# Mybatis第一天mybatis的基础知识

课程安排

Mybatis和spingmvc通过订单商品 案例驱动

第一天：基础知识（重点，内容量多）

对原生态jdbc程序（单独使用jdbc开发）问题总结

Mybatis框架原理

Mybatis人们程序

用户的增、删、改、查

Mybatis开发dao两种方法

原始dao开发方法（程序需要编写dao接口和dao实现类）（掌握）

Mybatis的mapper接口（相当于dao接口）代理开发方法（掌握）

Mybatis配置文件SqlMaoConfig.xml

Mybatis核心

Mybatis输入映射

Mybatis输出映射

Mybatis的动态sql

第二天：高级知识

订单商品数据模型分析

高级结果集映射（一对一、一对多、多对多）

Mybatis延迟加载

Mybatis查询缓存（一级缓存、二级缓存）

Mybatis和spring进行整合

Mybatis的逆向工程

## 对原生态jdbc程序（单独使用jdbc开发）问题总结

### 环境

Idk1.7、mysql5.1、myeclipse

### 问题总结

1. 数据库连接，使用时就创建，不使用立即释放，对数据库进行频繁连接开启和关闭，造成数据库资源浪费，影响数据库性能

设想：使用数据库连接池管理数据库连接

1. 将sql语句硬编码到java代码中，如果sql语句修改，需要重新编译java代码，不利于系统的维护。

设想：将sql语句配置到xml配置文件中，即使sql变化，不需要对java代码进行重新编译

1. 向preparedStatement中设置参数，对占位符位置和设置参数，硬编码在java代码中，不利于系统维护。

设想：将sql语句级占位符和参数全部配置在xml中

1. 从resultSet中遍历数据时，存在硬编码，将获取表的字段进行硬编码，不利于系统维护

设想：将查询的结果集，自动映射成java对象。

# Mybatis框架

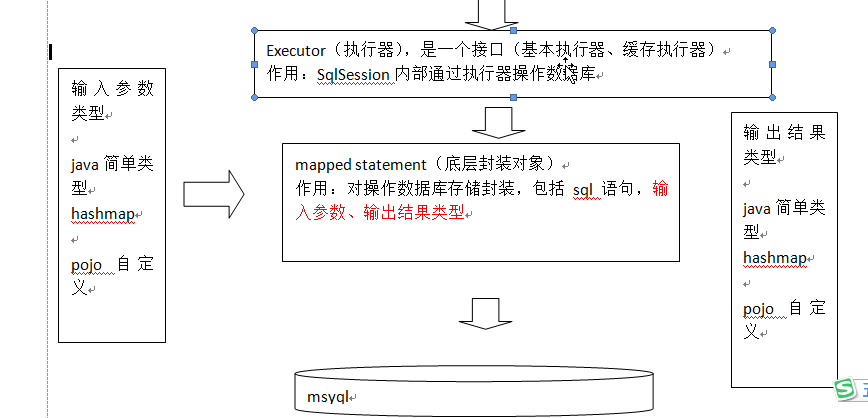
Mybatis是一个持久层的框架，是apache下的顶级项目

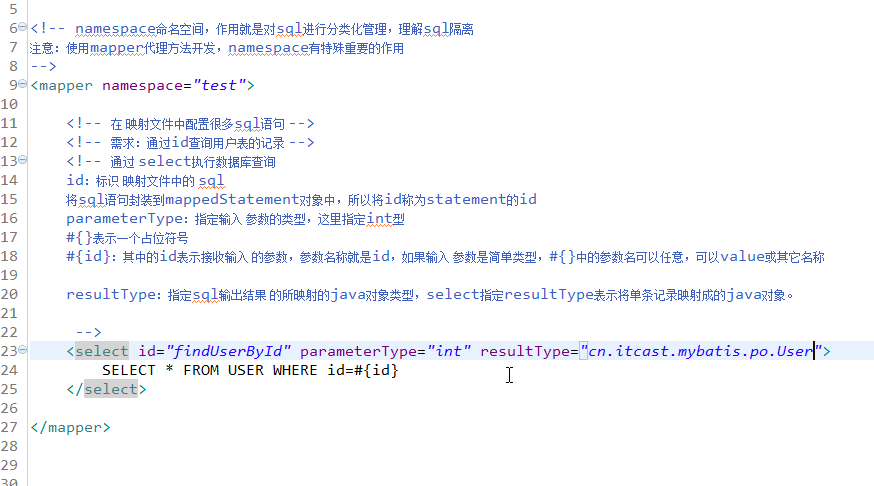
Mybatis托管到goolecode，再后来托管到github下（https://github.com/mybatis/mybatis-3/releases）

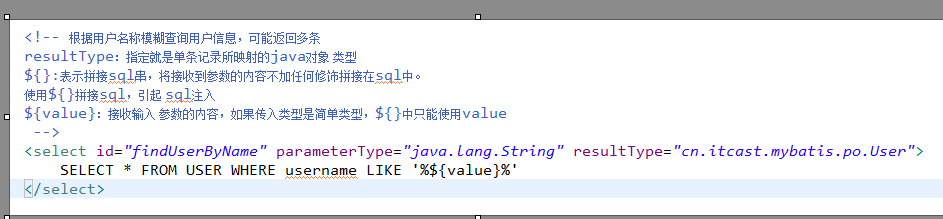
Mybatis让程序员主要精力放在sql中，通过mybatis提供的映射放射，自由的灵活生成（半自动化，大部分需要程序员编写sql）满足需要sql语句。

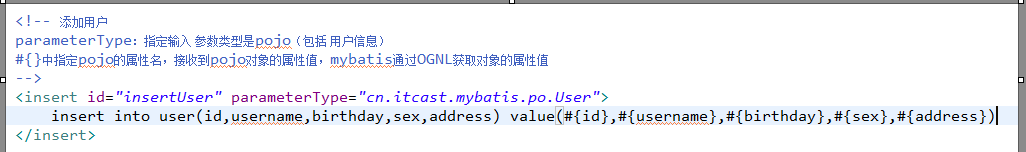
Mybatis可以将向preparedStatement中的输入参数自动进行输入映射，将查询结果集灵活映射成java对象（输出映射）

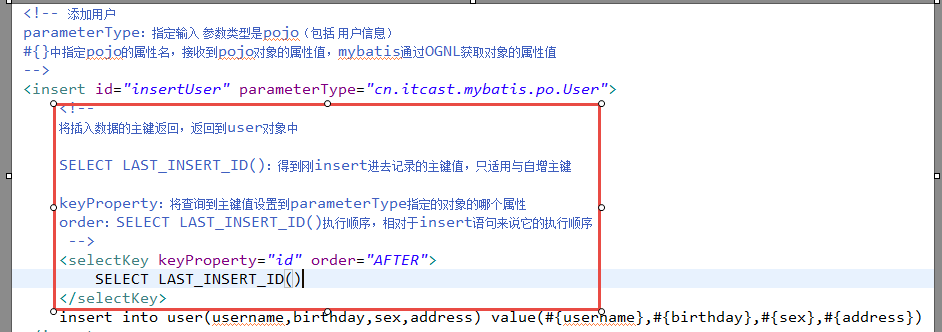


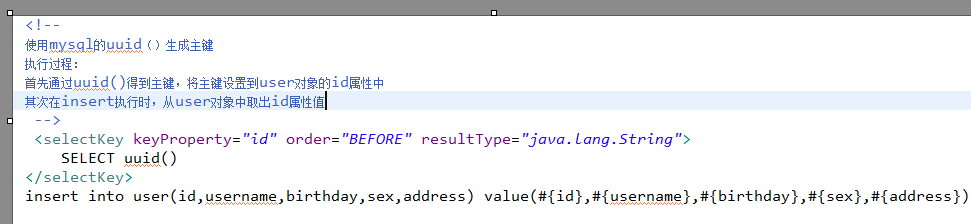


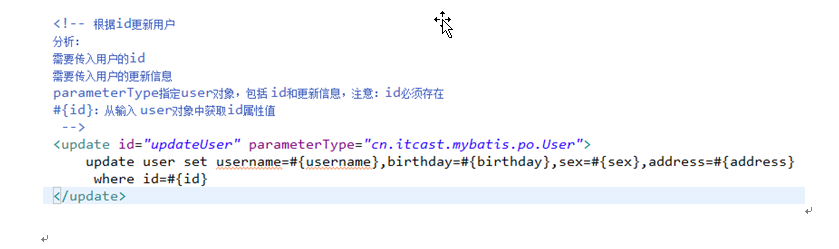


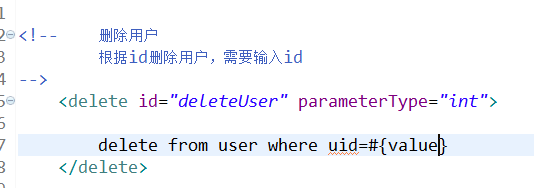


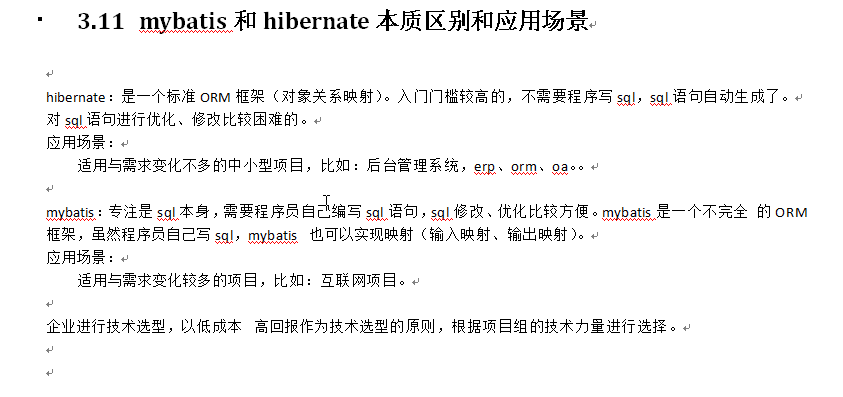












SqlSession是线程不安全的，在SqlSession实现类中除了有接口中的方法（操作数据库的方法）还有数据域属性

SqlSession最佳应用场合在方法体内，定义成局部变量使用

# 1、原始dao的开发方法

写dao，写dao的实现类

它存在的问题

1. Dao接口实现类方法中存在大量模板方法，设想能将这些代码提取出来
2. 调用sqlSession方法时将statemate的id硬编码了
3. 调用sqlSession方法时传入的变量，由于sqlSession方法使用泛型，在编译阶段也不报错，不利于程序员开发

# 2、mapper代理方法（程序员只需要实现mapper接口（相当于dao接口））

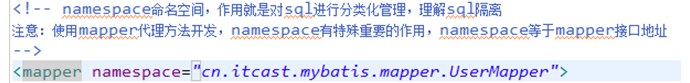
## 思路

程序员还需要编写mapper.xml映射文件

程序员编写mapper接口需要遵循一些开发规范，mybatis可以自动生成mapper接口实现类代理对象

开发规范：

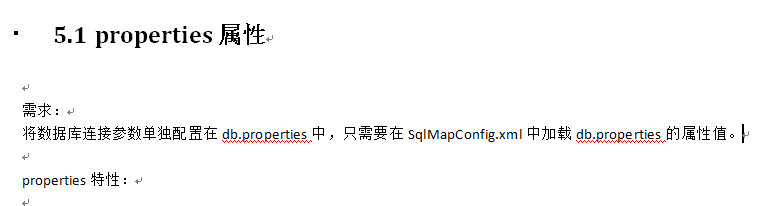
1. 在mapper.xml中namespace等于mapper接口地址



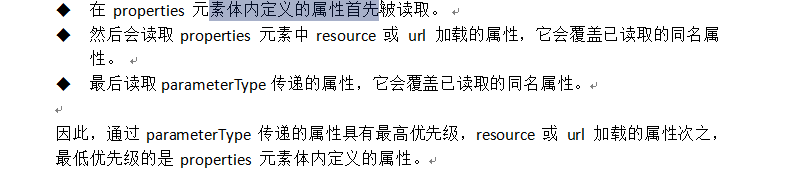
1. mapper.java接口接口中方法名和mapper.xml中statement的id一致
2. mapper.java接口中的方法输入类型和mapper.xml中statement的parameterType指定的类型一致
3. mapper.java接口中方法的返回值类型和mapper.xml中statement的resultType指定的类型一致

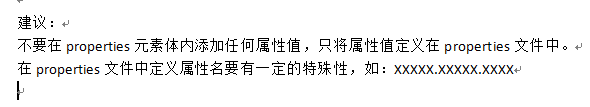


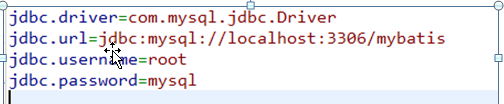
# SqlMapConfig.xml











## 5.2 typeAliases（别名）重点

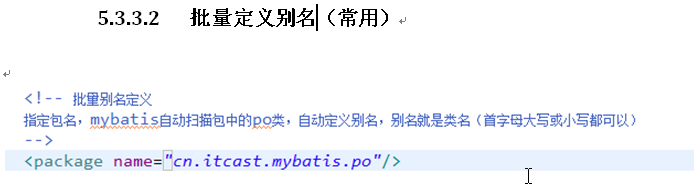
## 5.2.1需求

在mapper.xml中，定义很多的statement，statement需要parameterType指定输入参数的类型，需要resultType指定输出参数的类型

如果在指定类型的时候输入类型的全路径，不方便进行开发，可以针对parameterType或resultType指定的类型定义一些别名，在mapper.xml中通过别名定义，方便开发

