# 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。

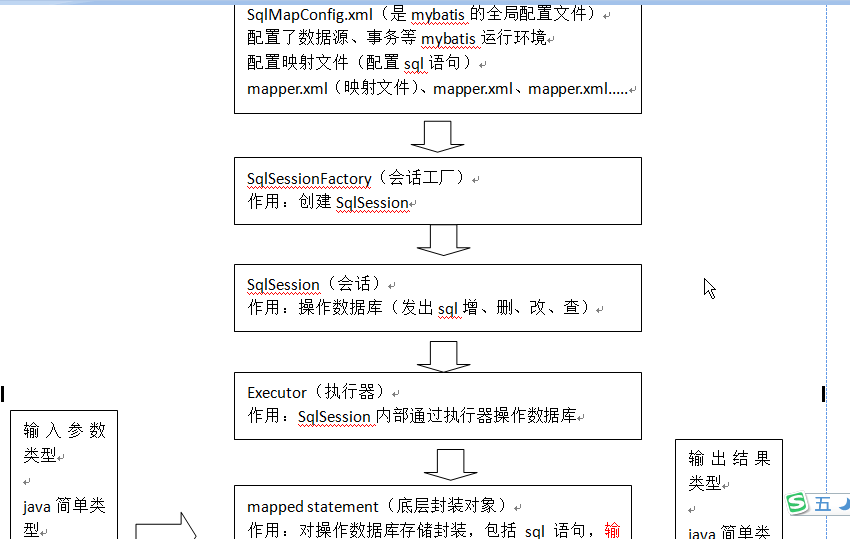
# Mybatis框架

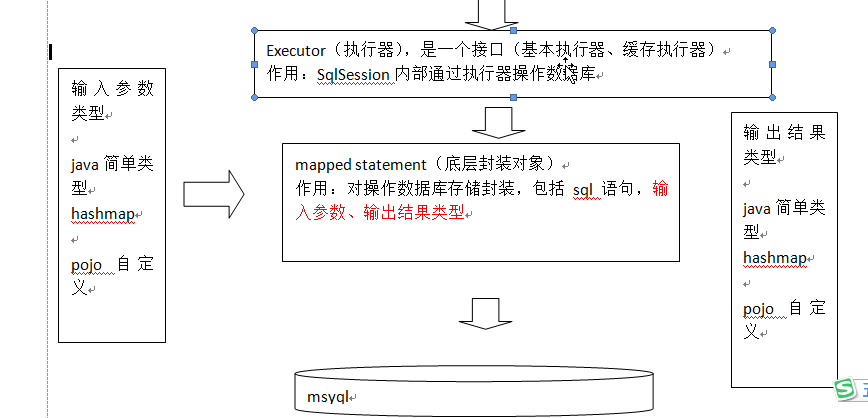
Mybatis是一个持久层的框架，是apache下的顶级项目

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

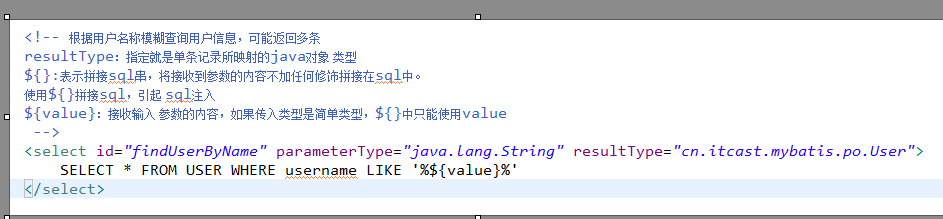
Mybatis让程序员主要精力放在sql中，通过mybatis提供的映射放射，自由的灵活生成（半自动化，大部分需要程序员编写sql）满足需要sql语句。

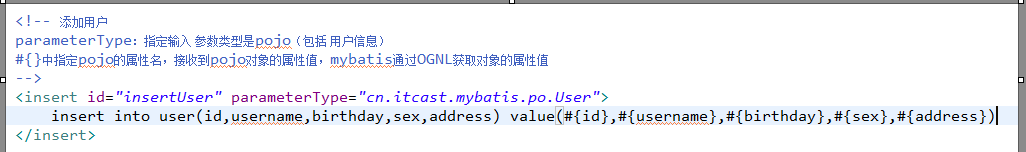
Mybatis可以将向preparedStatement中的输入参数自动进行输入映射，将查询结果集灵活映射成java对象（输出映射）

