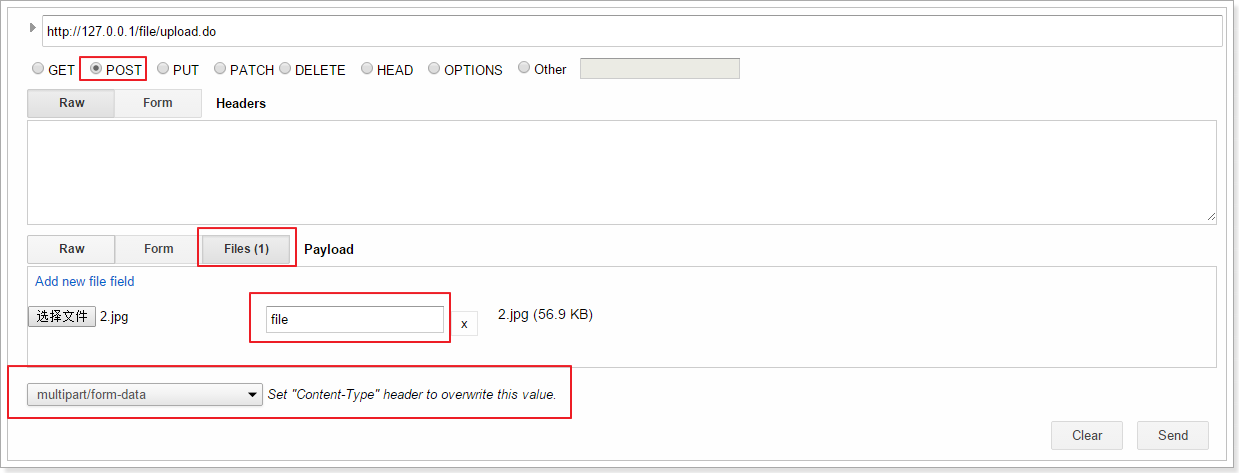
## 定义文件上传解析器

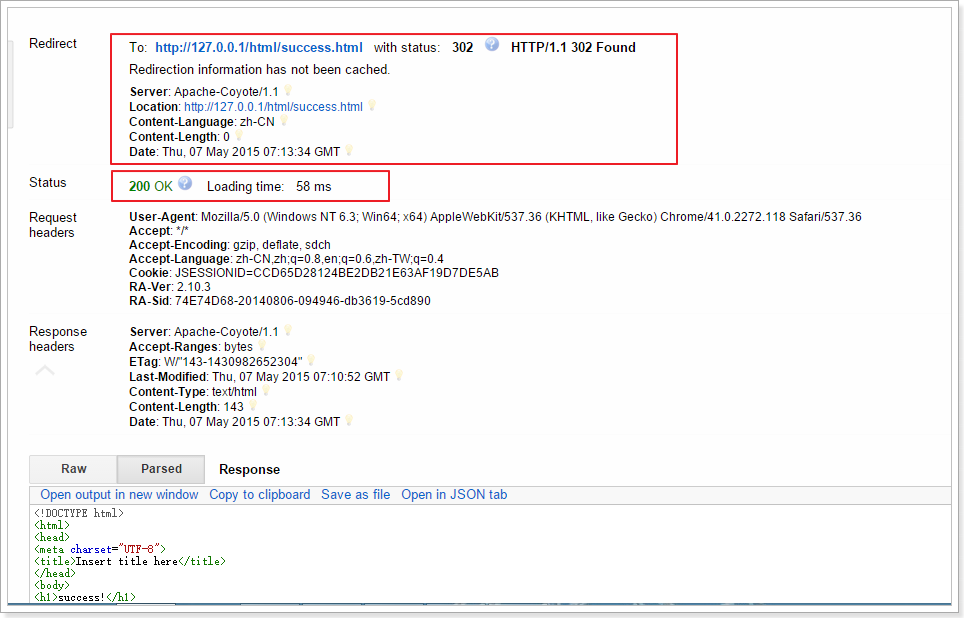


## 编写Controller



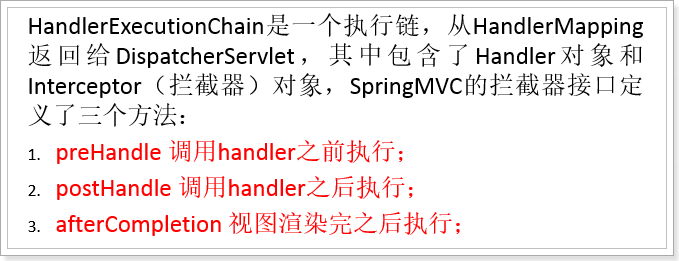
## 测试



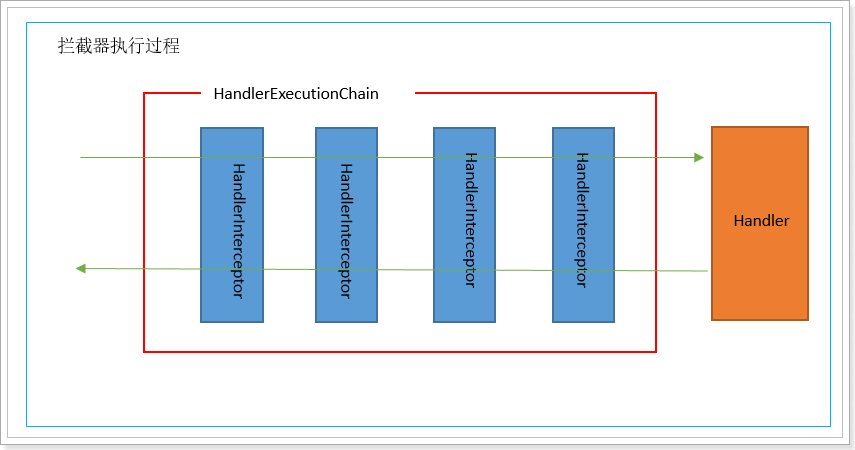


# 拦截器

## 理解HandlerExecutionChain



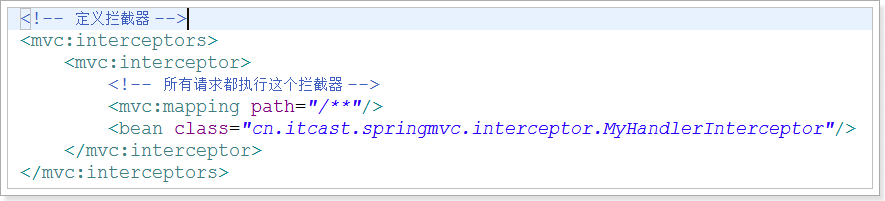
## 拦截的执行过程



## 实现自定义拦截器

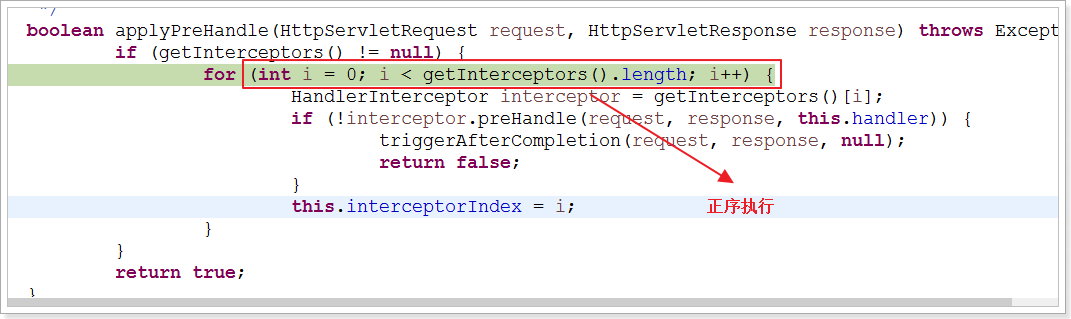


在SpringMVC中定义拦截器：

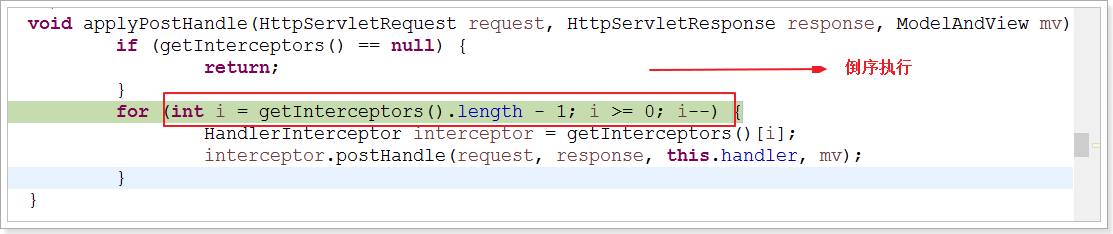


## SpringMVC的执行过程

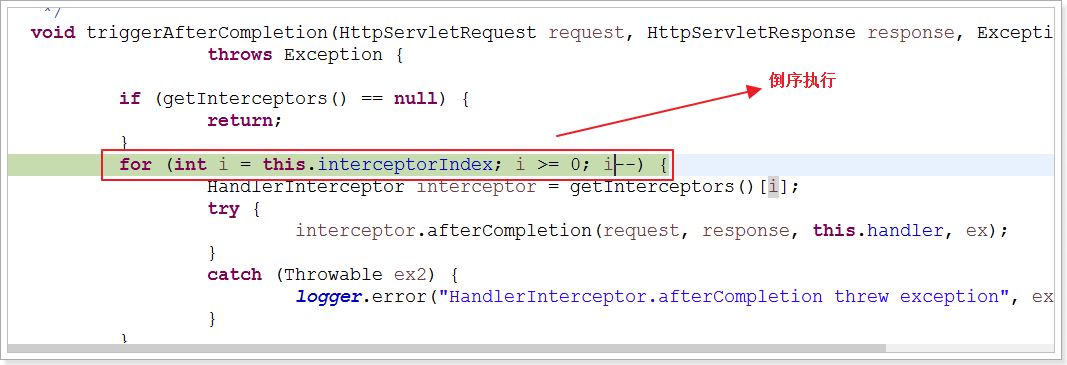
前置方法是正序执行的：



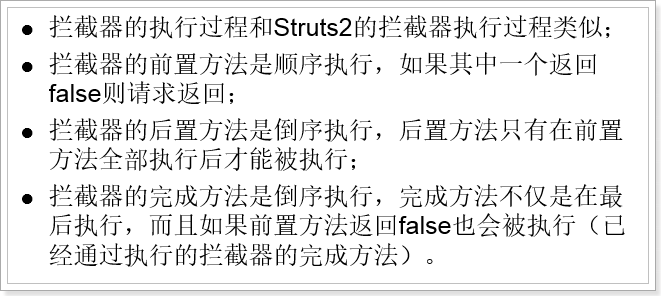
后置方法倒序执行：



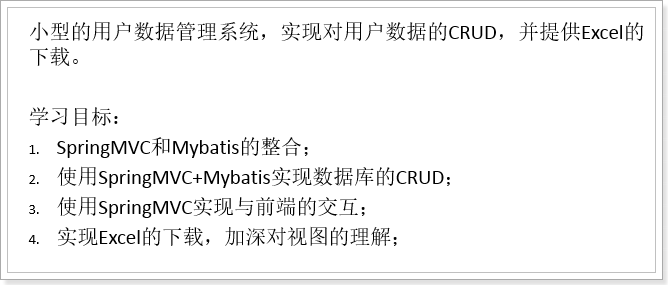
完成方法是倒序执行：



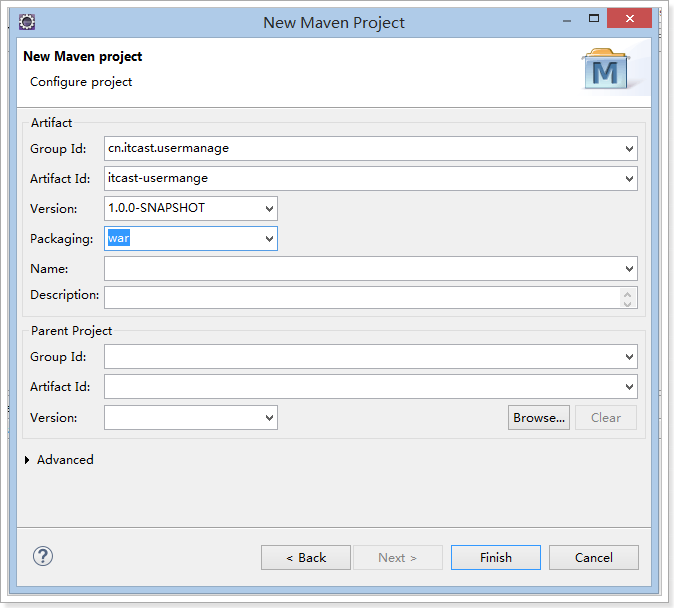
## 拦截器总结



# SpringMVC综合应用



## 创建工程



## 导入依赖

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>cn.itcast.parent</groupId>

<artifactId>itcast-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>cn.itcast.usermanage</groupId>

<artifactId>itcast-usermange</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

</dependency>

<!-- Mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

</dependency>

<!-- MySql -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</dependency>

<!-- Jackson Json处理工具包 -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

<dependency>

<groupId>com.jolbox</groupId>

<artifactId>bonecp-spring</artifactId>

<version>0.8.0.RELEASE</version>

</dependency>

<!-- Apache工具组件 -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-io</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<port>80</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

## 创建web.xml

<display-name>itcast-springmvc-usermanage</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/applicationContext\*.xml</param-value>

</context-param>

<!--Spring的ApplicationContext 载入 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 编码过滤器，以UTF8编码 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置SpringMVC -->

<servlet>

<servlet-name>usermanage</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!--

自定义SpringMVC的配置文件

-->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/springmvc-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>usermanage</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

## 添加配置文件 jdbc、log4j

## 添加Spring配置

Spring容器配主要配置数据源和扫描器。

<!-- 使用spring自带的占位符替换功能 -->

<bean

class=*"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"*>

<!-- 允许JVM参数覆盖 -->

<property name=*"systemPropertiesModeName"* value=*"SYSTEM\_PROPERTIES\_MODE\_OVERRIDE"* />