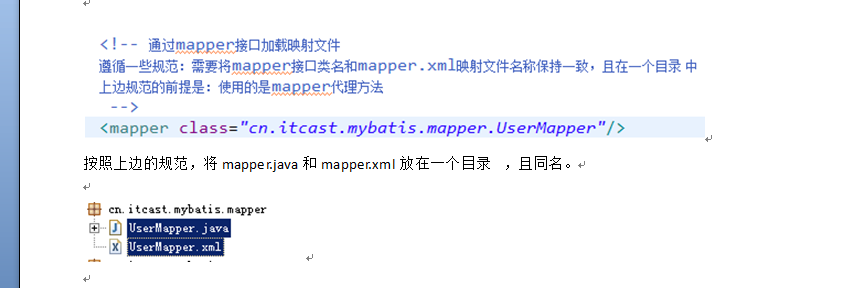
## 单个别名的定义







## 单个加载



## 批量加载

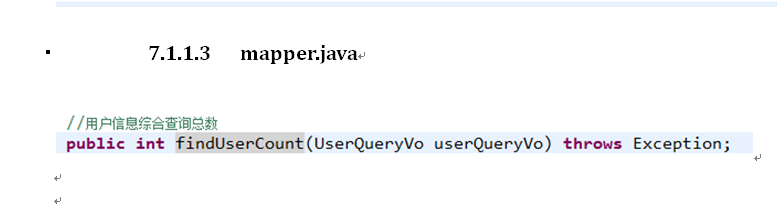


## 多domin的封装类型

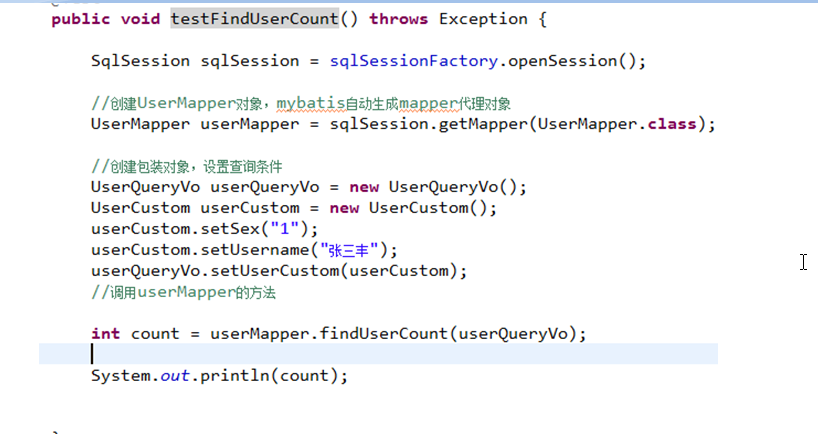






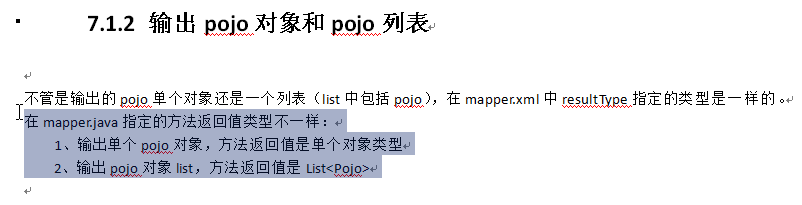


## 测试代码



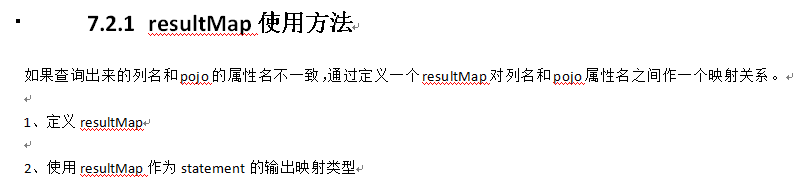
# 小结

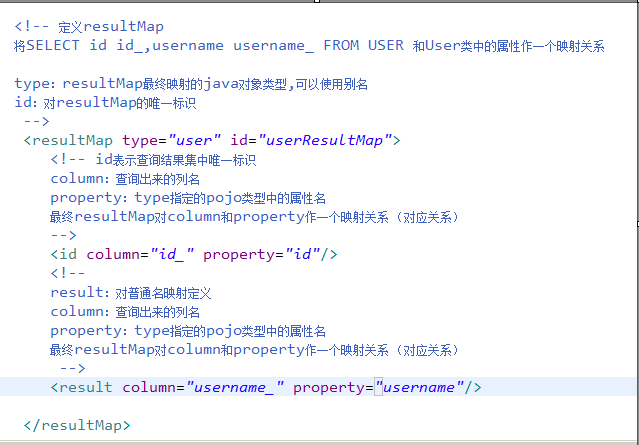
查询出来的结果集只有一行且一列，可以使用简单类型进行映射



# resultMap

mybatis中使用resultMap完成高级输出结果映射



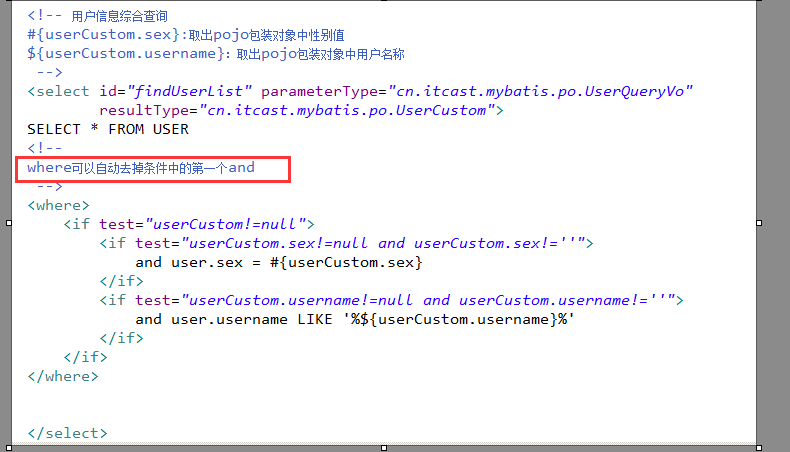




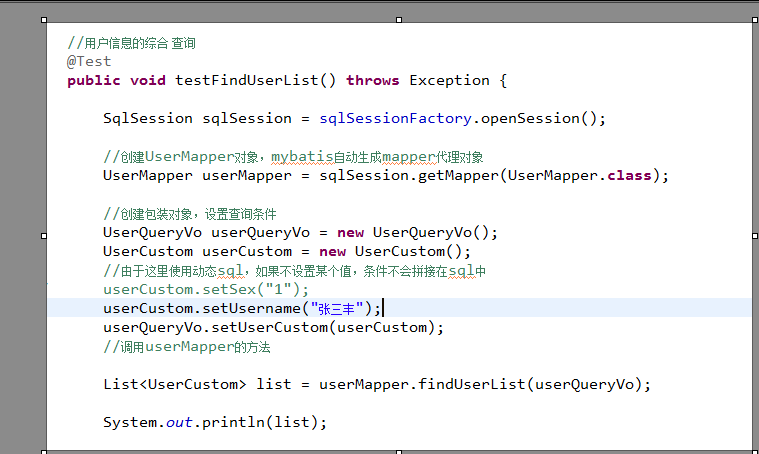
# 动态sql

## 什么是动态sql

Mybatis核心对sql语句进行灵活操作，通过表达式进行判断，对sql进行灵活拼接、组装



## 测试代码



# Sql片段

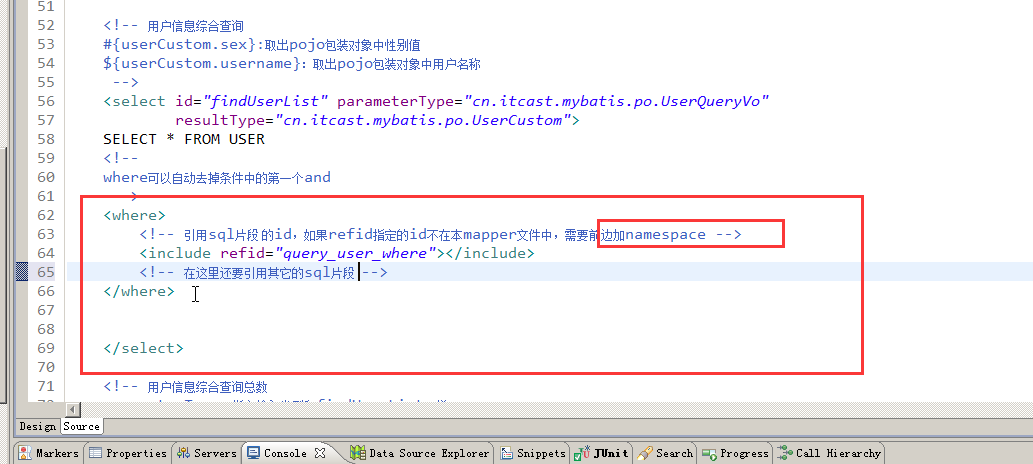
## 需求

将相同的动态代码块抽取出来，组成一个sql片段，其它的statement中就可以引用，方便程序员进行开发

## 定义sql片段



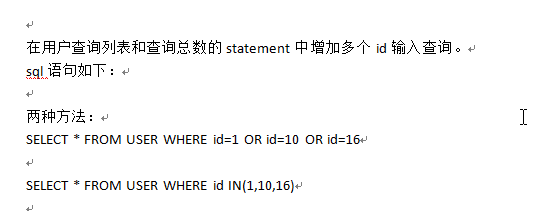
## 引用sql片段

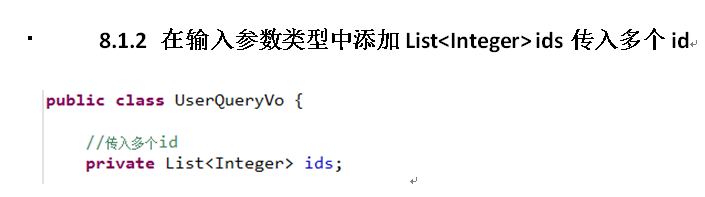


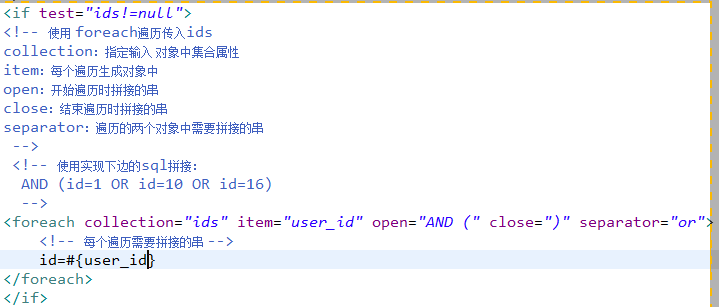
# Foreach标签

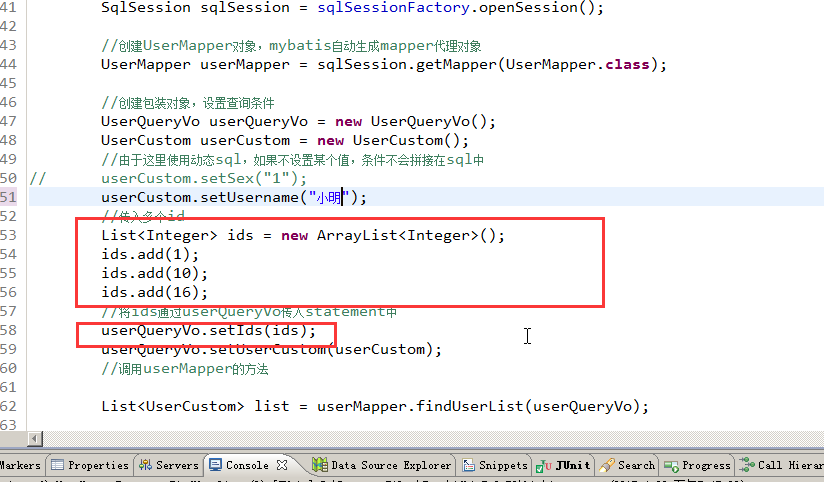
向sql传递数组或list，mybatis使用foreach解析

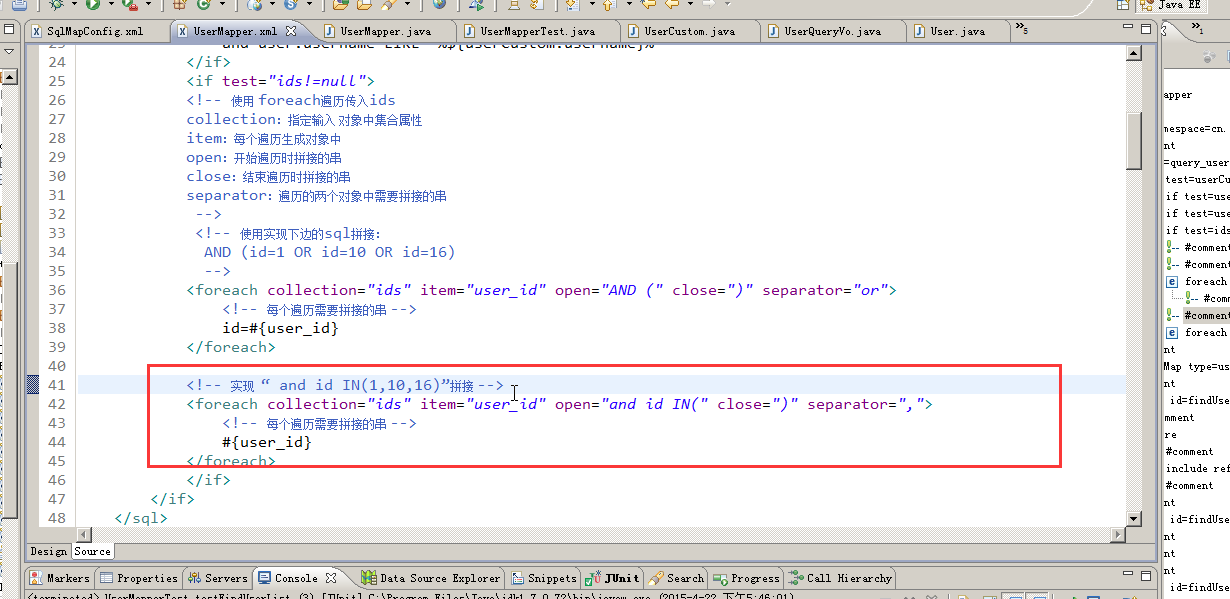
## 需求











# Mybatis第二天

Mybatis是什么？

Mybatis是一个持久层框架，mybatis是一个不完全的ORM。Sql预计需要程序员自己去编写，但是mybatis也有映射（输入参数映射、输出结果映射）

Mybatis入门的门槛不高，学习成本低，让程序员把精力放在sql语句上，对sql语句优化非常方便，适用于需求变化较多项目，比如互联网项目。

Mybatis框架执行过程：

1. 配置mybatis的配置文件，SqlMapConfig.xml（名称不固定）
2. 通过配置文件，加载mybatis运行环境，创建SqlSessionFactory会话工厂

SqlSessionFactory在实际使用时按单例方式

1. 通过SqlSessionFactory创建SqlSession

SqlSESSION是一个面向用户接口（提供操作数据库方法），实现对象是线程不安全的，建议SqlSession应用场合在方法体内

1. 调用SqlSession的方法去操作数据

如果需要提交事务，需要执行SqlSession的commit()方法

1. 释放资源，关闭SqlSession.

Mybatis开发dao的方法

1. 原始dao的方法

需要程序员自己编写dao接口和实现类

需要在dao实现类中注入一个SqlSessionFactory工厂。

1. mapper代理的开发（建议使用）

只需要程序员编写mapper接口（就是dao接口）

程序员在编写mapper.xml和mapper.java需要遵循一个开发规范：

1. mapper.xml中namespace就是mapper.java的全路径
2. mapper.xml中id值和mapper中的方法名一致
3. mapper.xml中statement的parameterType指定的输入参数类型和mapper.java的方法的参数类型一致
4. mapper.xml中statement的resultType指定的输出结果的类型和mapper.java的方法返回值类型一致

SqlMapConfig.xml配置文件“可以配置properties属性、别名、mapper加载…….”

输入映射

parameterType：指定输入参数的类型可以简单类型、pojo、hashmap…

对于综合查询建议parameterType使用包装的pojo，有利于系统维护和扩展

输出映射

resultType:

查询到的列名和resultType指定的pojo的属性名一致，才能映射成功

resultMap:

可以通过resultMap完成一些高级映射

如果查询到的列名和映射的pojo的属性名不一致时，通过resultMap设置列名和属性名之间的对应关系（映射关系）。可以完成映射

高级映射：

将关联查询到的列映射到一个pojo属性中（一对一）

将关联查询到的列映射到一个List<pojo>中（一对多）

动态sql：（重点）

If 判断（掌握）

where

foreach

sql片段（掌握）

课程安排

高级映射：（了解）

实现一对一查询、一对多、多对多查询

延迟加载

查询缓存

一级缓存

二级缓存（了解mynatis二级缓存使用场景

Mybatis和spring整合（掌握）

逆向工程

## resultType和resultMap它们的用法

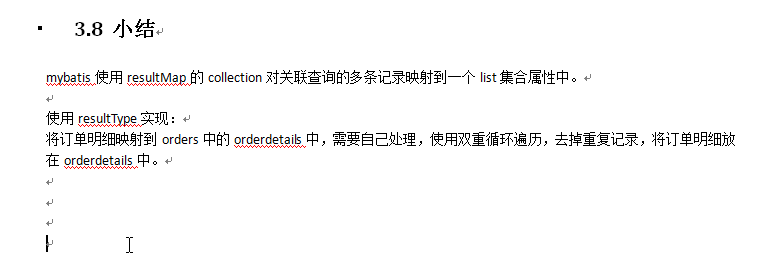
实现一对一查询

resultType:使用resultType实现较为简单，如果pojo中没有包括查询出来的列名，需要增加列名的属性，即可完成映射。

如果没有查询结果的特殊要求建议使用resultType。

resultMap:需要单独定义resultMap，实现有点麻烦，如果对查询结果有特殊的要求，使用ResultMap可以完成将关联查询映射pojo的属性中

resultMap可以实现延时加载，resultType无法实现延时加载。



# 延迟加载

## 什么是延迟加载



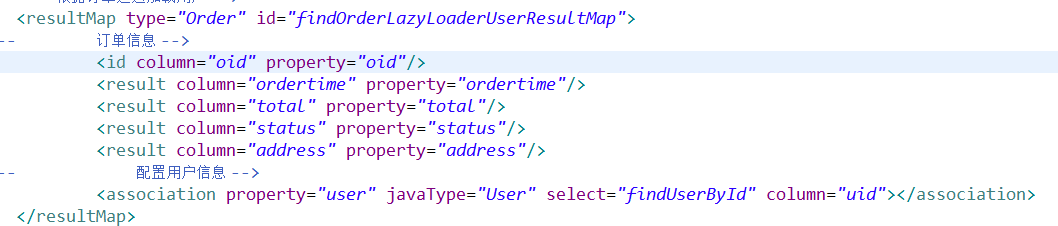
6.2使用assocation实现延迟加载

需求：查询订单并且关联查询用户

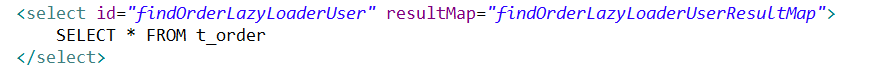
mapper.xml

需要定义两个mapper.xml

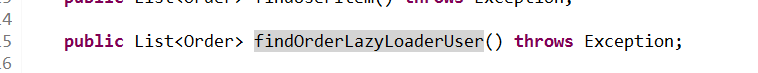
1. 先查询订单信息
2. 再查询用户信息

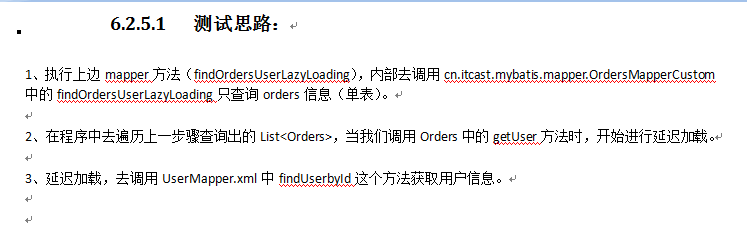




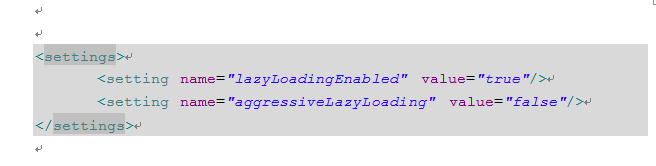


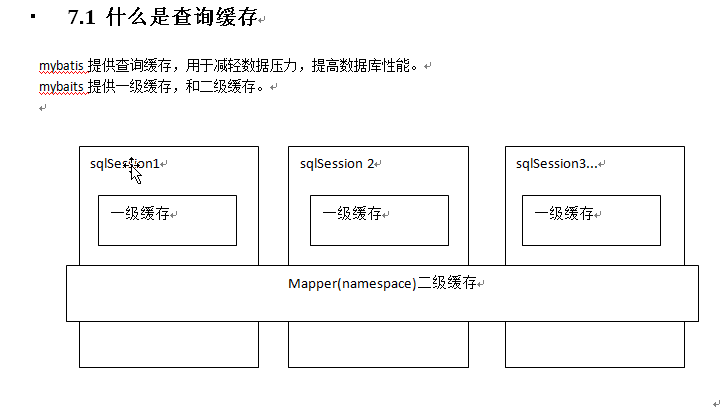
mapper.java





SqlMapConfig.xml把懒加载打开，并设置为不灵活



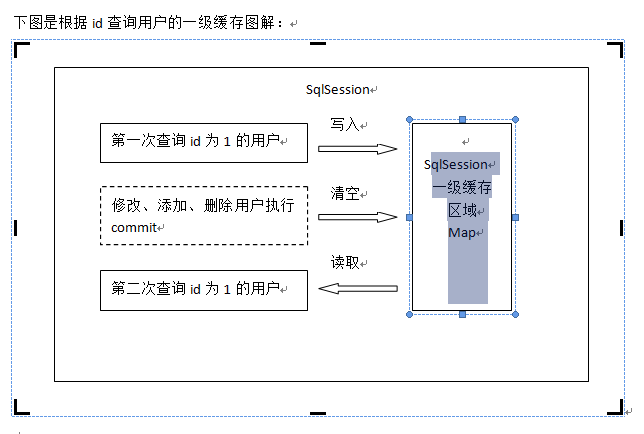


一级缓存是SqlSession级别的缓存。在操作数据库时需要构造sqlSession对象，在对象中有一个数据结构（HashMap）用于存储缓存数据。不同的SqlSession之间的缓存数据（HashMap）是互相不影响的。二级缓存是mapper级别的缓存，多个SqlSession去操作同一个Mapper的sql语句，多个SqlSession可以共存二级缓存，二级缓存是跨SqlSession的