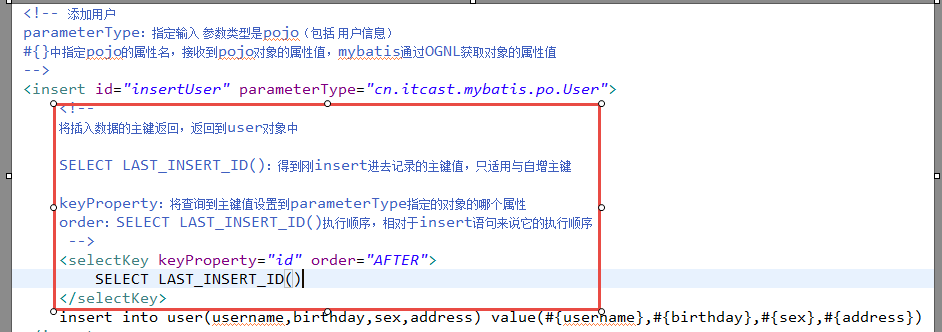


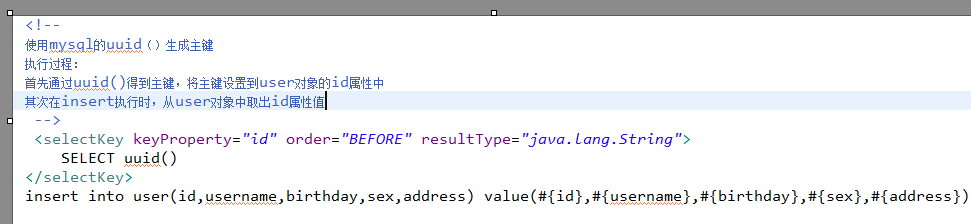


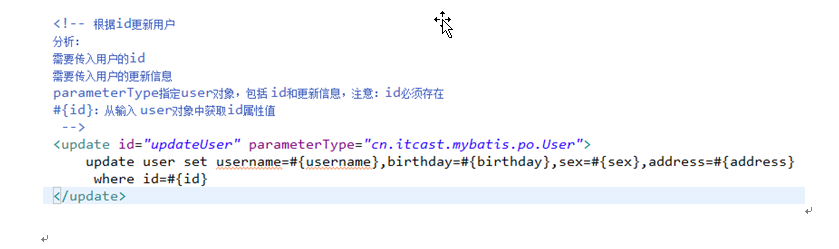


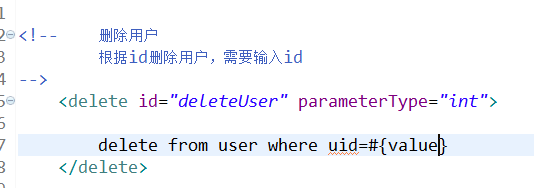


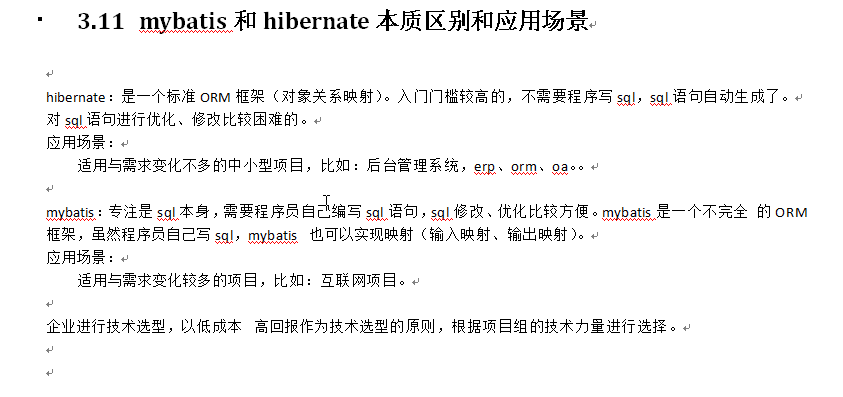












SqlSession是线程不安全的，在SqlSession实现类中除了有接口中的方法（操作数据库的方法）还有数据域属性

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））

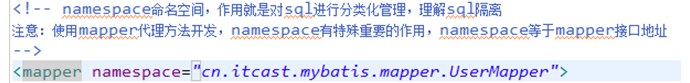
## 思路

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：

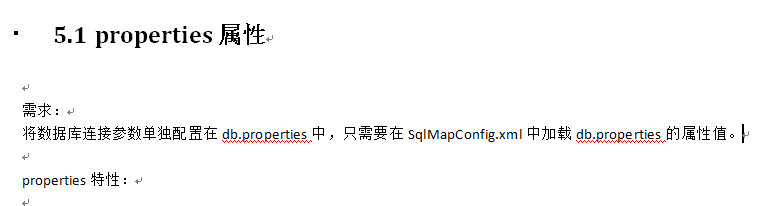
1. 在mapper.xml中namespace等于mapper接口地址



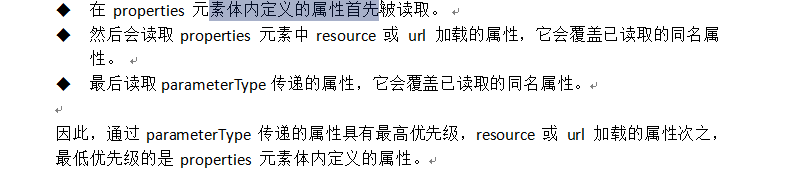
1. mapper.java接口接口中方法名和mapper.xml中statement的id一致
2. mapper.java接口中的方法输入类型和mapper.xml中statement的parameterType指定的类型一致
3. mapper.java接口中方法的返回值类型和mapper.xml中statement的resultType指定的类型一致