<!-- 忽略没有找到的资源文件 -->

<property name=*"ignoreResourceNotFound"* value=*"true"* />

<!-- 配置资源文件 -->

<property name=*"locations"*>

<list>

<value>classpath:jdbc.properties</value>

</list>

</property>

</bean>

<!-- 配置扫描包 -->

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

<bean id=*"dataSource"* class=*"com.jolbox.bonecp.BoneCPDataSource"*

destroy-method=*"close"*>

<!-- 数据库驱动 -->

<property name=*"driverClass"* value=*"${jdbc.driver}"* />

<!-- 相应驱动的jdbcUrl -->

<property name=*"jdbcUrl"* value=*"${jdbc.url}"* />

<!-- 数据库的用户名 -->

<property name=*"username"* value=*"${jdbc.username}"* />

<!-- 数据库的密码 -->

<property name=*"password"* value=*"${jdbc.password}"* />

<!-- 检查数据库连接池中空闲连接的间隔时间，单位是分，默认值：240，如果要取消则设置为0 -->

<property name=*"idleConnectionTestPeriod"* value=*"60"* />

<!-- 连接池中未使用的链接最大存活时间，单位是分，默认值：60，如果要永远存活设置为0 -->

<property name=*"idleMaxAge"* value=*"30"* />

<!-- 每个分区最大的连接数 -->

<property name=*"maxConnectionsPerPartition"* value=*"150"* />

<!-- 每个分区最小的连接数 -->

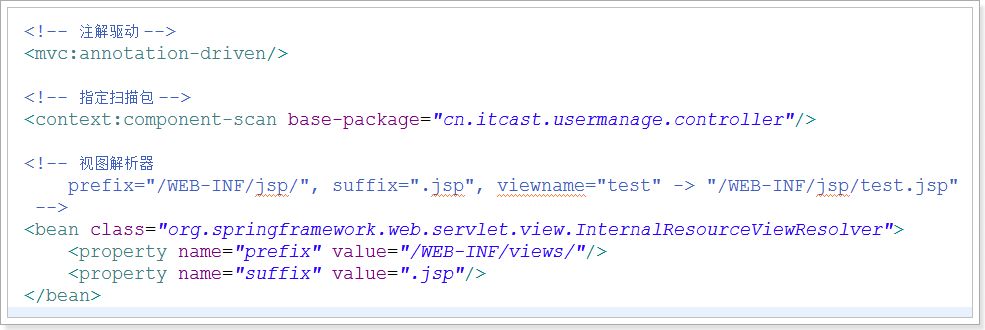
<property name=*"minConnectionsPerPartition"* value=*"5"* />