为什么要用缓存？

如果缓存中有数据就不用再数据库中获取，大大提高了系统的性能

## 一级缓存



第一次发起插叙用户，id为1的用户信息，先去找缓存中id为1的用户信息，如果没有，从数据库查询用户信息，得到用户信息，将用户信息存储到一级缓存中。

如果SqlSession去执行commit操作（执行插入、删除、更新），清空SqlSession中的一级缓存，这样做的目的为了让缓存中存储的是最新的信息，避免脏读。

第二次发起查询用户id为1的用户信息，先去缓存中找是否有id为1的用户信息，缓存中有，直接从缓存中获取用户信息

## 一级缓存测试

Mybatis默认支持一级缓存，不需要在配置文件中配置。

正式开发，是将mybatis和spring进行整合开发，事务控制在service中，一个service中包括很多mapper方法调用

Service

//开始执行时，开启事务，创建SqlSession对象

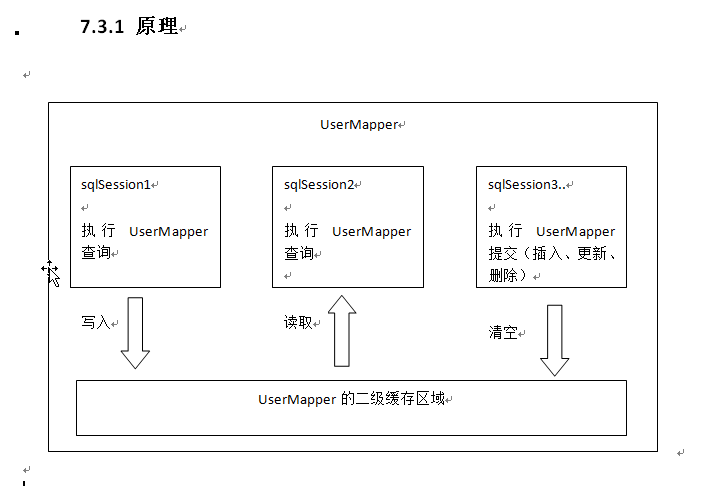
//第一次调用mapper的方法findUserById(1)

//第二次调用mapper的findUserById(1)，从一级缓存中获取数据

//方法结束，SqlSession关闭

如果是执行两次service调用查询相同的用户信息，不走一级缓存，因为session方法结束，SqlSession就关闭，一级缓存就清空

## 二级缓存



首先开启mybatis的二级缓存

SqlSession1去查询用户id为1的用户信息，查询到用户信息会将查询数据存储到二级缓存中

如果SqlSession3区执行相同mapper下sql，执行commit提交，清空mapper下的二级缓存

SqlSession2去查询用户id为1的用户信息，去缓存中找是否存在数据，如果存在直接从缓存中取出数据。

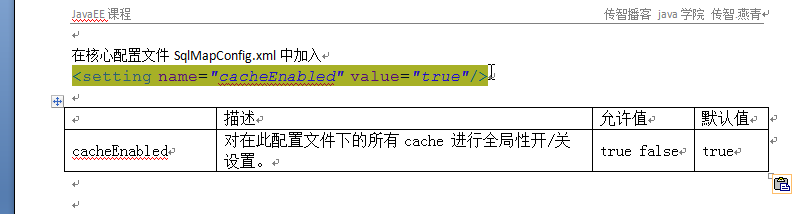
## 二级缓存和一级缓存的区别

二级缓存的范围大，多个SqlSession可以共享一个UserMapper的二级缓存区域。UserMapper有一个二级缓存区域（按namespace分），其它的mapper也有自己的二级缓存区域（按namespace分）

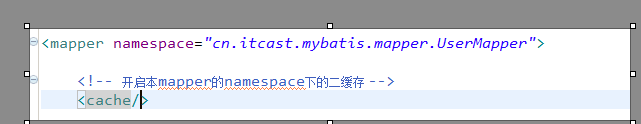
每一个namespace的mapper有一个二级缓存区域，两个mapper的namespace如果相同，这两个mapper执行sql查询到数据将存在相同的二级缓存区域中

开启二级缓存

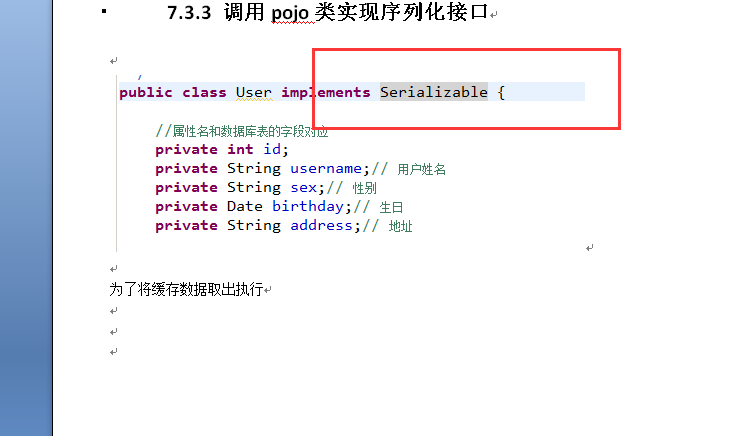
Mybatis的二级缓存是mapper范围级别，除了在SqlMapConfig.xml设置二级缓存的总开关，还要在具体的mapper.xml中开启二级缓存



---------------------------------------------------------------------------------------------------------------------------------

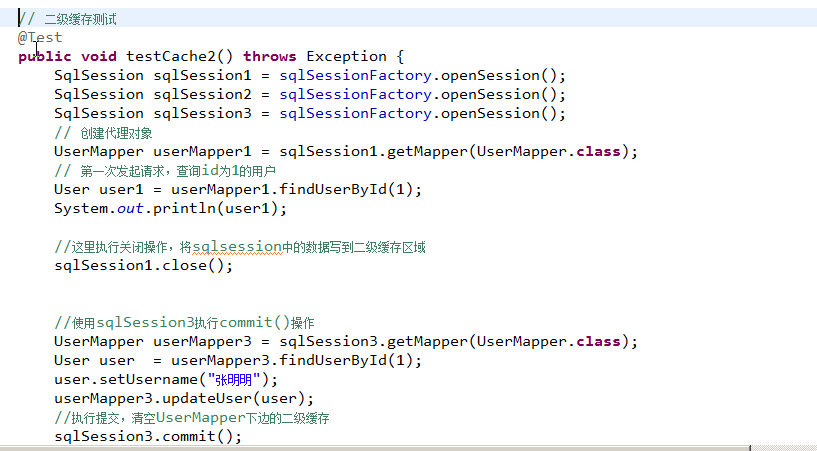


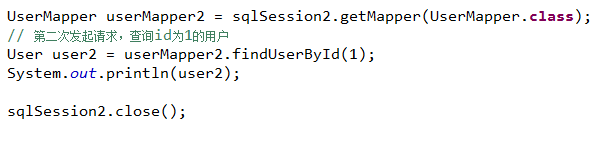
调用pojo类实现序列化接口

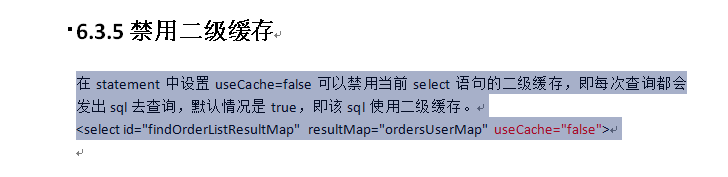


为了将缓存数据取出执行反序列化操作，因为二级缓存数据存储介质多种多样，不一定在内存（序列化为了存在在任意的地方）

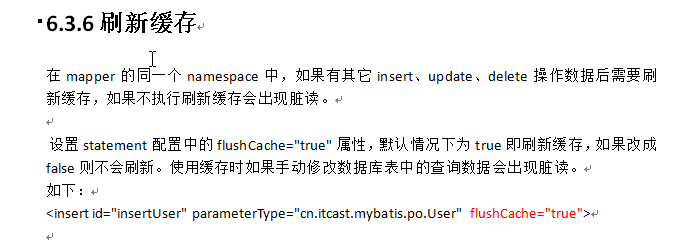
## 测试方法







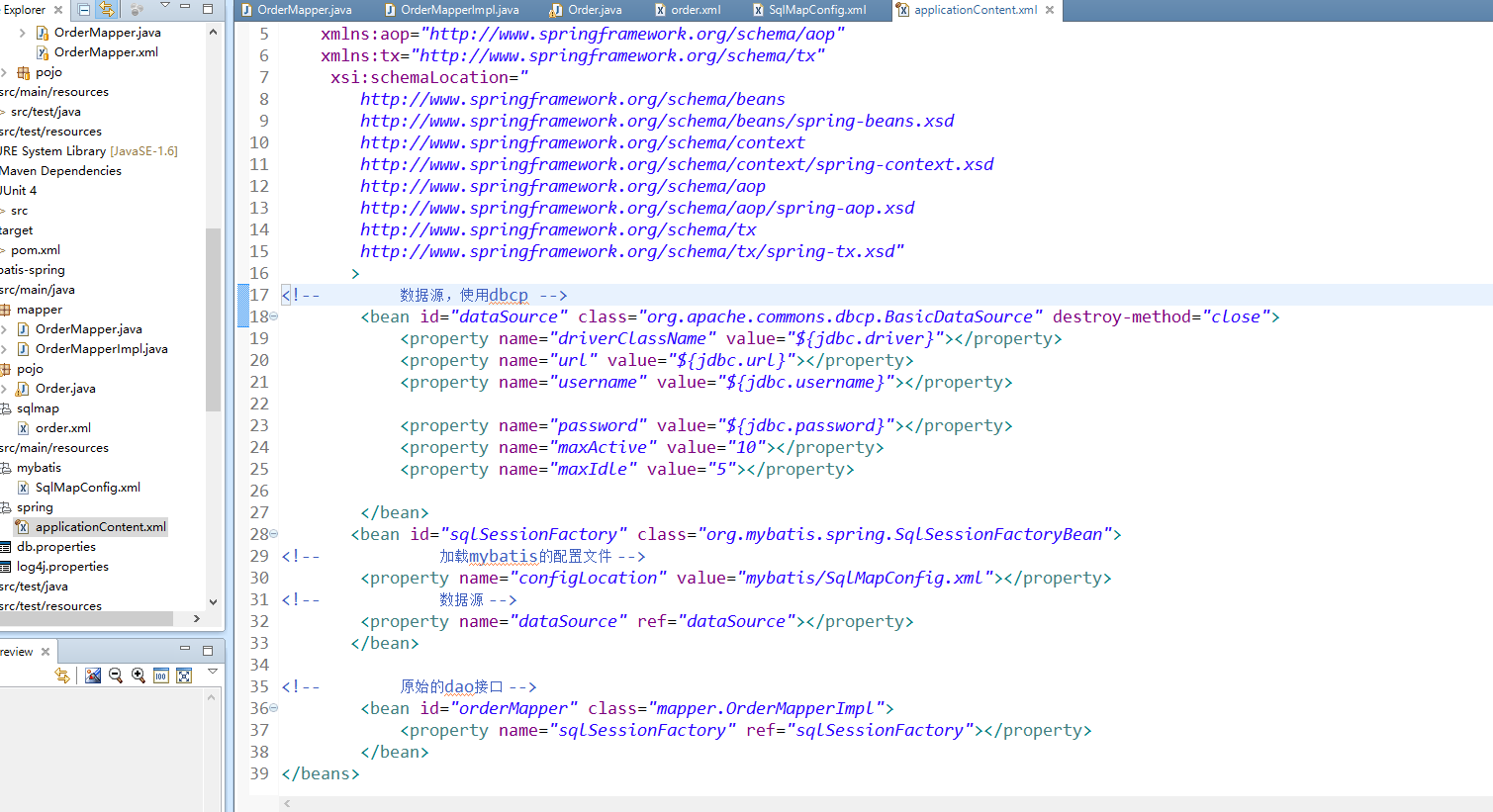
总结：针对每次查询都需要最新的数据sql，要设置成useCache=”false

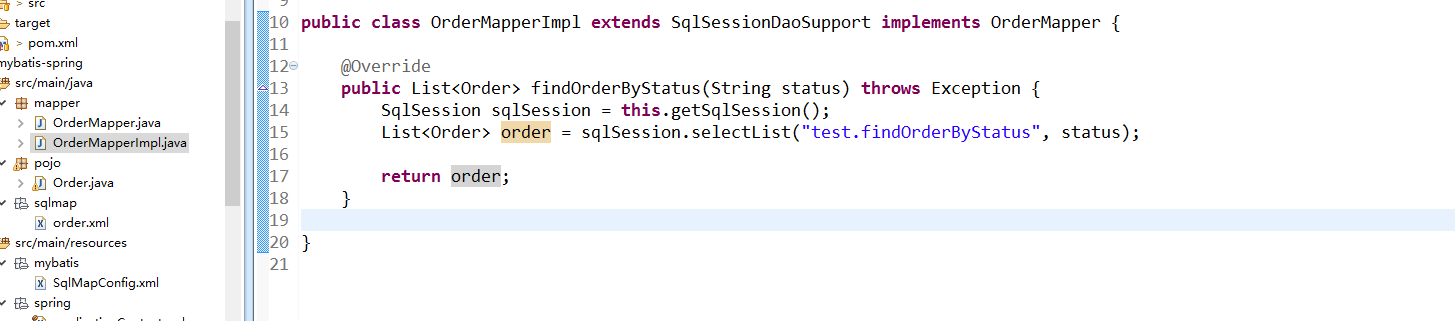


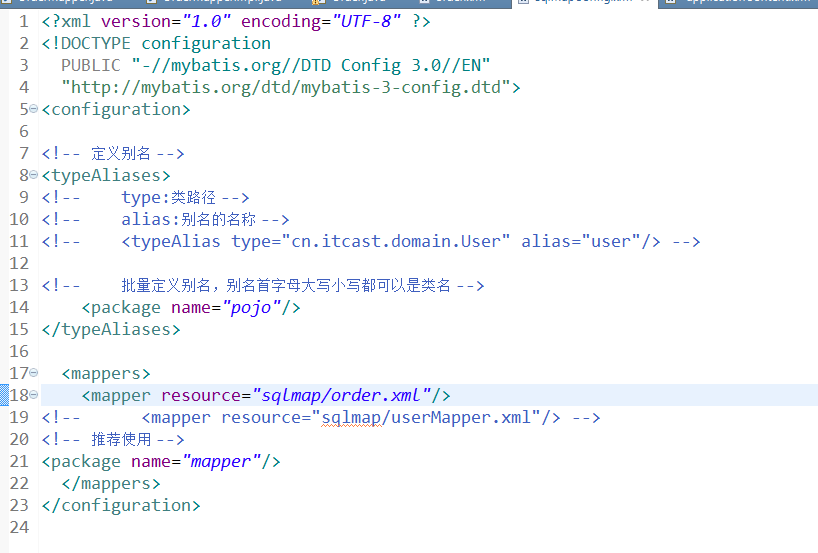
总结：一般执行完commit操作都需要刷新缓存，flushCache=true表示刷新缓存，这样可以避免数据库的脏读”