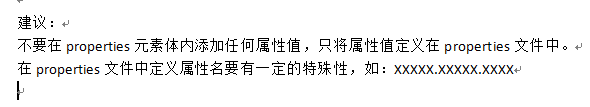


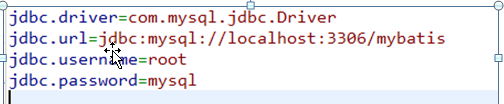
# SqlMapConfig.xml











## 5.2 typeAliases（别名）重点

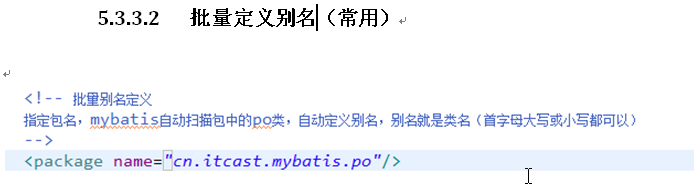
## 5.2.1需求

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

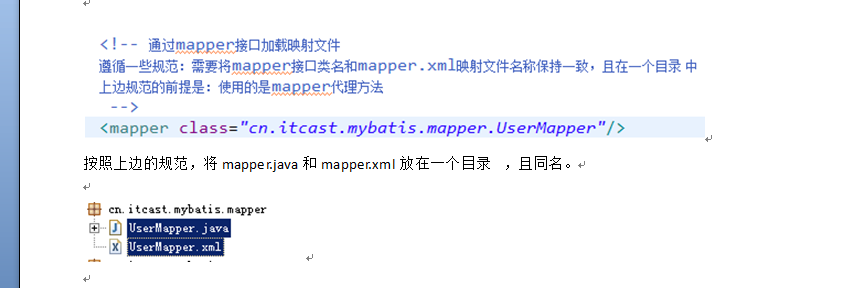
## 单个别名的定义







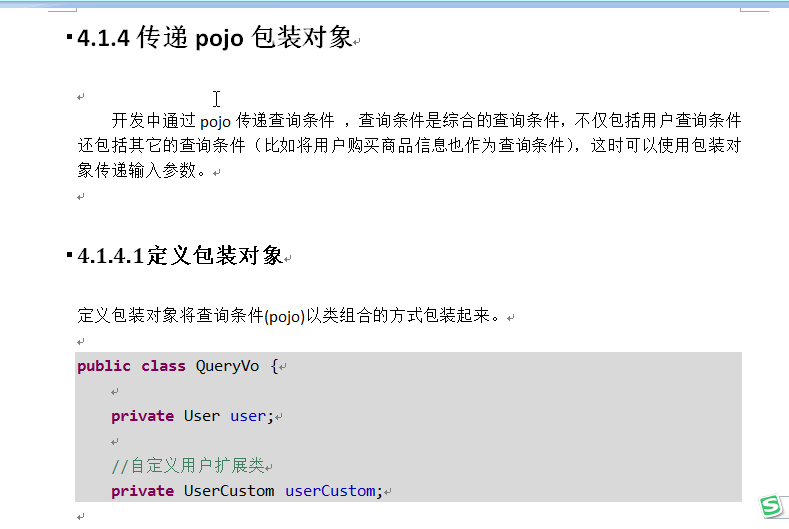
## 单个加载

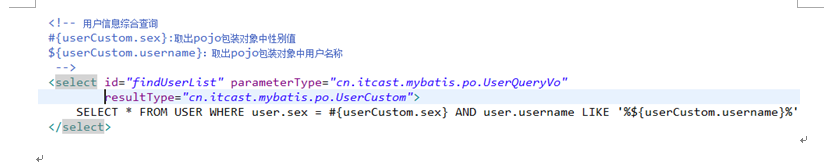


## 批量加载

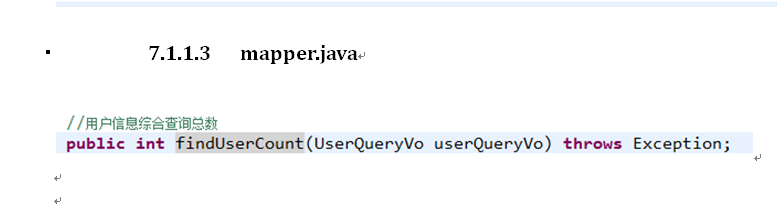


## 多domin的封装类型

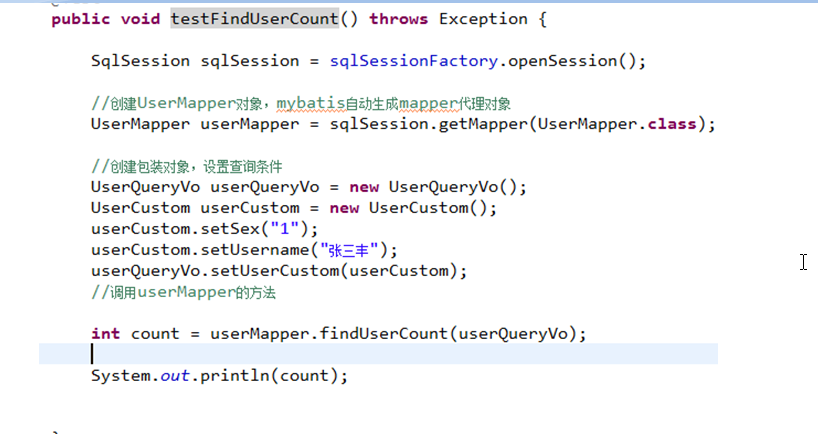