</bean>

## 配置mybatis



## 配置SpringMVC

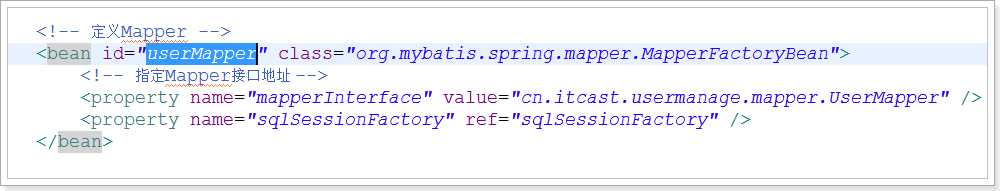


## Mybatis和Spring的整合

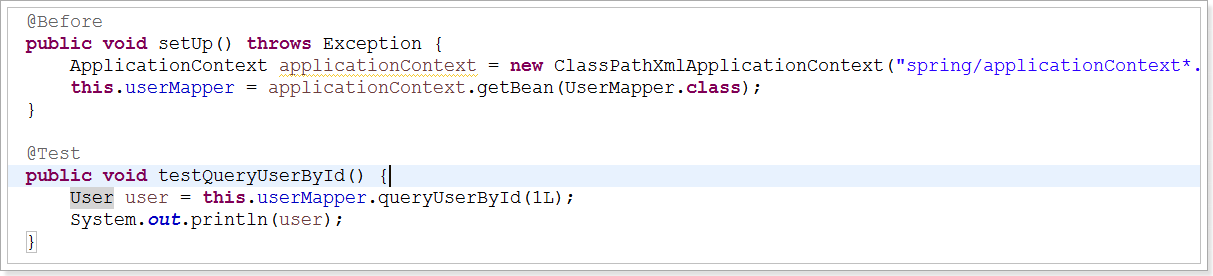
### 配置SqlSessionFactory



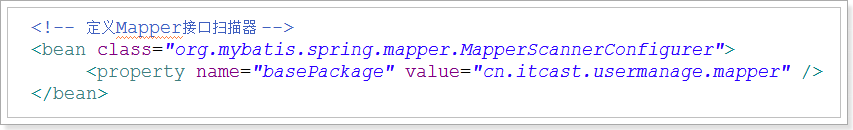
### 在Spring容器中配置Mapper接口



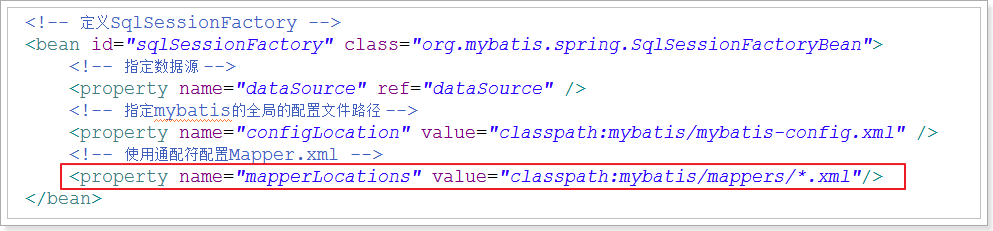