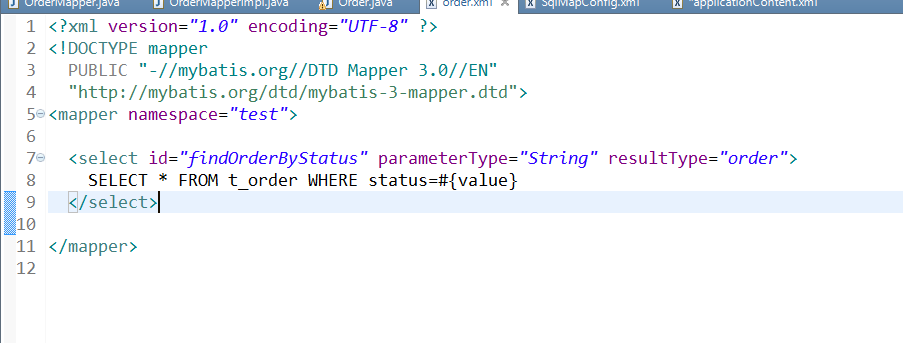
# Mybatis整合spring

## 原始dao与dao实现类

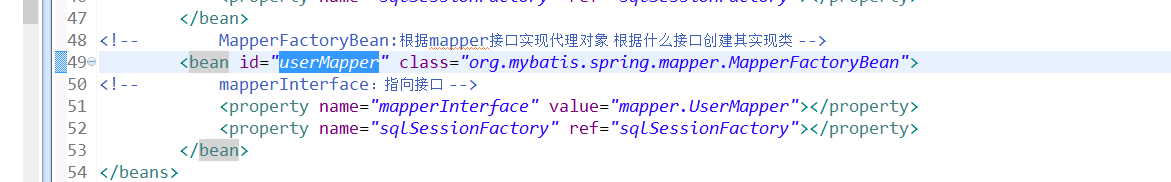


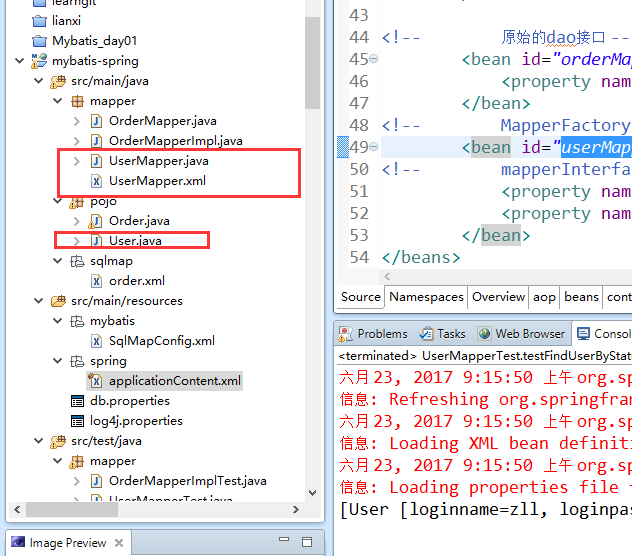


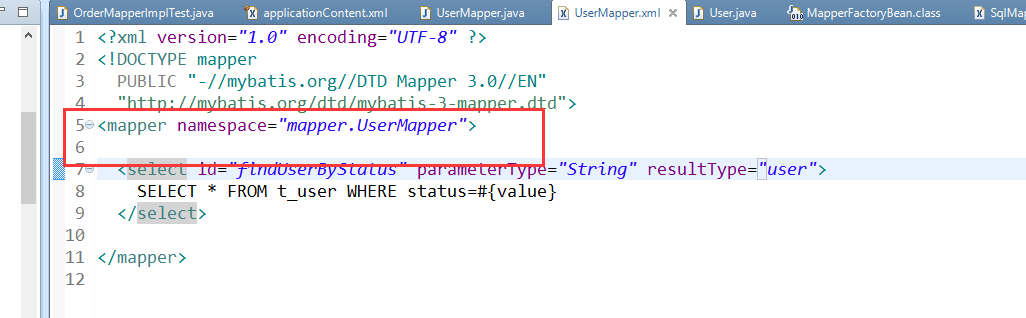


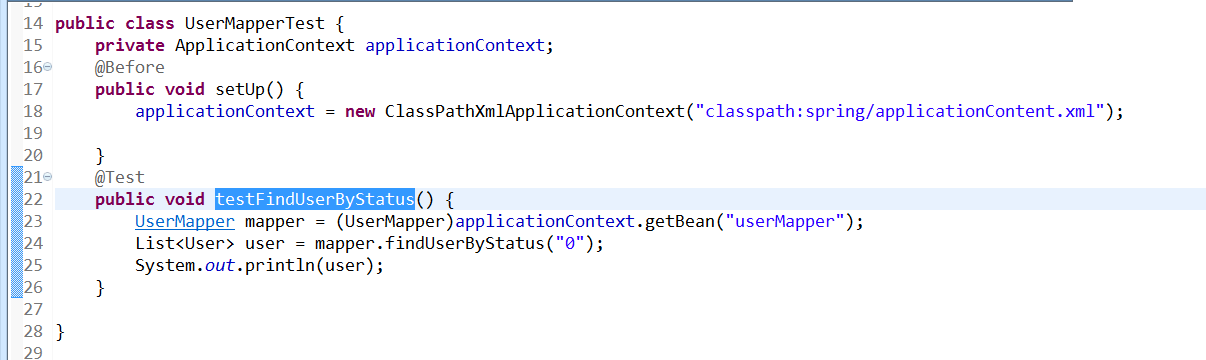


## Mapper代理









此方法问题：

需要针对每个mapper进行配置，麻烦。

解决方法：mapper批量扫描，从mapper包中扫描出mapper接口，自动创建代理对象并且在spring容器中注册

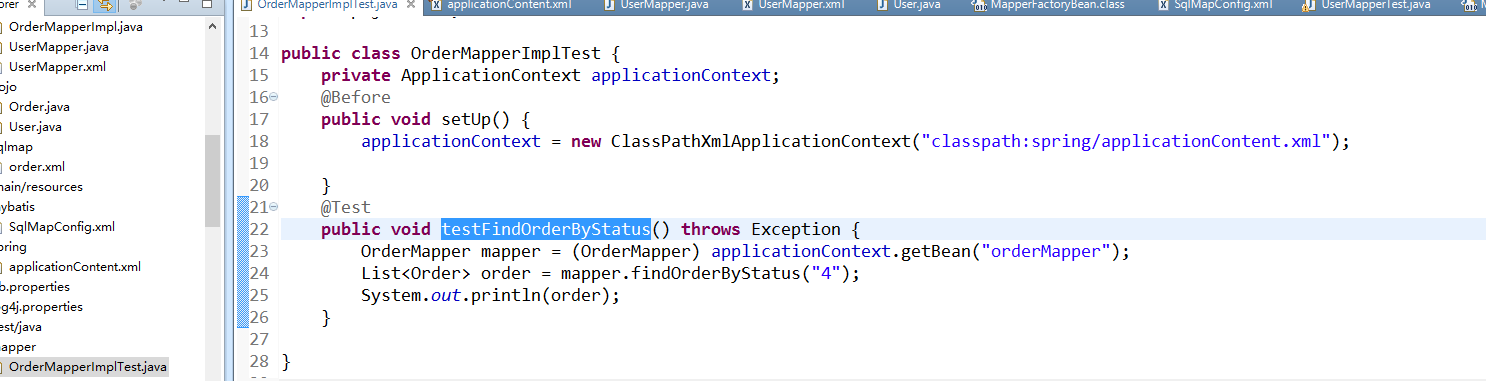
通过配置扫描器：MapperScannerConfigurer

注意：如果扫描多个包，每个包之间使用半角逗号分隔

这里面设置半角符



测试的方法：把前面配置的mapper相关的注释掉



http://blog.csdn.net/pucao\_cug/article/details/64499355

http://www.cnblogs.com/sprinng/p/5141233.html

# Springmvc第一天的基础

课程安排

第一天：springmvc的基础知识

什么是springmvc?

Springmvc框架原理（掌握）

处理器映射器、处理器适配器

Springmvc入门程序

目的：对前端控制器、**处理器映射器、处理器适配器**、视图解析器学习

非注解的处理器映射器、处理器适配器

注解的处理器映射器、处理器适配器（掌握）

Stringmvc和mybatis整合（掌握）

Springmvc注解开发

常用的注解学习

参数绑定（简单类型、pojo、集合类型）

自定义参数绑定（掌握）

Springmvc和struts2区别

第二天：springmvc的高级应用

参数绑定（集合类型）

数据回显

上传图片

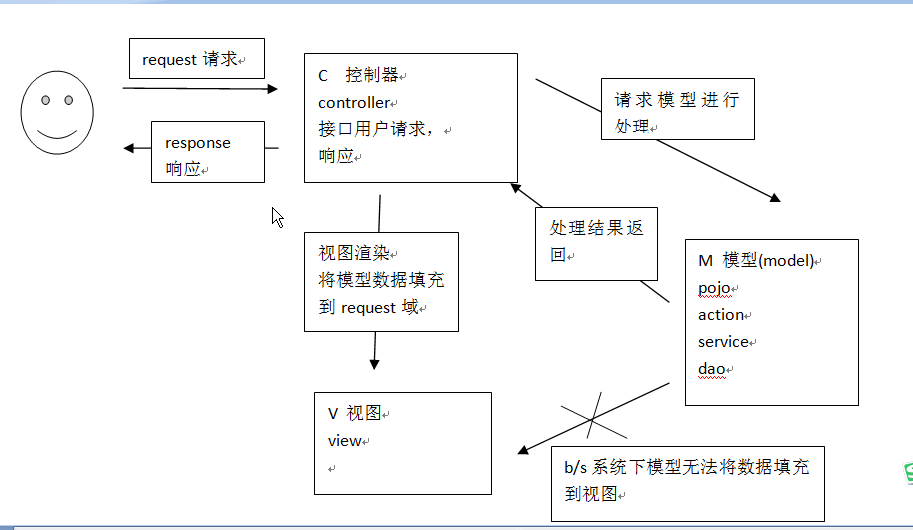
Json数据交互

RESTful支持

拦截器

# Mvc在b/s系统下的应用

mvc是一个设计模式，mvc在b/s系统下的应用



# Spring和mybatis整合

1整合思路：

需要spring通过单例方式管理SqlSessionFactory

Spring和mybatis整合生成代理对象，使用SqlSessionFactory创建SqlSession。（spring和mybatis整合自动完成）

持久层的mapper都需要由spring进行管理

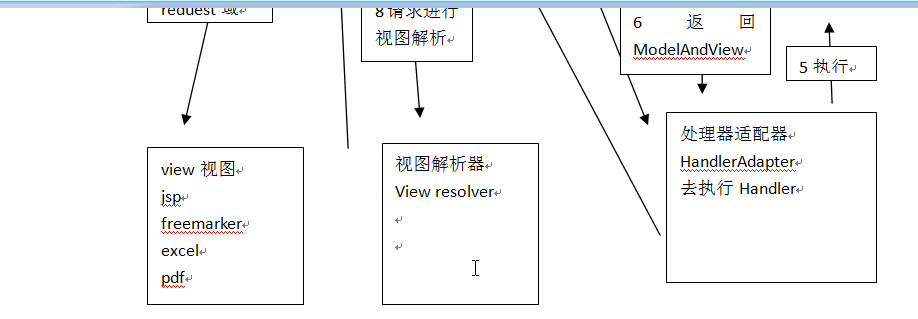
2整合环境

创建一个新的java工程（接近实际开发的工程）

3 SqlSwssionFactory

# Springmvc的框架





第一步：发起请求到前端控制器（DispatcherServlet）

第二步：前端请求器请求handMapping查找Hander

可以根据xml配置，注解进行查找

第三步：处理器映射器HaddingMapping向前端控制器返回Hadler

第四步：前端控制器调用处理器适配器去执行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ModelAndView

第七步：处理适配器向前端控制器返回ModelAndView

ModelAndView是springmvc框架的一个底层对象，包括Model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视图（jsp）

第九步：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：前端控制器向用户响应结果

# 组件

1. 前端控制器DispatcherServlet（不需要程序员开发）

作用：接收请求，相应结果，相当于一个转发器，重要处理器。

有了DispatcherServlet减少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. 处理器适配器HandlerAdapter

作用：按照特定规则（Handler要求的规则）去执行Handler

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

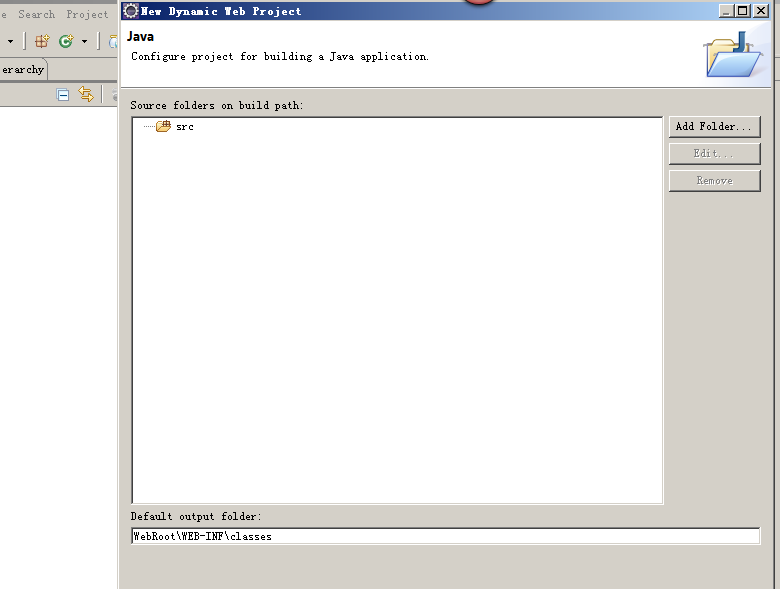
1. 处理器handler（需要程序员开发）
2. 视图解析器View resolver（不需要程序员开发