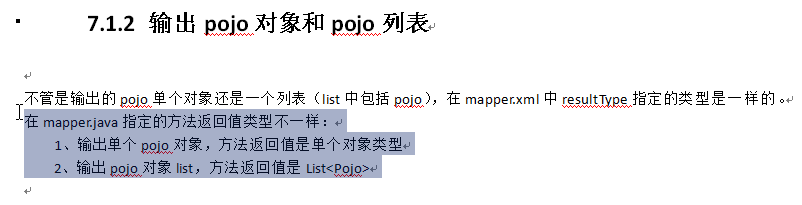


## 测试代码



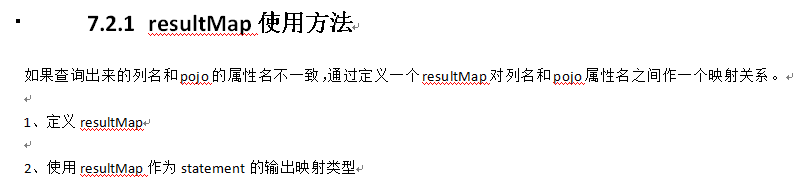
# 小结

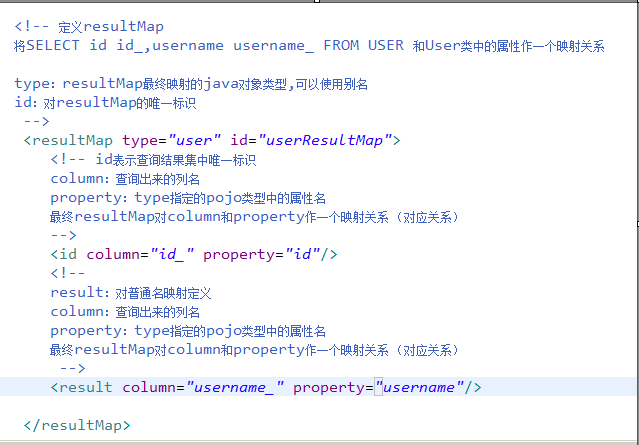
查询出来的结果集只有一行且一列，可以使用简单类型进行映射

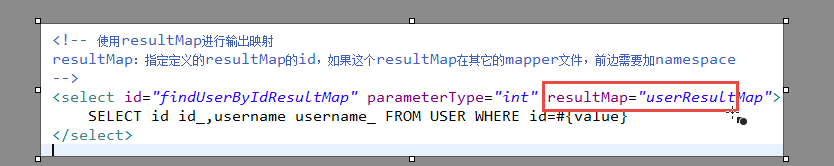


# resultMap

mybatis中使用resultMap完成高级输出结果映射



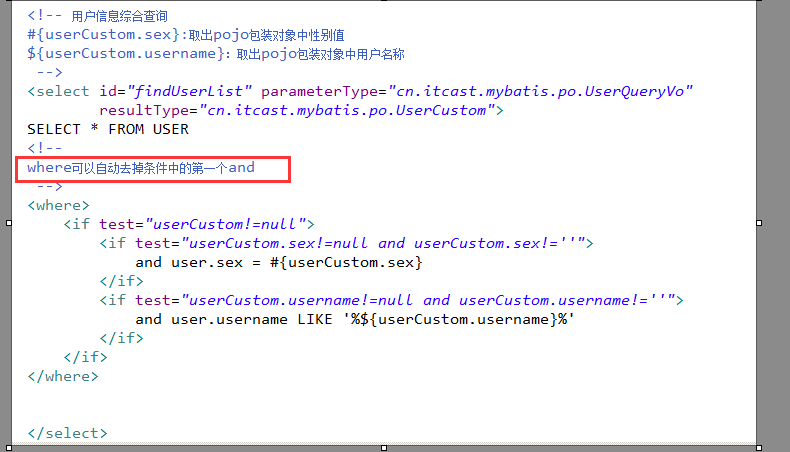




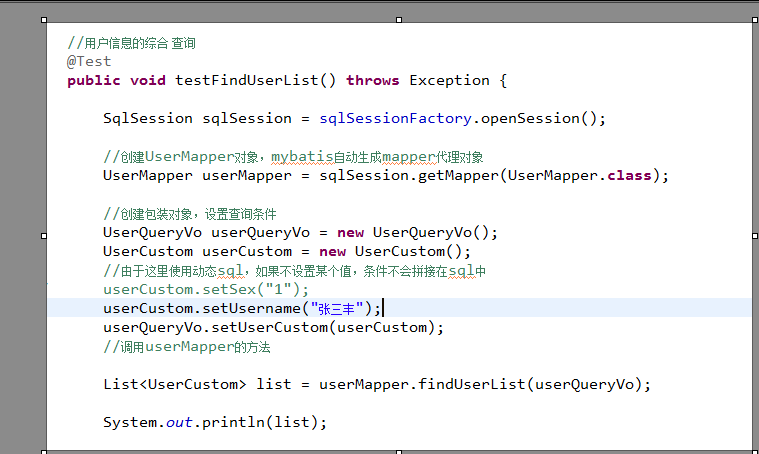
# 动态sql

## 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

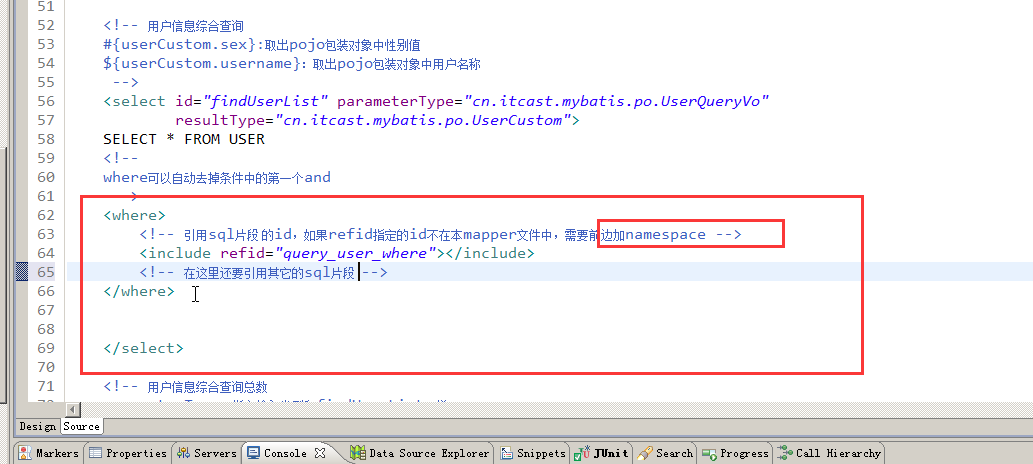
## 需求

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



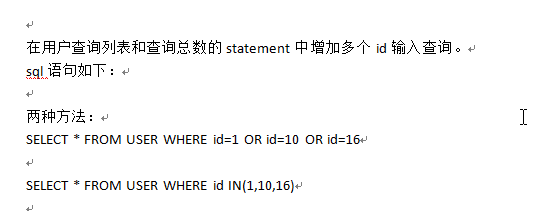