直接可以在Spring容器中获取Mapper的动态代理实现类。



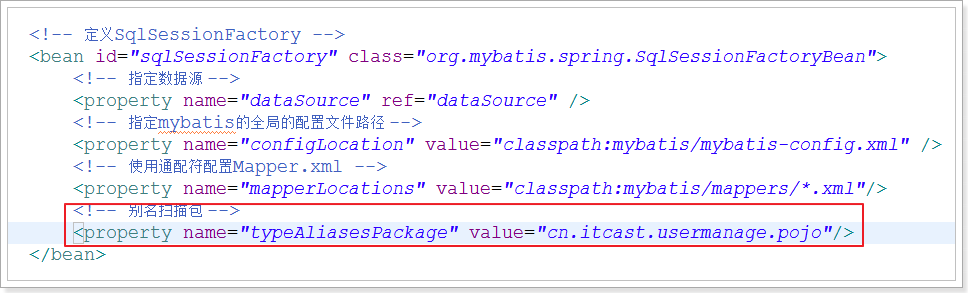
### 配置Mapper接口扫描器



### 通配符配置Mapper.xml



### 配置别名扫描包

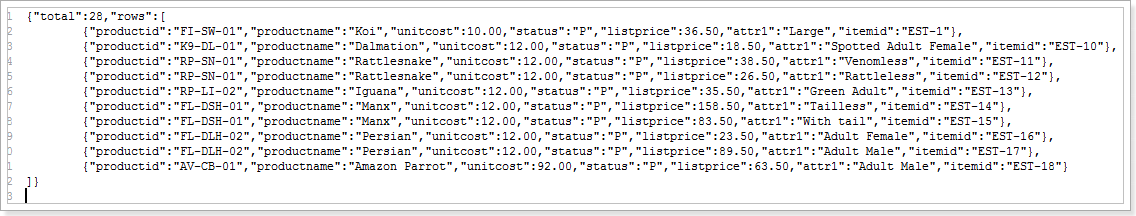


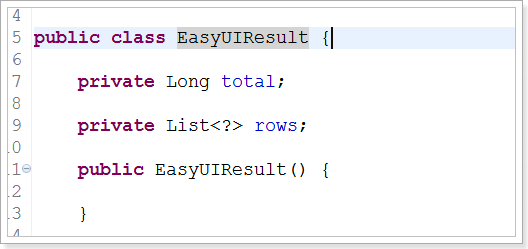
## 解决静态资源404问题



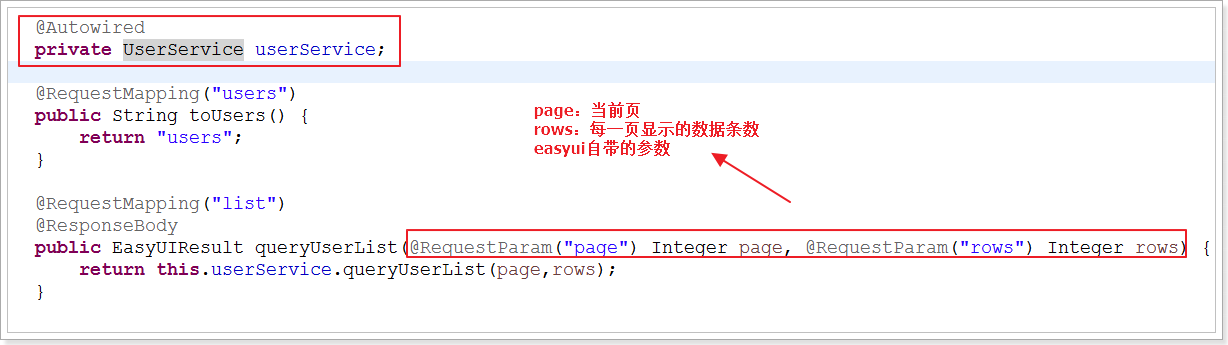
## 实现用户列表查询

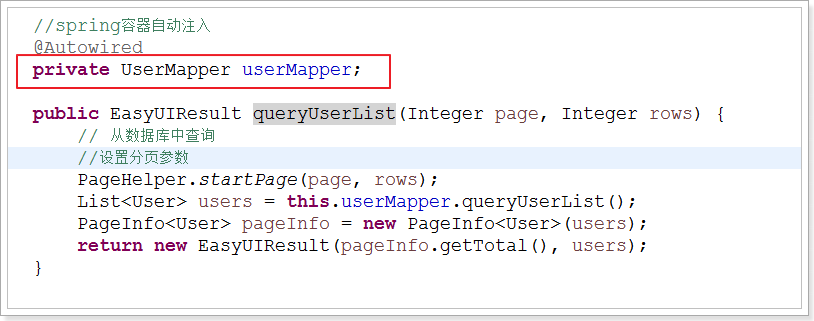