作用：进行试图解析，根据逻辑视图名解析成真正的视图（view）

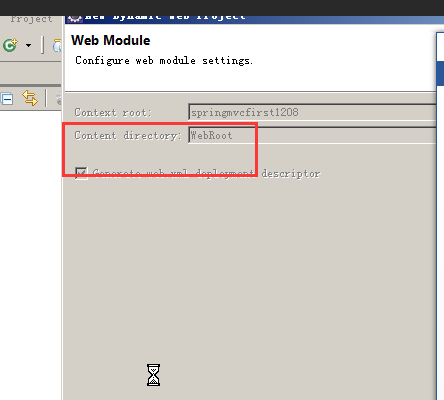
1. 视图View（需要程序员开发）

View是一个接口，实现类支出不同的View类型（jsp、freemarker、pdf…）

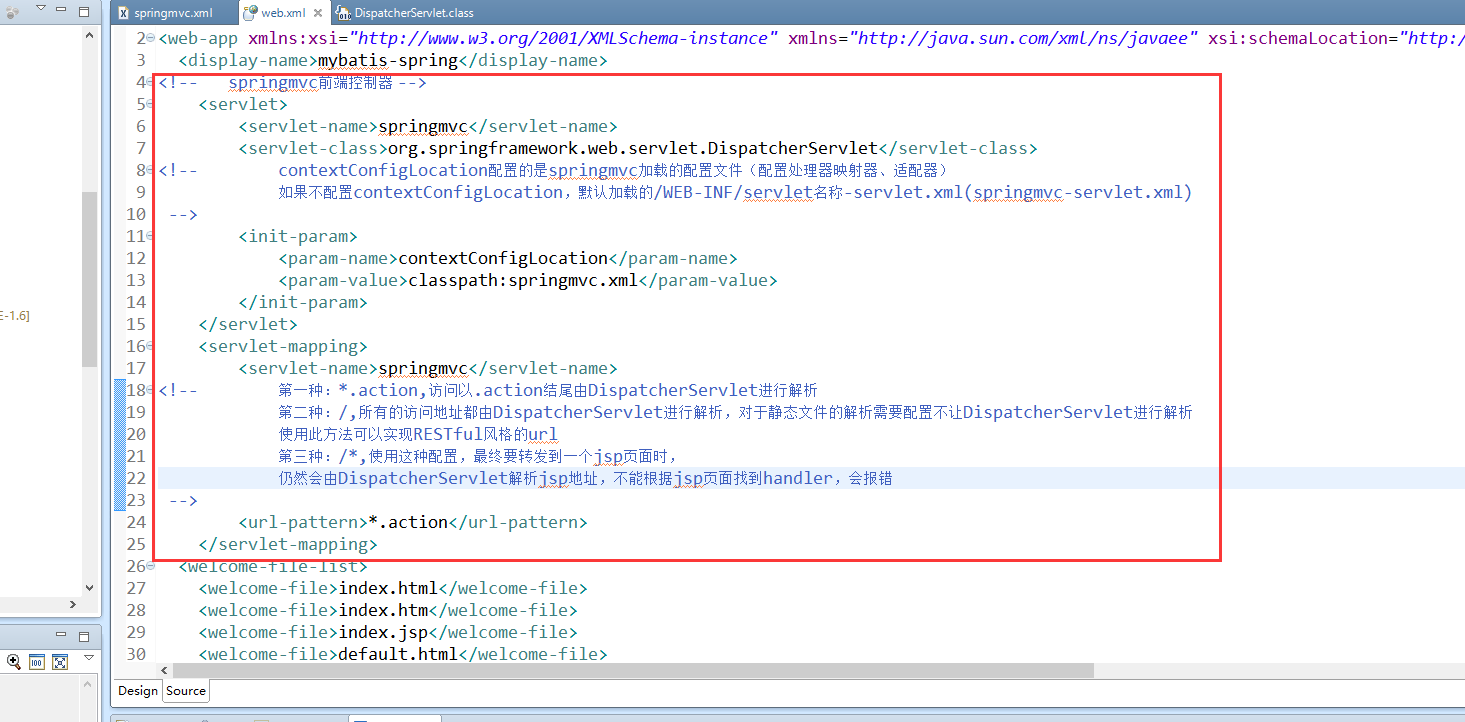
若是想要eclipse的程序能导在myeclipse中



把Dedault output folder 改成：WebRoot\WEB-INF\classes



1. Web.xml的配置（前端控制器的配置）



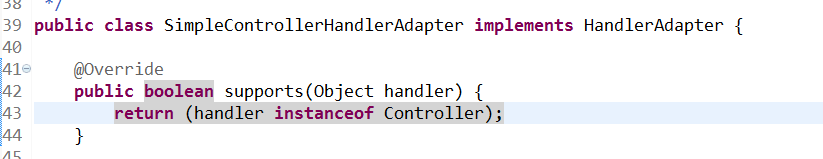
2、配置处理器映射器

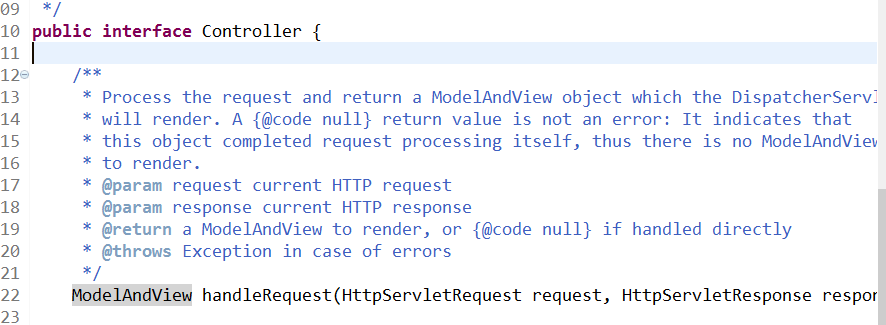
在springmvc.xml

3、配置处理器适配器

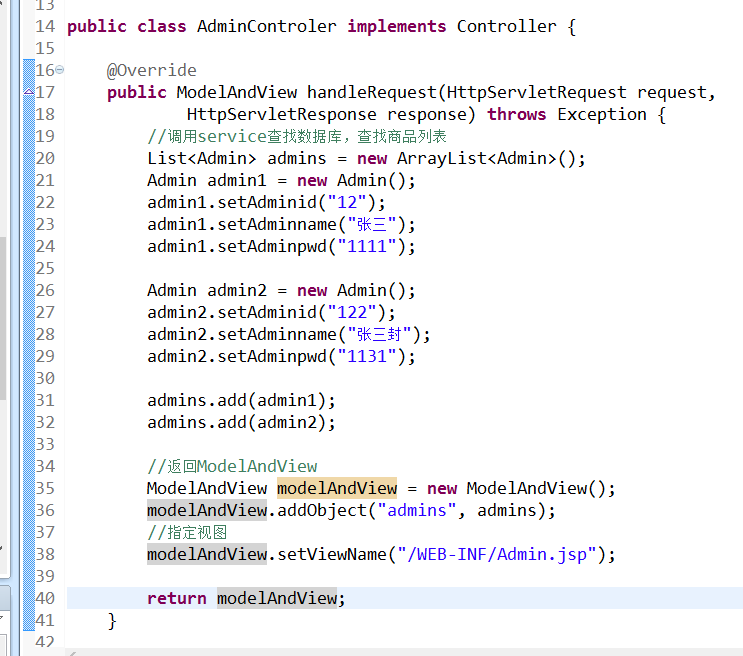
4、配置视图解析器

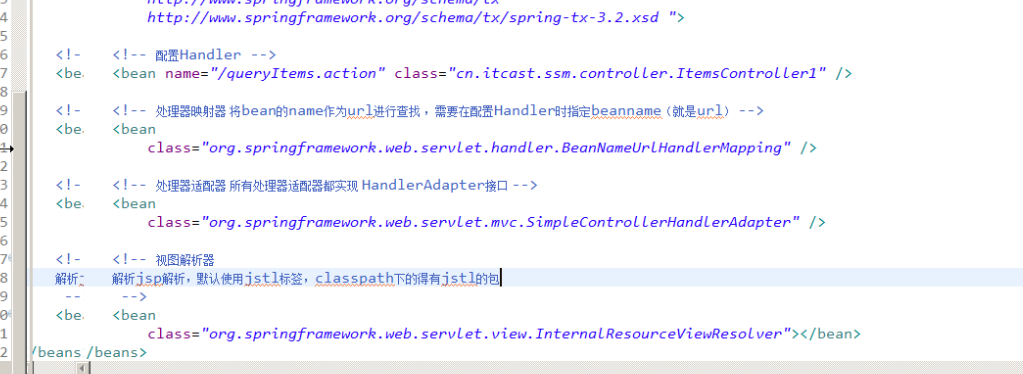
在springmvc.xml中进行配置查看源码

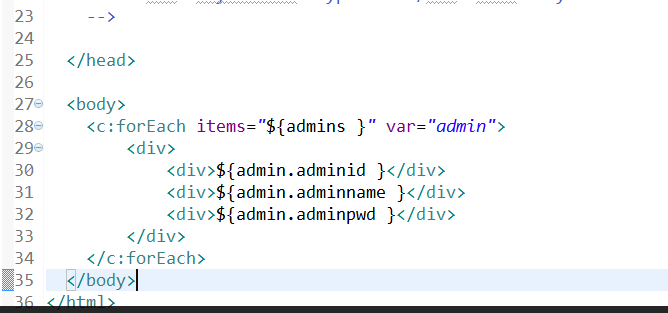




Controller



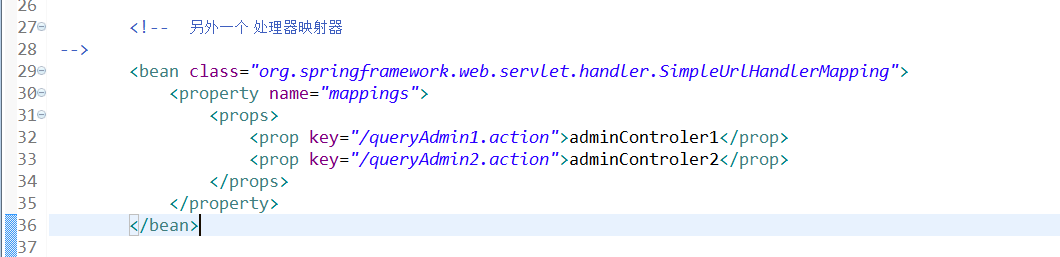




注意要是出现：说明视图指定的位置找不到



其他的处理器适配器和处理器映射器，它们之间可以混合着使用



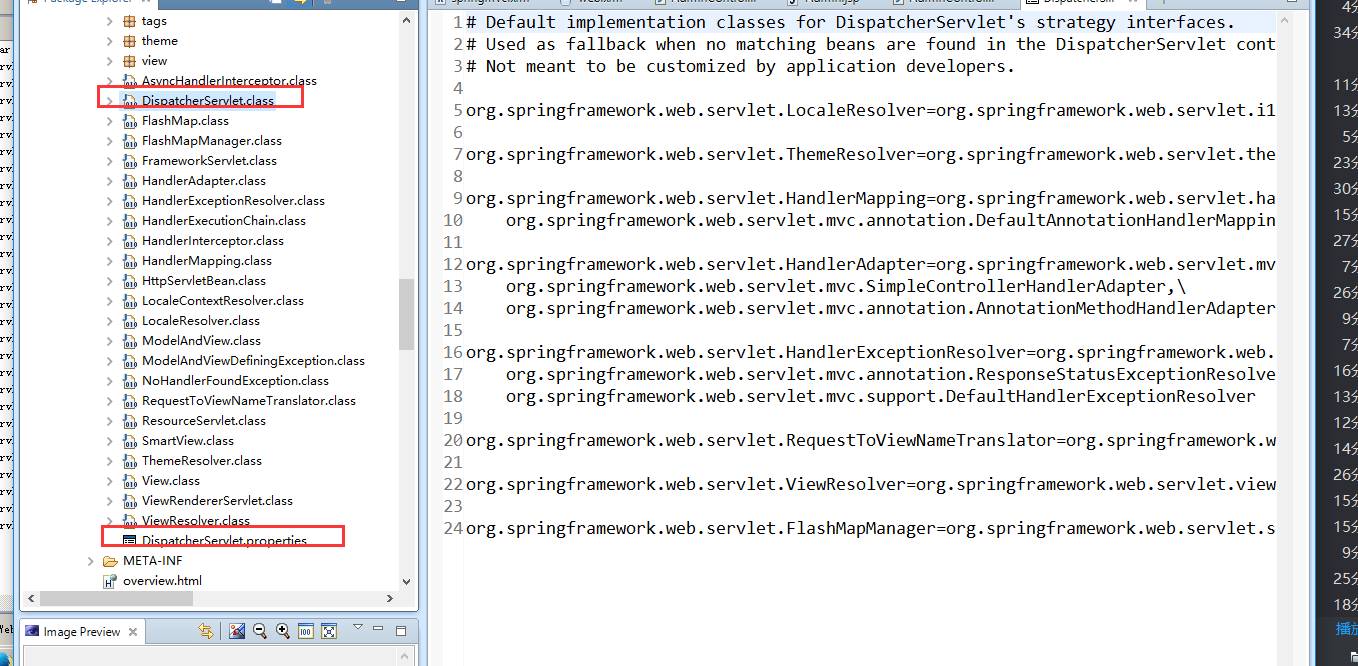




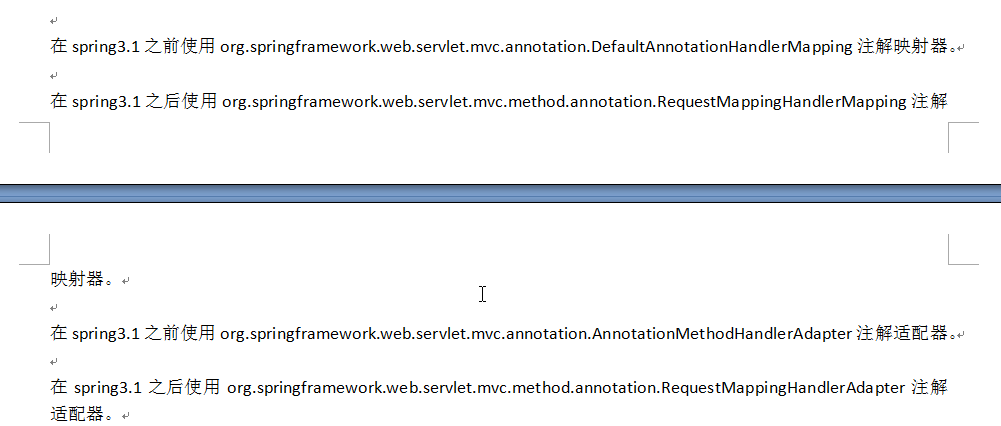
这个处理器适配器的好处是可以通过response来设置响应的格式如：json等

# 重点

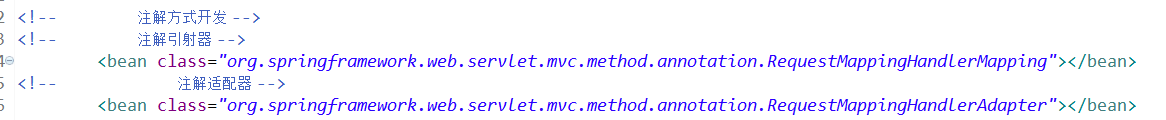
注解方式的映射器和适配器



前端控制器从上边的文件中加载处理映射器、适配器、视图解析器，如果不在springmvc中配置，使用默认加载的。



配置注解映射器和适配器（他们必须配对使用）

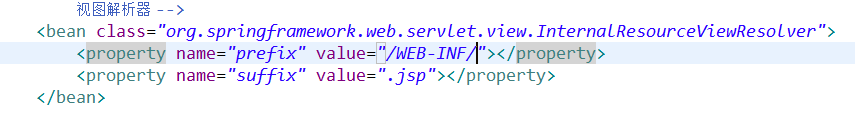


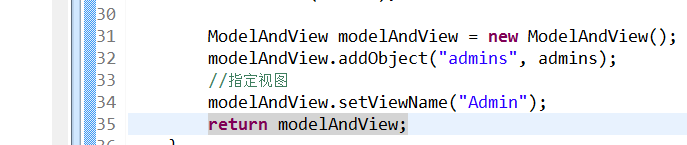
注解驱动 <mvc:annotation-driven></mvc:annotation-driven>