## 引用sql片段

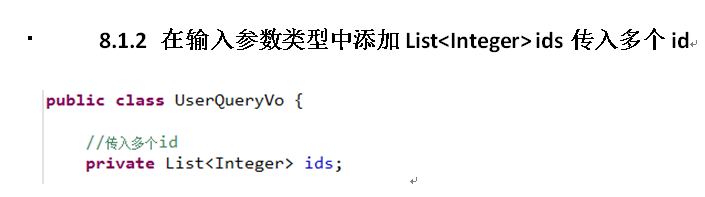


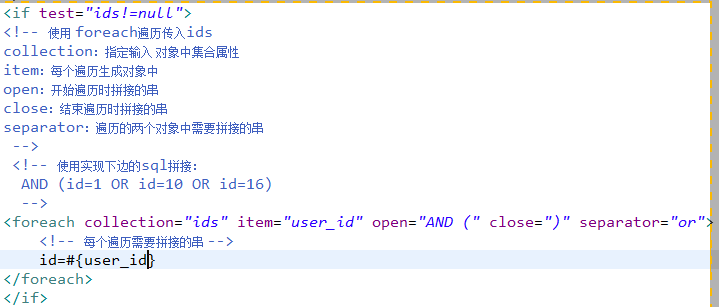
# Foreach标签

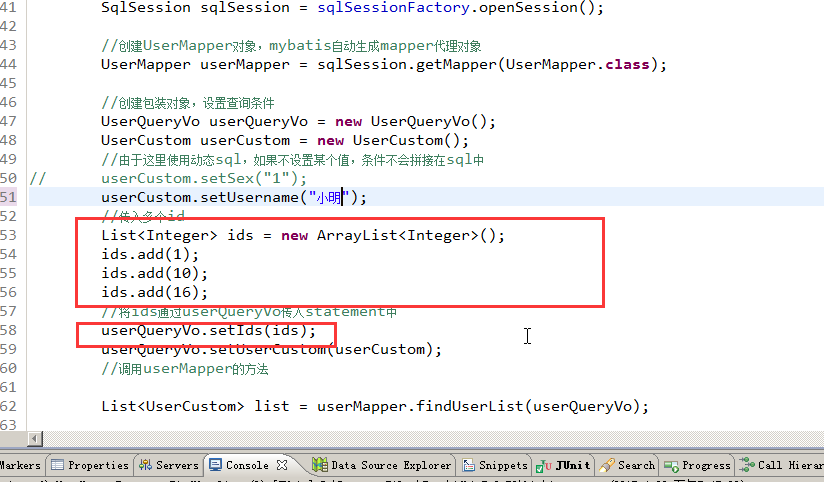
向sql传递数组或list，mybatis使用foreach解析

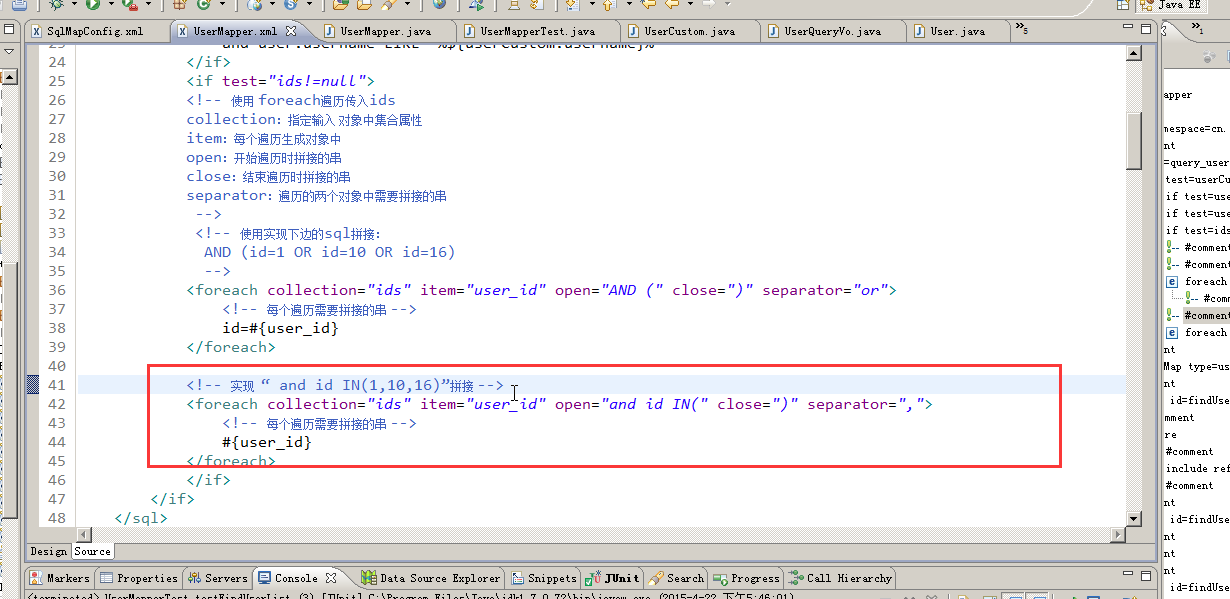
## 需求











# 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

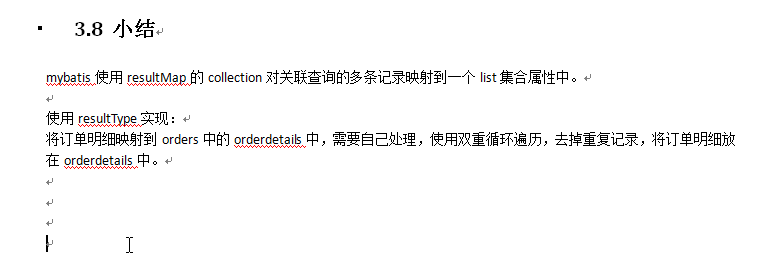
实现一对一查询

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。

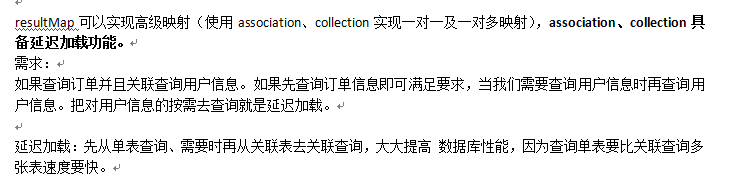
resultMap:需要单独定义resultMap，实现有点麻烦，如果对查询结果有特殊的要求，使用ResultMap可以完成将关联查询映射pojo的属性中