### EasyUI结构

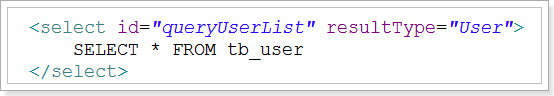




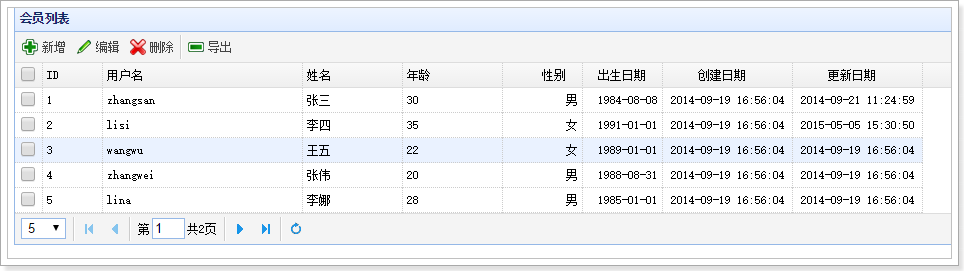
### 查询







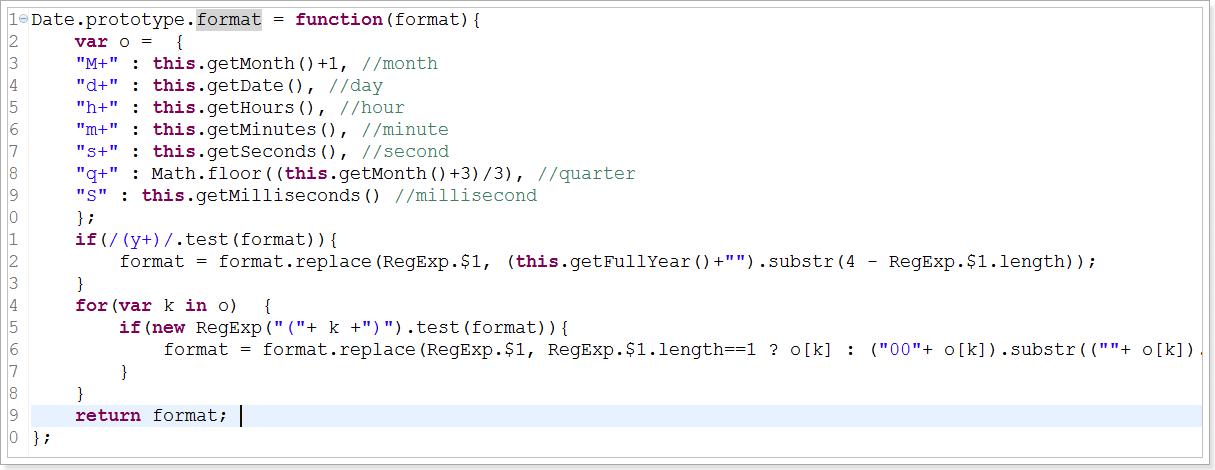
### 页面中的显示

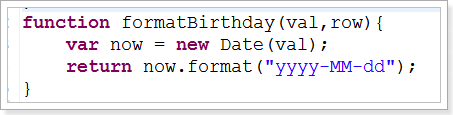




### 日期显示

扩展js的Data对象：





## 解决表单提交String2Date问题



# 小结