注解映射器的扫描器

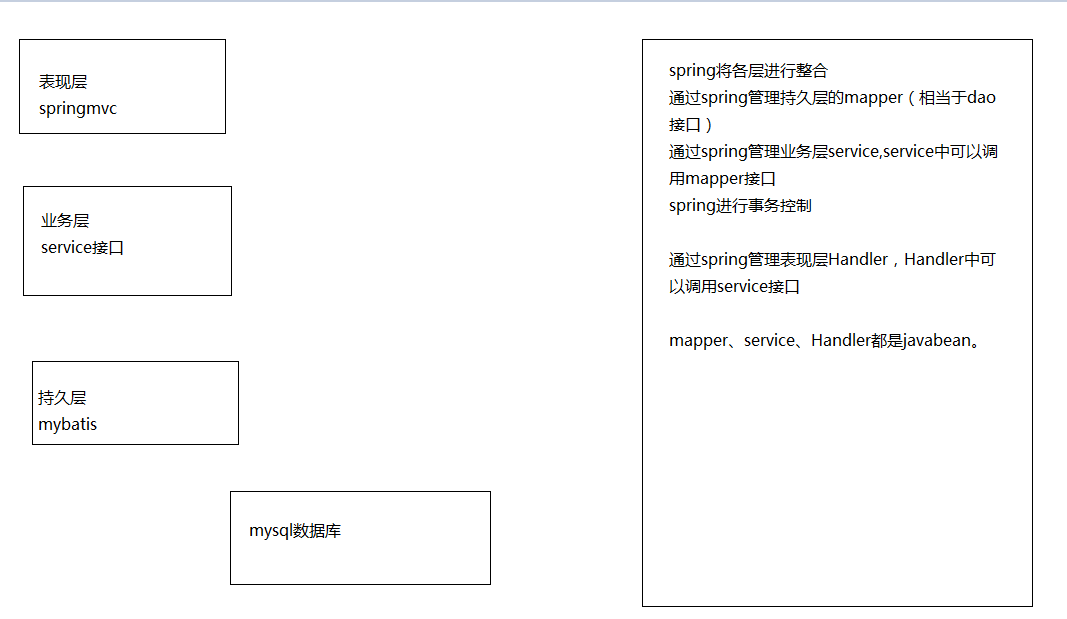


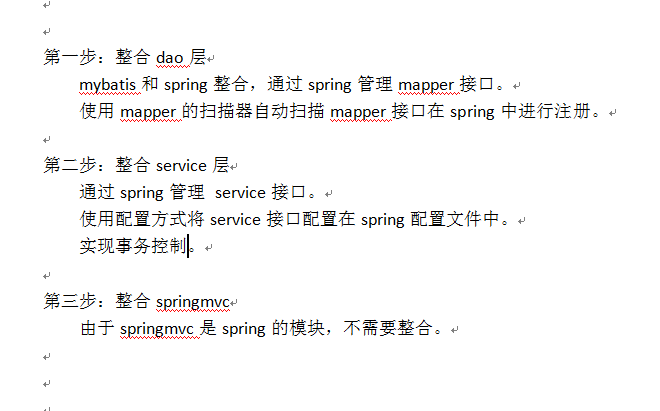




不用指定前缀和后缀

Springmvc和mybatis整合





所需要的jar包

数据库驱动的jar包

Mybatis的jar包

Mybatis和spring整合包

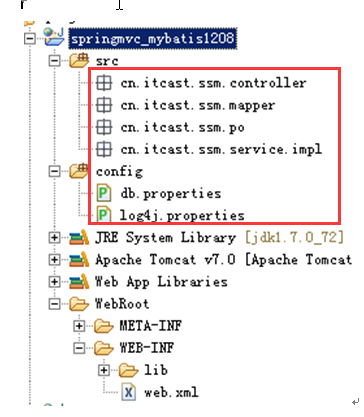
Log4j包

数据库连接池的包

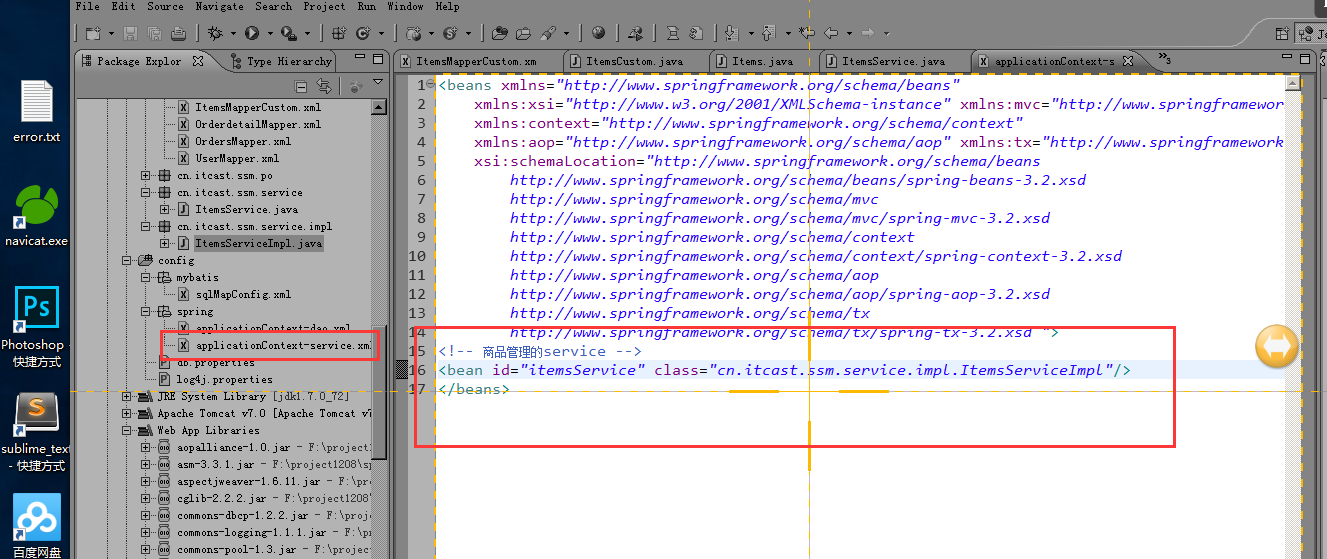
Spring所有的包

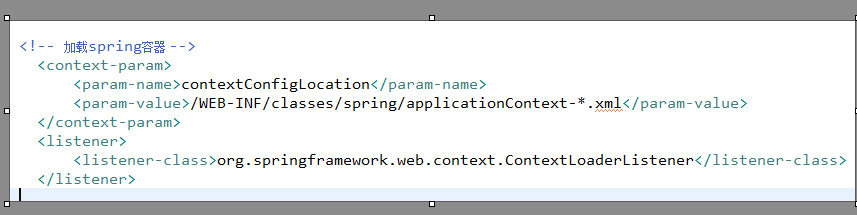
Jstl包

工程结构



注意：对于简单的增删查改等mapper.xml，mapper.java可以利用逆向工程生成，但是对于综合查询条件的，自己定义mapper编写。



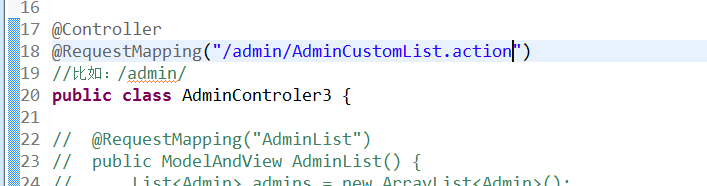




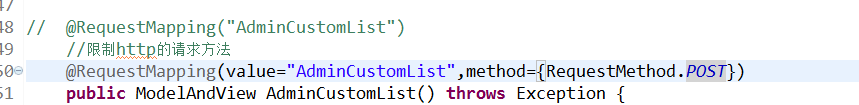
## @RequestMapping

定义controller方法对应的url，进行处理器映射使用

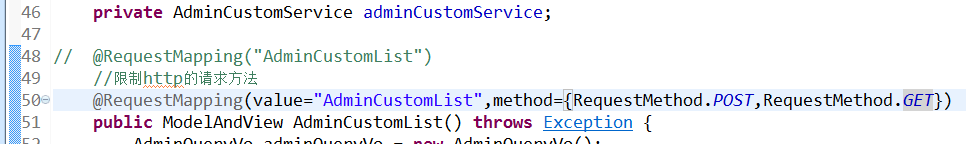
\*\* 窄化请求映射



\*\* 限制请求的方法



如果限制请求为post的方法那么不支持get。作用保证安全性



也可以两种方式都支持

## Controller方法的返回值

\*\* 返回ModelAndView

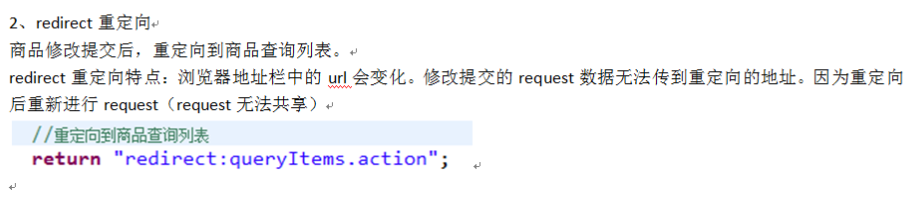
需要方法结束时，定义ModelAndView，将model和view分别进行设置

\*\* 返回String

如果controller方法返回string，表示返回逻辑视图名

真正试图（jsp路径）= 前缀+逻辑视图名+后缀





因为在同一个controller中，不可不写RequestMapping()的跟路径（就是前面说的窄化请求路径）

\*\* 返回void

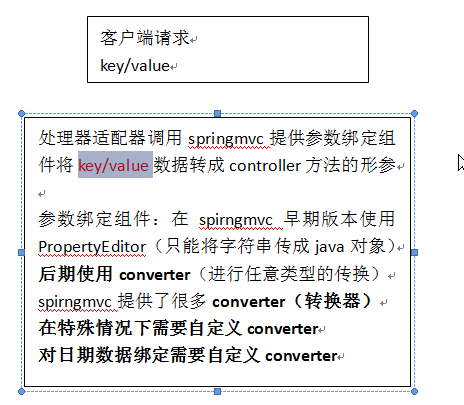


# 参数绑定

## Spring参数绑定的过程

从客户端请求key/value数据，经过参数绑定，将key/value数据绑定到controller方法的形参上。

Springmvc中，接受页面提交的数据是通过方法形参来接收。而不是像struts2中定义成员变量来接收！！！！！！



Controller方法（形参）

# 默认支持的类型

直接在controller方法形参

## HttpServletRequest

## HttpServletResponse

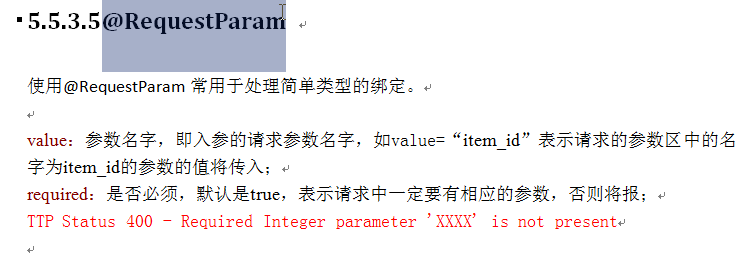
## HttpSession

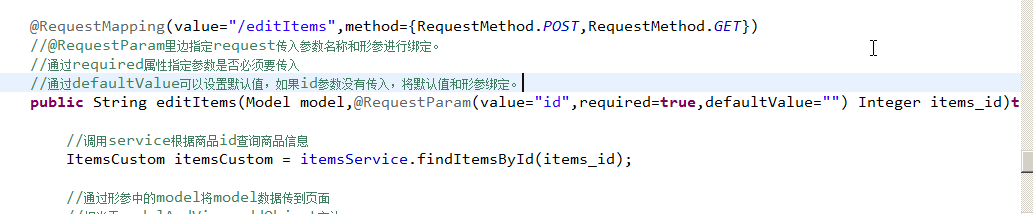
## Model/ModelMap

model是一个接口，modelMap是一个接口实现

作用：将model数据填充到request域

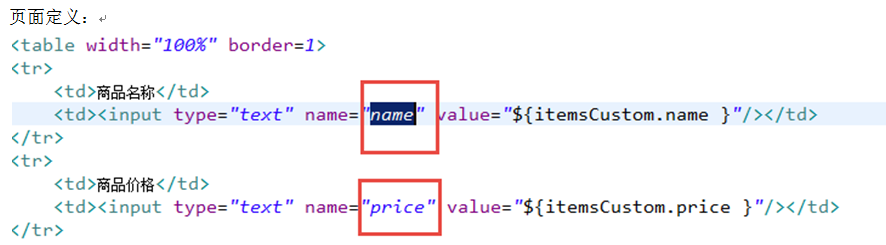
### 简单类型

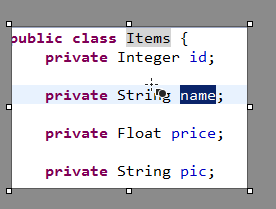


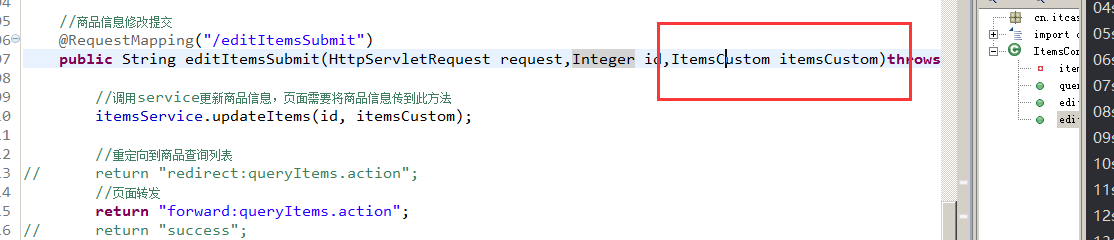


# Pojo的绑定

页面中input中name和controller的pojo形参中的属性名称一致，将页面中数据绑定到pojo。



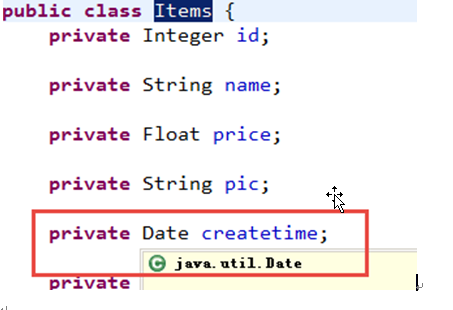




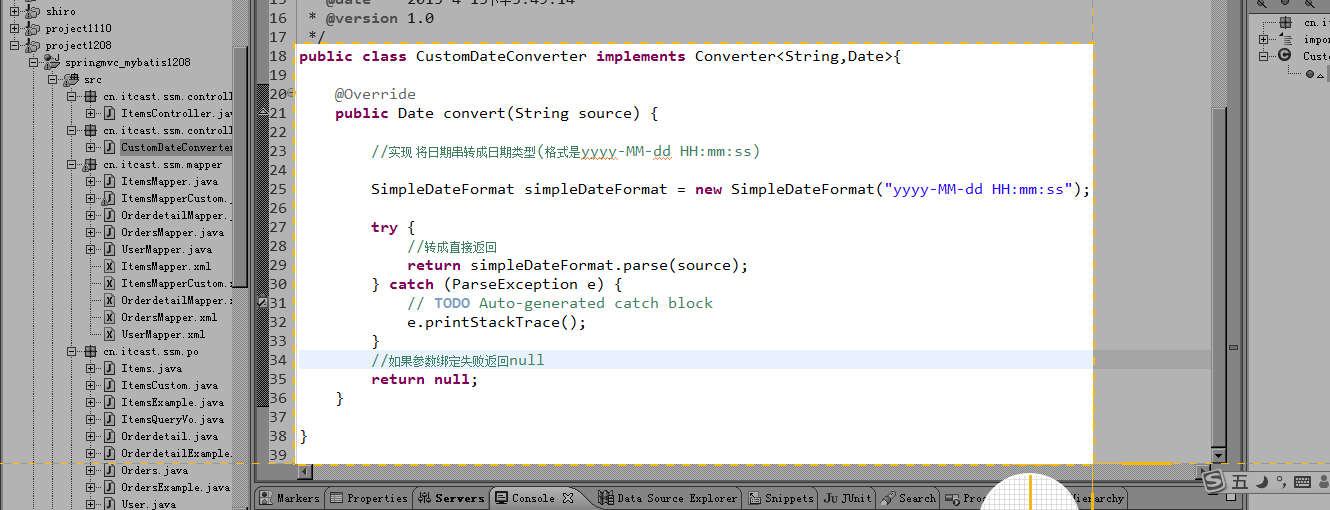
### 对日期类型的绑定（自定义参数）

对于controller形参中pojo对象，如果属性中有日期类型，需要自定义参数绑定

将请求日期数据串转换成日期类型，要转换的日期类型和pojo中日期属性的类型保持一致



所以自定义参数绑定将日期串转成java.util.Date类型





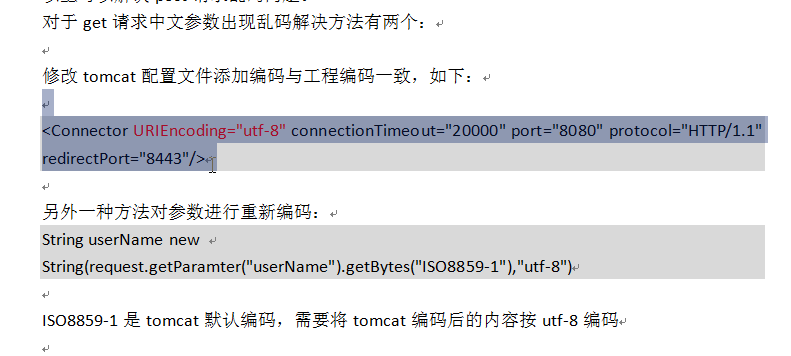
## 问题

会出现中文乱码的问题

Post乱码

在web.xml添加过滤器

Get乱码



## Springmvc和Struts2的区别

1. springmvc基于方法开发，struts2基于类开发的。

Springmvc将url和controller方法映射。映射成功后springmvc生成一个Handler对象，对象中只包含一个method。方法执行结束，形参数据销毁

Springmvc的controller开发类似service开发

1. springmvc可以进行单例开发，并且建议使用单例开发，struts2通过类的成员变量接收参数，无法使用单例，只能使用多例。
2. struts2经过测试，struts2速度慢，在于使用struts标签，如果使用struts建议使用jstl。

# Springmvc第二天 高级知识

复习：

Springmvc框架

DispatcherServlet前端控制器：接收request,进行response

HandlerMapping处理器映射器：根据url查找Handler（可以通过xml配置方式，注解方式）

HandleAdapterr处理器设配器：根据具体规则去执行Handler，编写Handler时需要按照HandlerAdapter的要求去编写

Handler处理器（后端控制器）：需要程序员去编写，常用注解开发方式

Handler处理器执行后结果是ModelAndView，具体开发时Handler返回值类型包括：ModelAndView、String（逻辑视图名）、void（通过在Handler形参中添加request和response,类似原始servlet开发：注意：可以通过指定response响应的结果类型实现json数据输出）

View resolver视图：根据逻辑视图名来生成真正的视图（在springmvc中使用view对象

表示）

View视图：jsp页面，仅是数据展示，没有业务逻辑





@controller注解必须要加，作用标识类是一个Handler处理器

@requestMapping注解必须要加，作用：

1. 对url和Handler的方法进行映射
2. 可以窄化请求映射，设置Handler的根路径，url就是根路径+子路径
3. 可以限制http请求的方法

映射成功后，springmvc框架会生成一个Handler对象，对象中只包括一个成功的method

## 注解开发中参数绑定

将request请求过来的key/value的数据（理解成一个串），听过转换（餐阿叔绑定的一部分）将key/value串将转换后的结果传给形参（整个参数绑定过程）