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



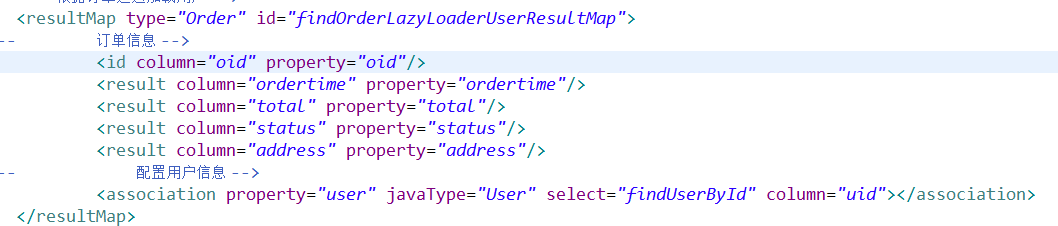
6.2使用assocation实现延迟加载

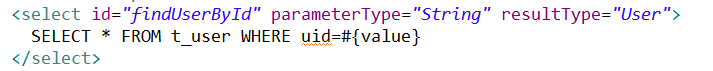
需求：查询订单并且关联查询用户

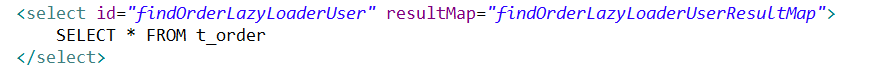
mapper.xml

需要定义两个mapper.xml

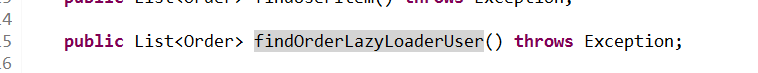
1. 先查询订单信息
2. 再查询用户信息

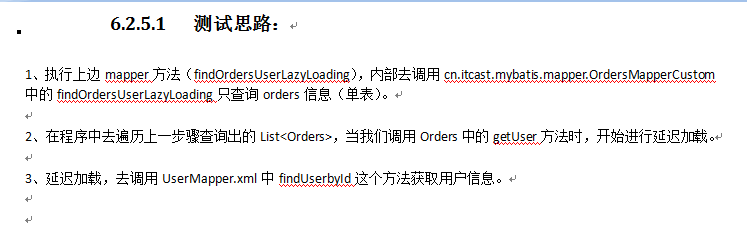




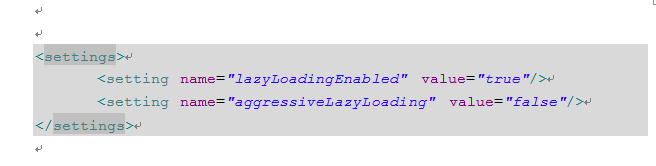


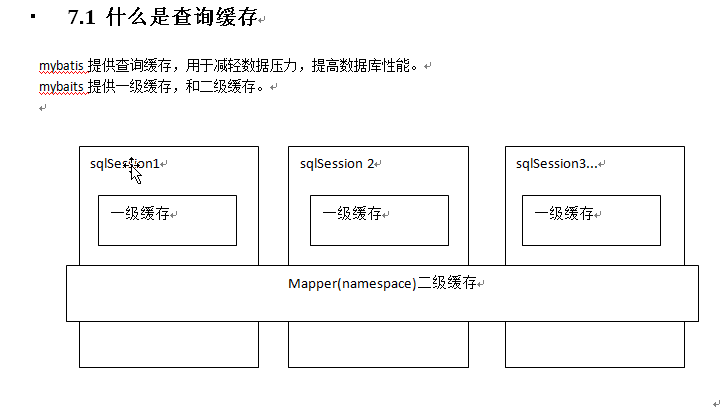
mapper.java





SqlMapConfig.xml把懒加载打开，并设置为不灵活



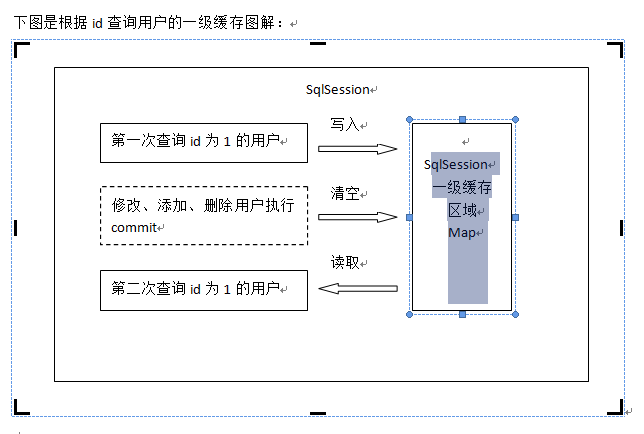


一级缓存是SqlSession级别的缓存。在操作数据库时需要构造sqlSession对象，在对象中有一个数据结构（HashMap）用于存储缓存数据。不同的SqlSession之间的缓存数据（HashMap）是互相不影响的。二级缓存是mapper级别的缓存，多个SqlSession去操作同一个Mapper的sql语句，多个SqlSession可以共存二级缓存，二级缓存是跨SqlSession的

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

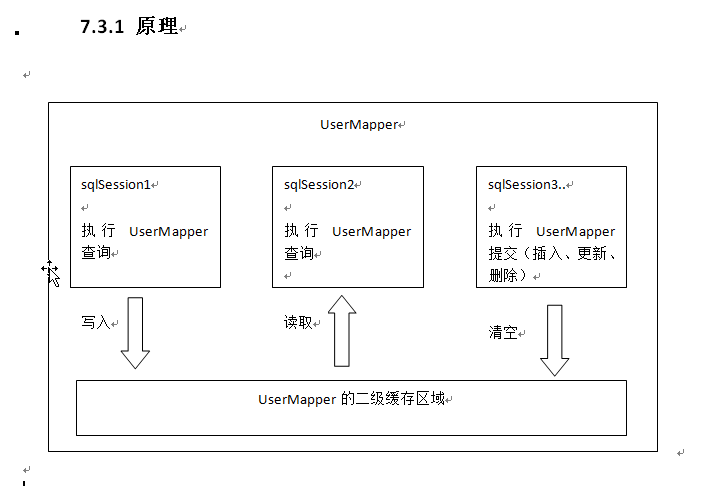
//第一次调用mapper的方法findUserById(1)

//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。

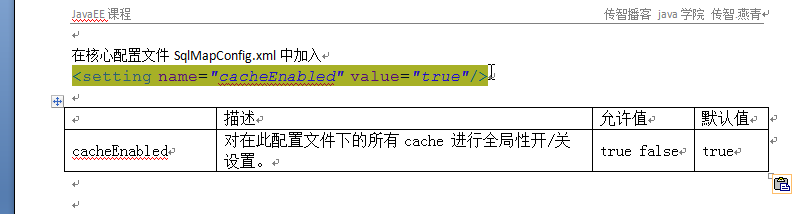
## 二级缓存和一级缓存的区别

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）

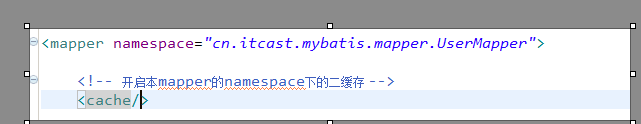
每一个namespace的mapper有一个二级缓存区域，两个mapper的namespace如果相同，这两个mapper执行sql查询到数据将存在相同的二级缓存区域中