Springmvc所支持参数绑定

1、默认支持很多类型，HttpServletRequest、reponse、session、model/modelMap（将模型数据填充到request域）

2、支持简单数据类型，整数、字符串、日期

只要保证request请求的参数名和形参名称一致，自动绑定成功

如果request请求的参数名和形参名称不一致，可以使用@RequestParam（指定request请求的参数名），@RequestParam加在形参的前边

1. 支持pojo

只要保证request请求的参数名称和pojo中属性名称一致。自动将request请求的参数设置到pojo的属性中

注意：形参中即有pojo类型又有简单类型，参数绑定互不影响。

## 自定义参数绑定

日期类型数据绑定

日期类型绑定自定义：

定义的Converter<源类型，目标类型>接口实现类，比如：

Converter<String,Date>表示：将请求的日期数据串转换成java中的日期类型。

注意：要转换的目标类型一定和接受的pojo中的属性类型一致

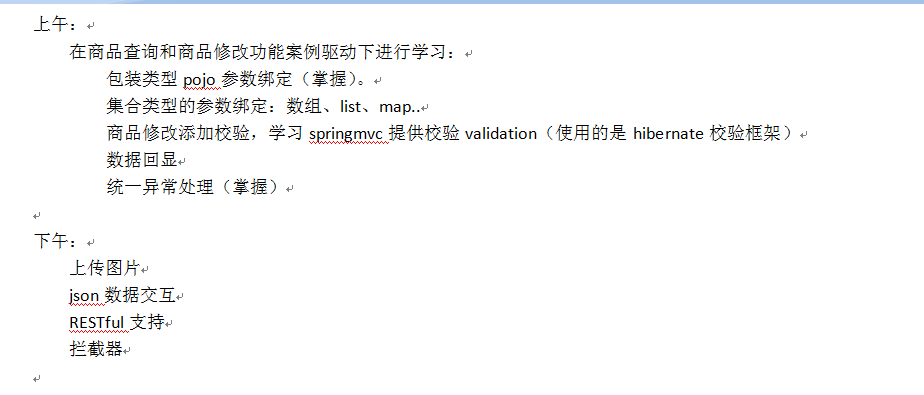
将定义的Converter实现类注入到处理器适配器中。



Springmvc面向方法开发的，struts2面向类开发。

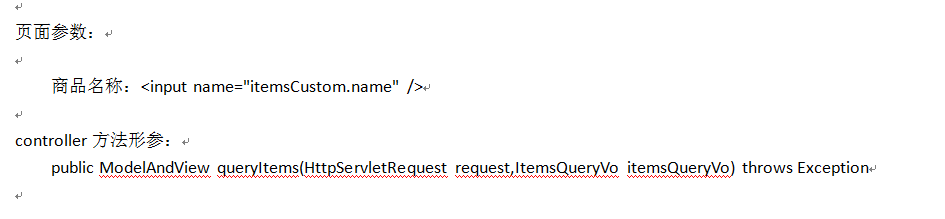
Springmvc可以单例开发，struts2是多例开发

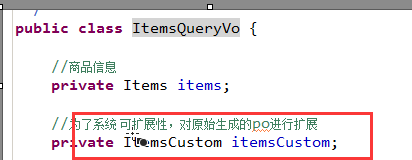
# 今天内容





# 页面参数和controller方法形参定义





## 数组绑定

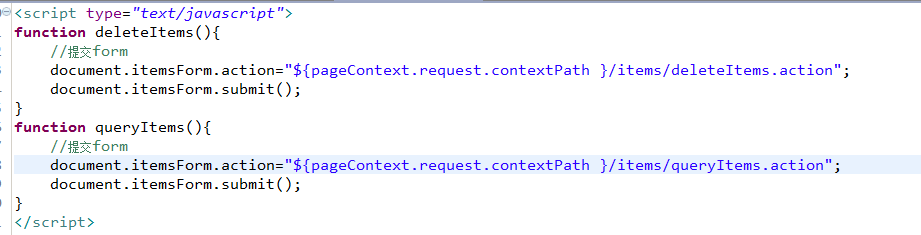
### 需求

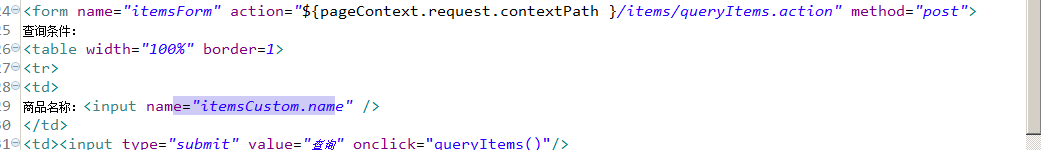
商品批量删除，用户在页面选择多个商品，批量删除

### 表现层实现

关键：将页面选择（多选）的商品id，传到controller方法的形参，方法形参使用数组来接收页面请求的多个id

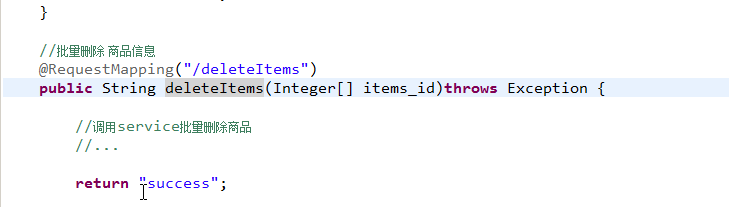
页面的定义







Controller的定义



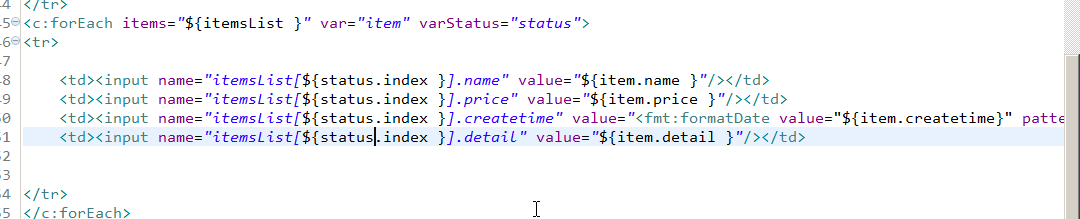
# List绑定

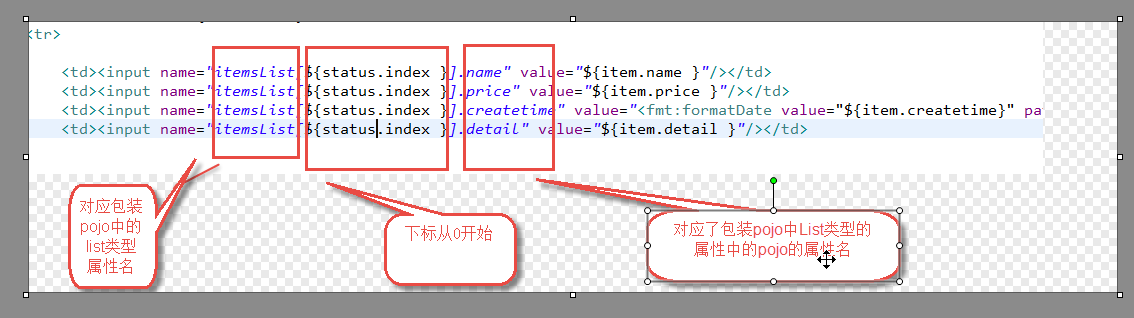
需求：

通常在需要批量提交数据时，将提交的数据绑定到list<pojo>中，比如：成绩的录入（录入多门课成绩，批量提交）

本例子需求：批量商品修改，在页面输入多个商品信息，将多个商品信息提交到controller方法中

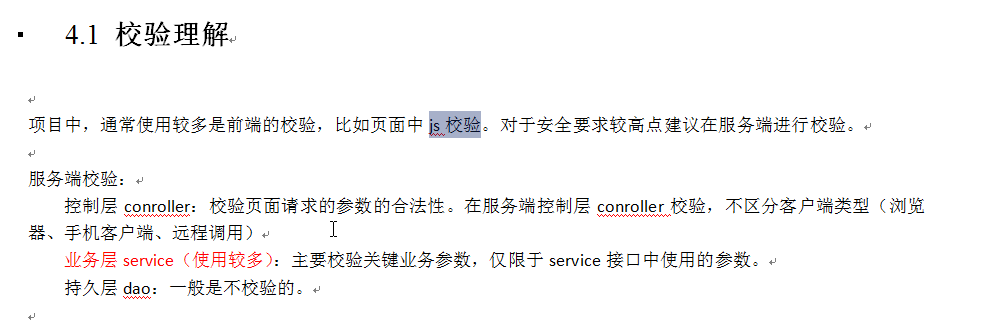


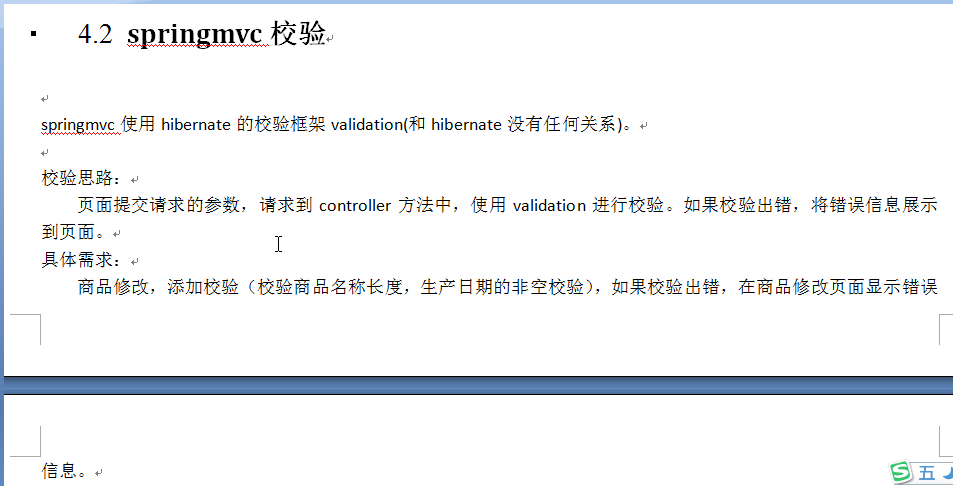


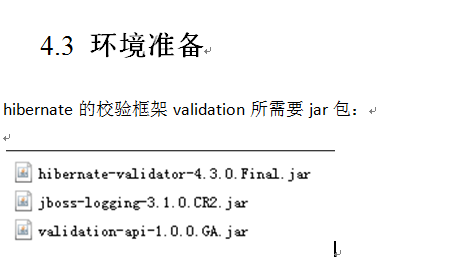




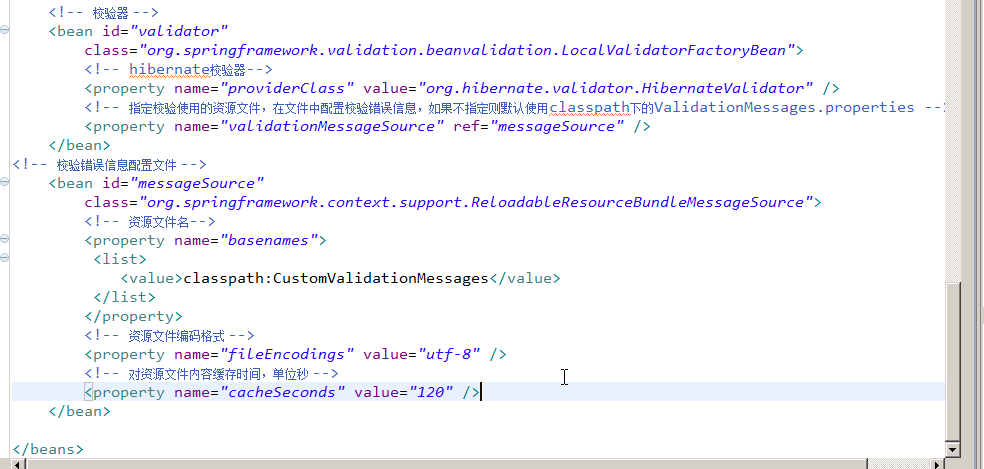
# 校验

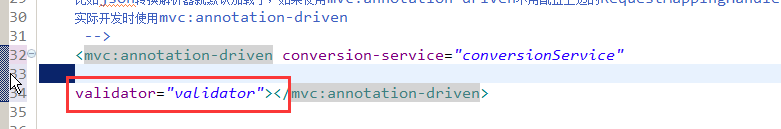




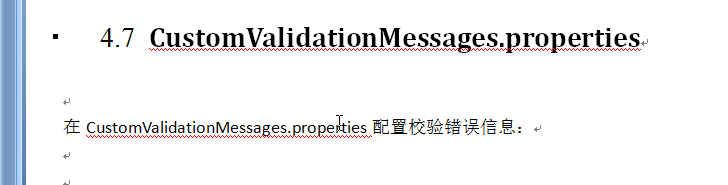


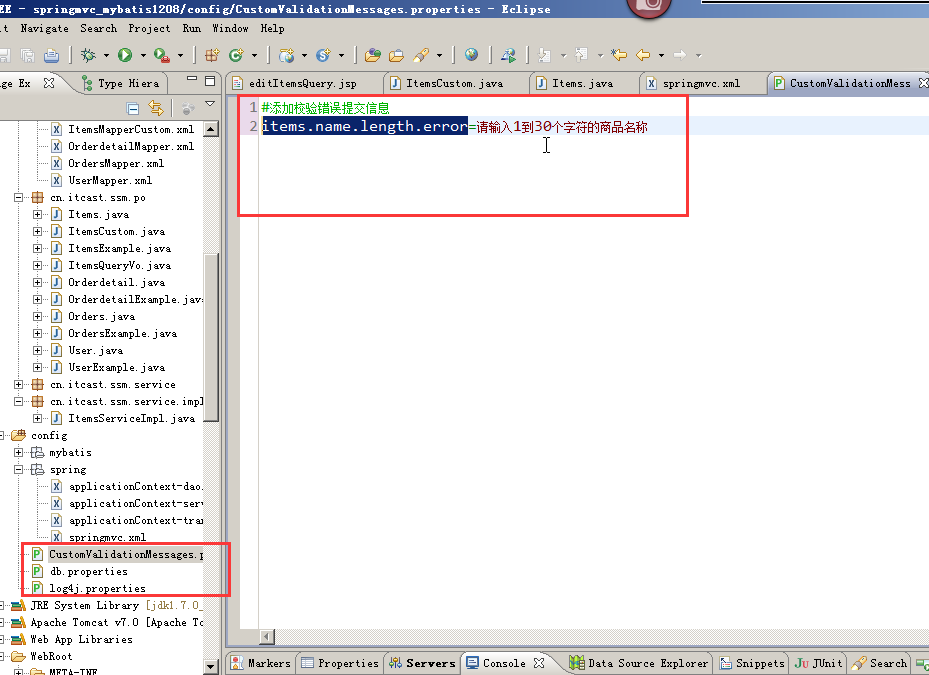
## 校验器的配置

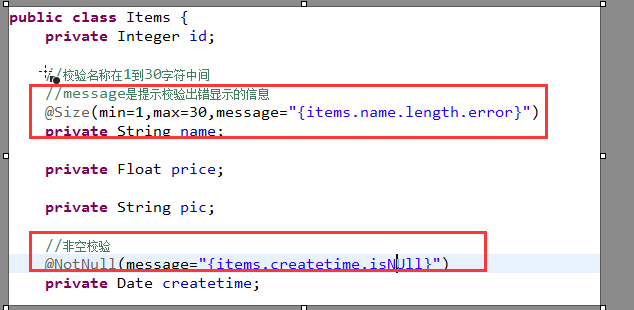




## 在pojo中添加校验规则





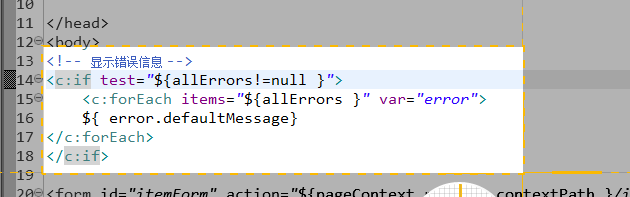


## 捕获校验信息





## 页面显示错误



## 分组校验

### 需求：

在pojo中定义校验规则，而pojo是被多个controller所公用，当不同的controller对同一个但是每个controller方法需要不同的校验。

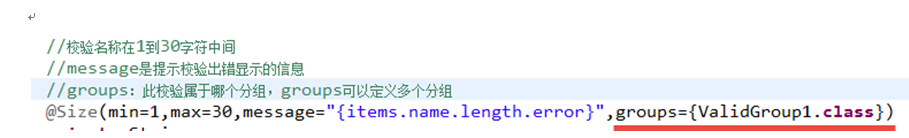
解决方法

定义多个校验分组（其实是一个java接口）

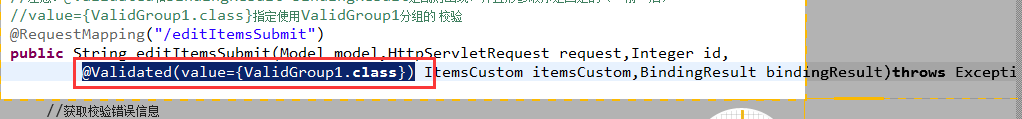
每个controller方法使用不同的校验分组



## 在校验规则中添加分组



## 在controller中使用指定分组的参数



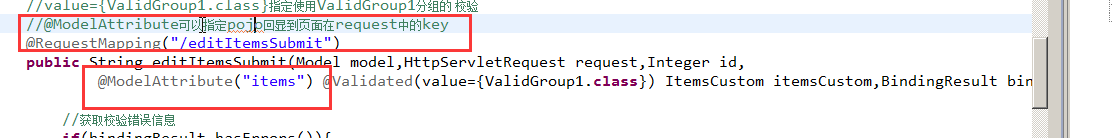
# 数据回显

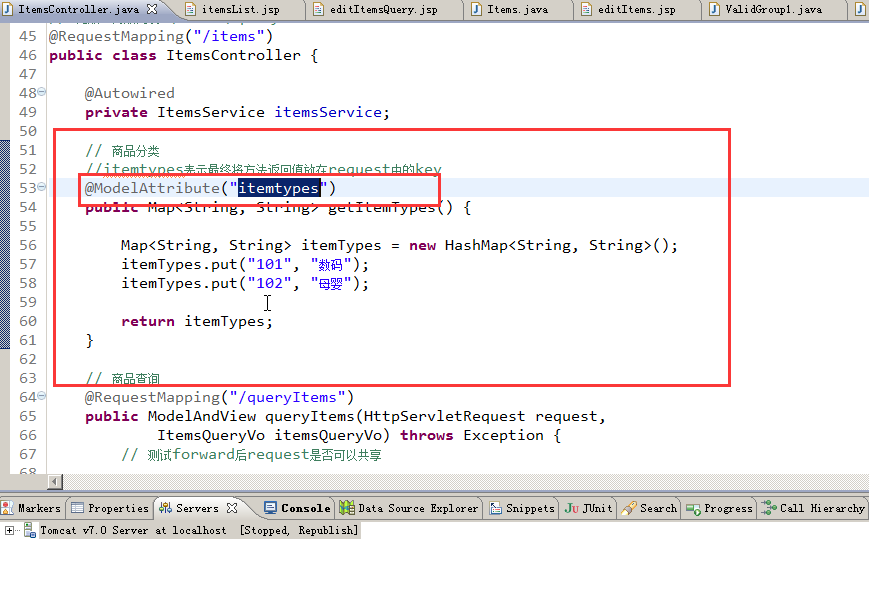
## 什么是数据回显

提交后，如果出现错误，将刚才提交的数据回显到刚才提交的页面

## 数据回显的方法

1. springmvc默认对pojo数据进行回显
   1. pojo数据传入

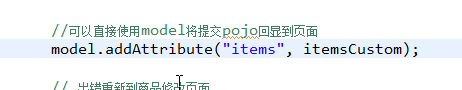


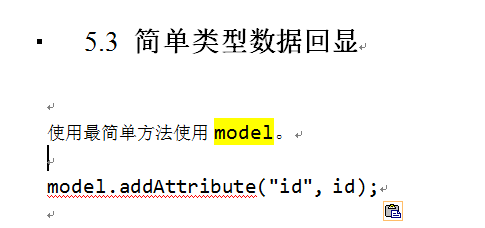


通过url是没有办法返回的，页面上可以得到itemtypes的数据



## 使用最简单的方法model





# 异常处理

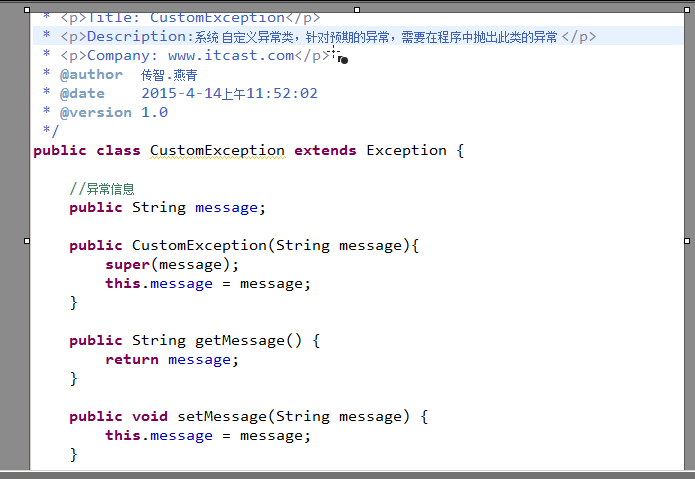
## 全局异常处理器

系统中异常包括两类，预期异常和运行时异常RuntimeException，前者通过捕获异常从而获取异常信息、后者主要通过规范代码开发、测试通过手段减少运行时异常的发生

Springmvc提供全局异常处理器（一个系统只有一个异常处理器）进行统一异常处理

## 自定义异常类

对不同的异常类型定义异常类，继承Exception



## 全局异常处理器

思路：

系统遇到异常，在程序中手动抛出，dao抛给service、service给controller、controller抛给前端控制器，前端控制器调用全局异常处理器。

全局异常处理器处理思路

解析出异常类型

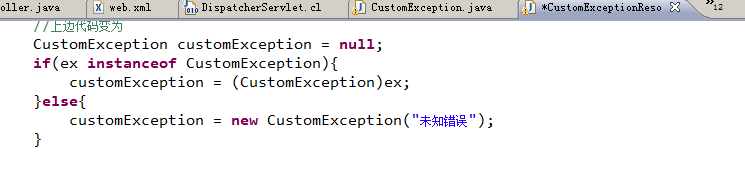
如果该异常类型是系统自定义的异常，直接取出异常信息，在错误页面展示。

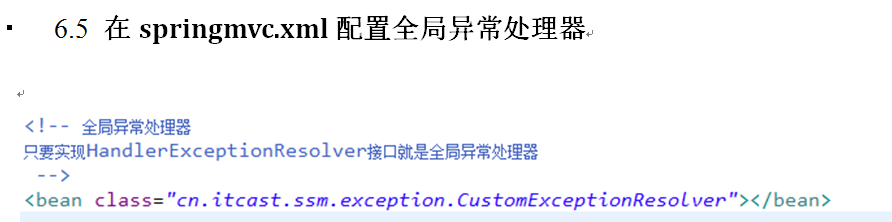
如果该异常类型不是系统自定义的异常，构造一个自定义的异常类型（信息为“未知错误”）。

Springmvc提供了一个HandlerExceptionResolver

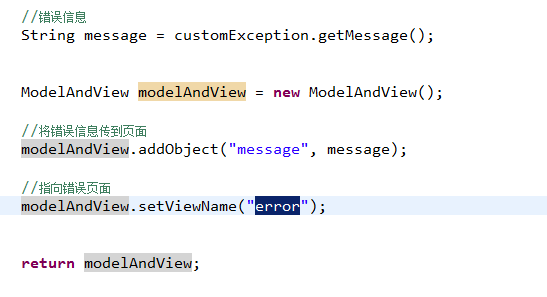




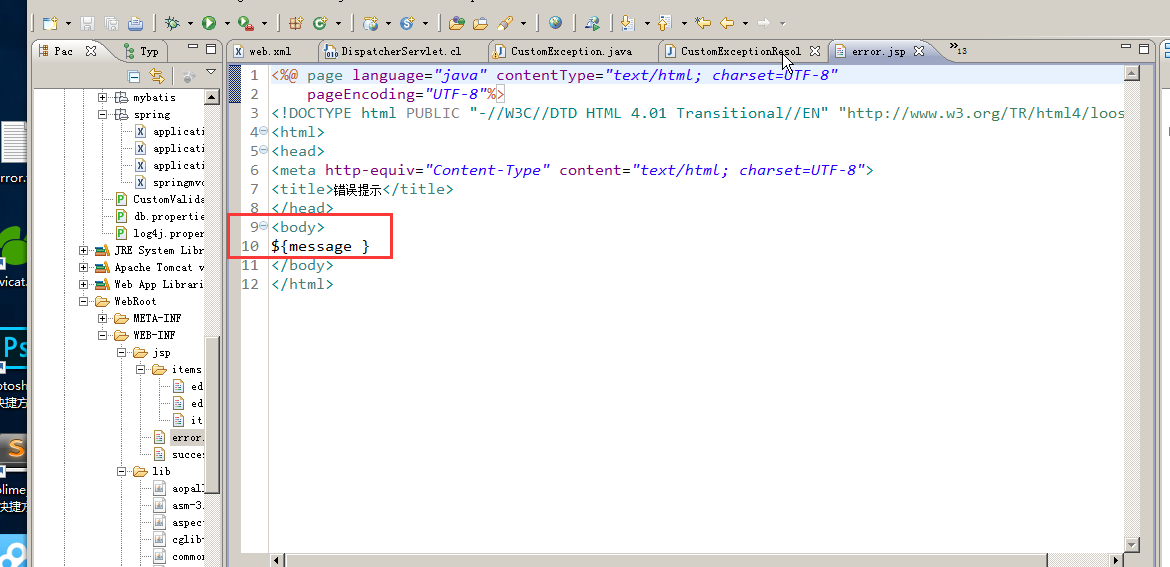




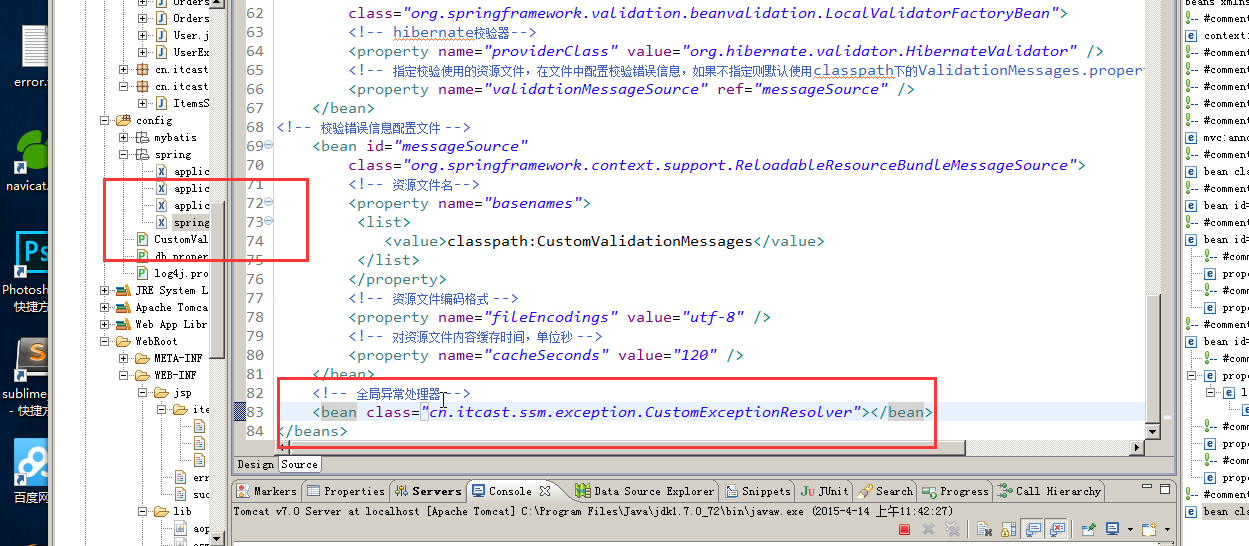
异常测试



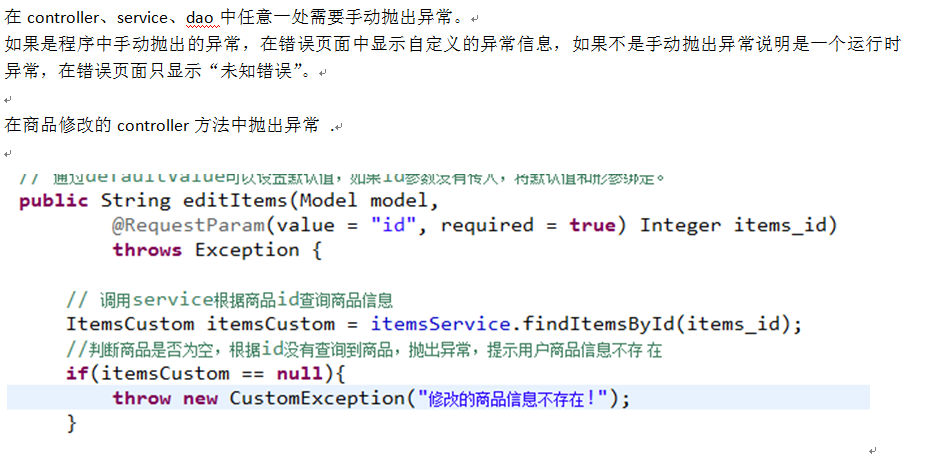
## 页面展示

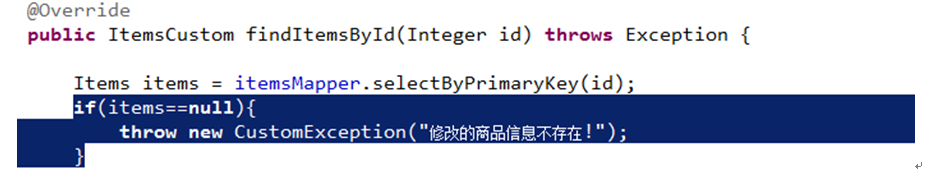


## 在配置文件中配置



## 异常测试





如果与业务功能相关的异常，建议在serivce中抛出

与业务功能没有关系的异常，建议在controller中抛出。

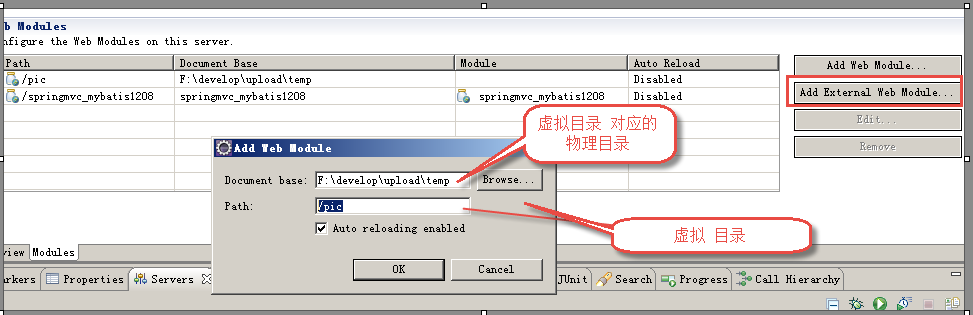
# 上传图片

1. 需求

在修改商品页面，添加上传商品图片的功能

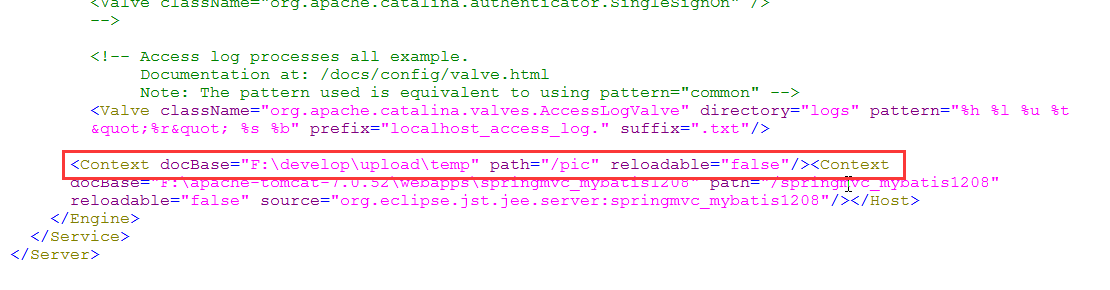
2、

3、创建一个图片的虚拟目录，存储图片（通过图形界面的方式）



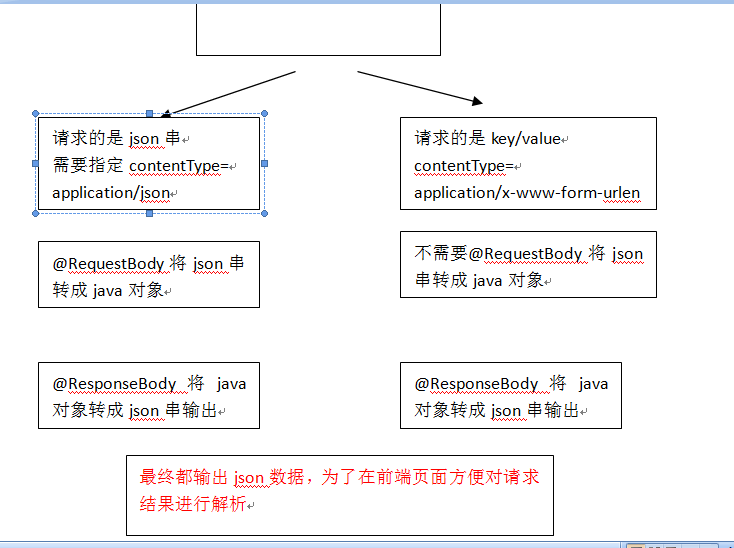
也可以直接修改tomcat的配置文件

在conf/server.xml文件，添加虚拟目录

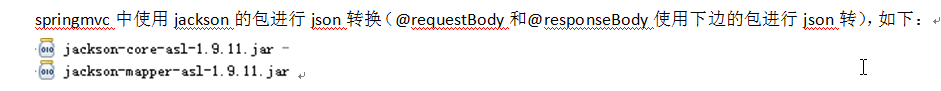


注意：在图片虚拟目录中，一定将图片目录分级创建（提高i/o性能），一般我们采用（年、月、日）方法进行创建

# Json转换



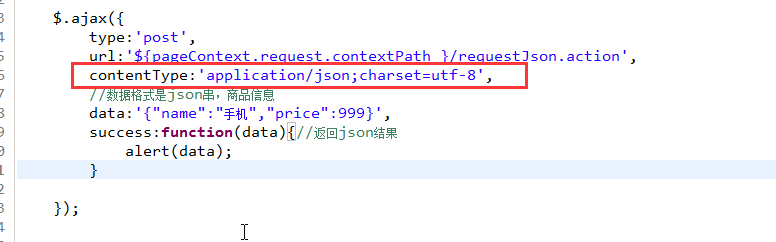
## 环境准备

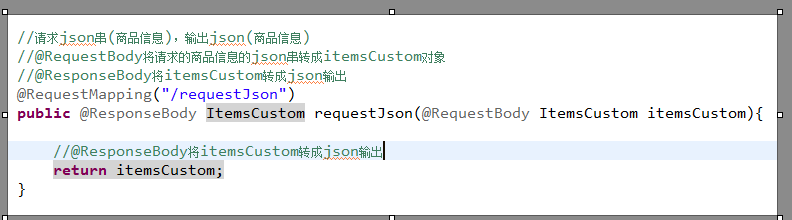




## Json交互测试

### 输入json串，输出是json串





### 输入key/value，输出json串