开启二级缓存

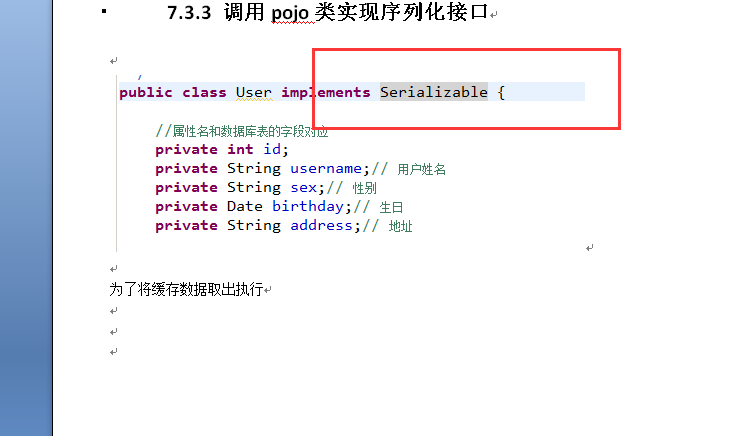
Mybatis的二级缓存是mapper范围级别，除了在SqlMapConfig.xml设置二级缓存的总开关，还要在具体的mapper.xml中开启二级缓存



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

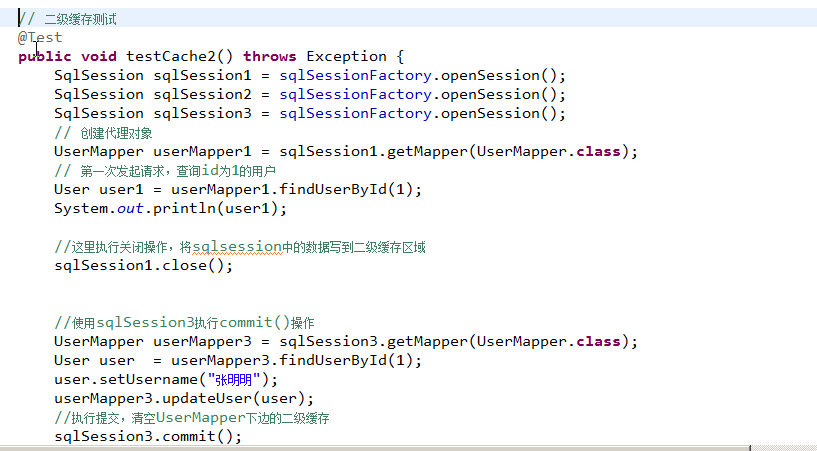


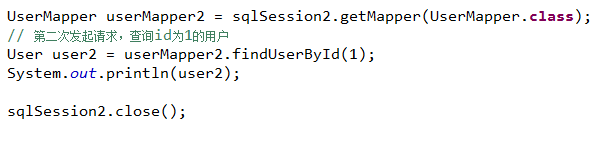
调用pojo类实现序列化接口

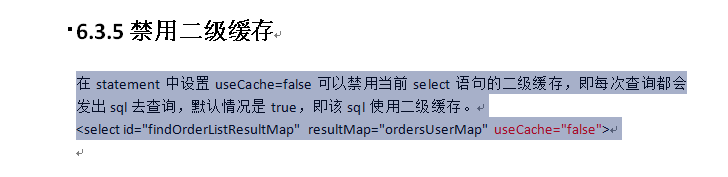


为了将缓存数据取出执行反序列化操作，因为二级缓存数据存储介质多种多样，不一定在内存（序列化为了存在在任意的地方）

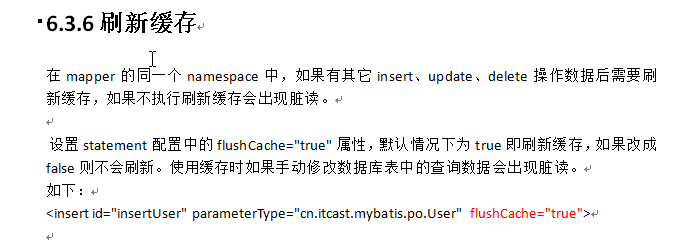
## 测试方法







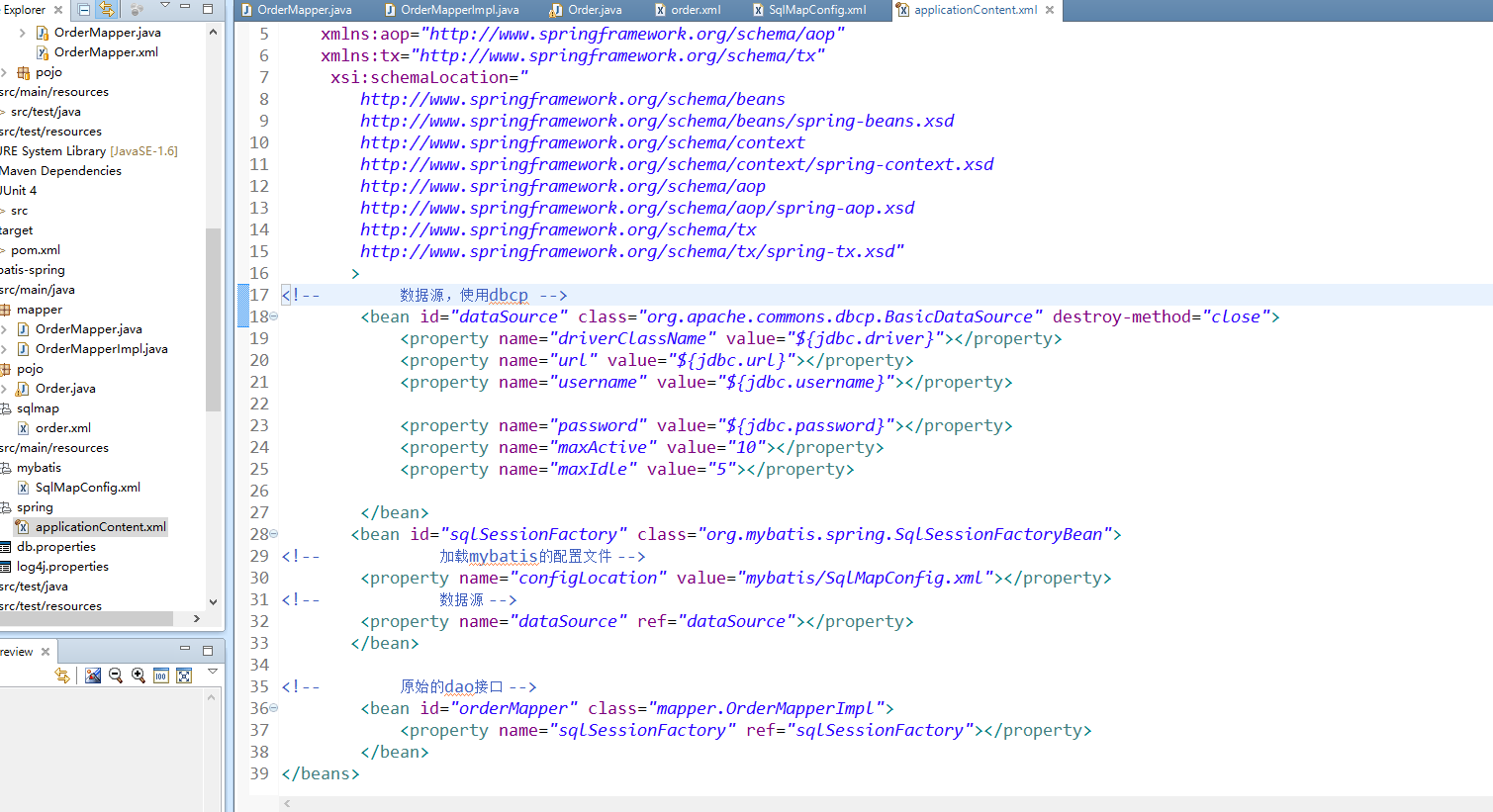
总结：针对每次查询都需要最新的数据sql，要设置成useCache=”false

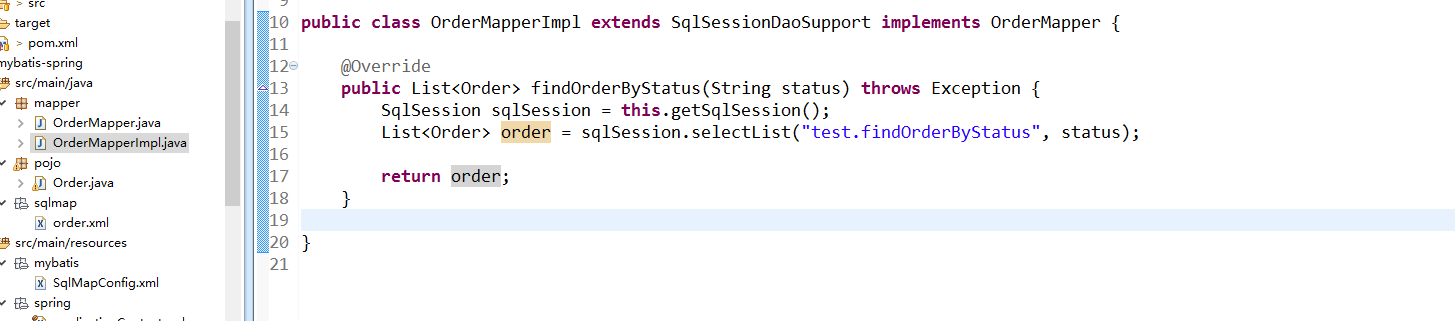


总结：一般执行完commit操作都需要刷新缓存，flushCache=true表示刷新缓存，这样可以避免数据库的脏读”

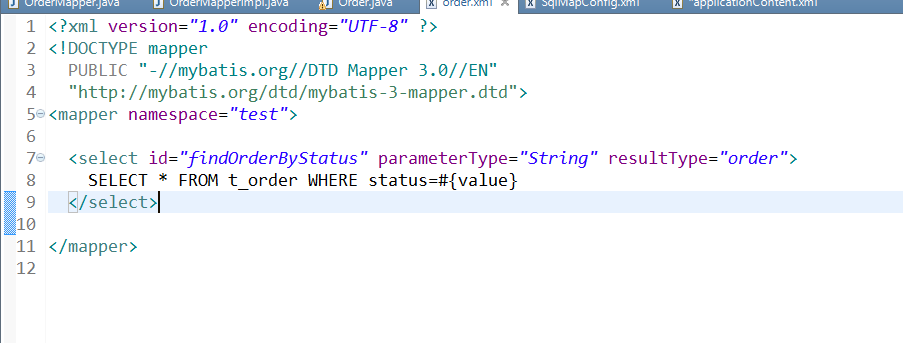
# Mybatis整合spring

## 原始dao与dao实现类

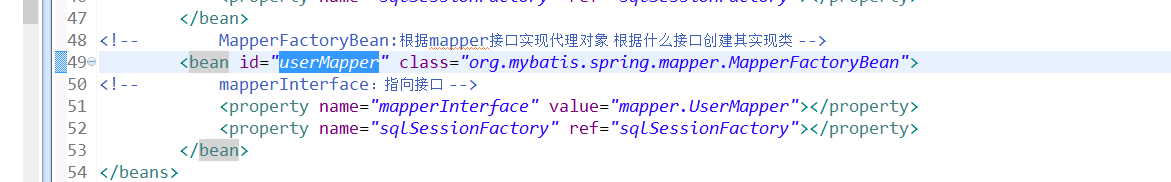


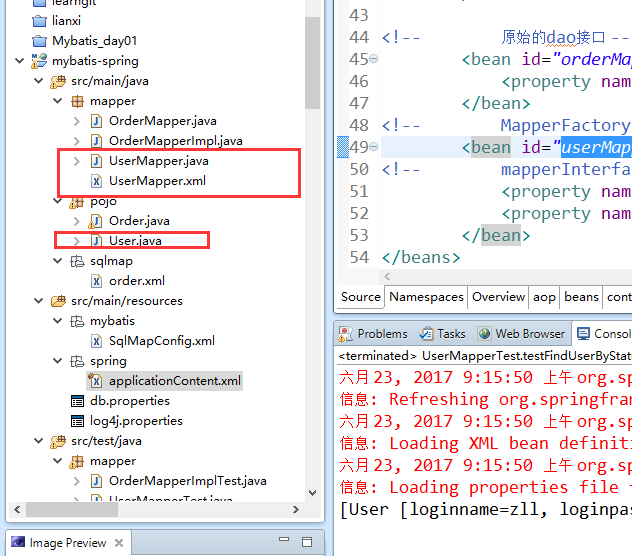


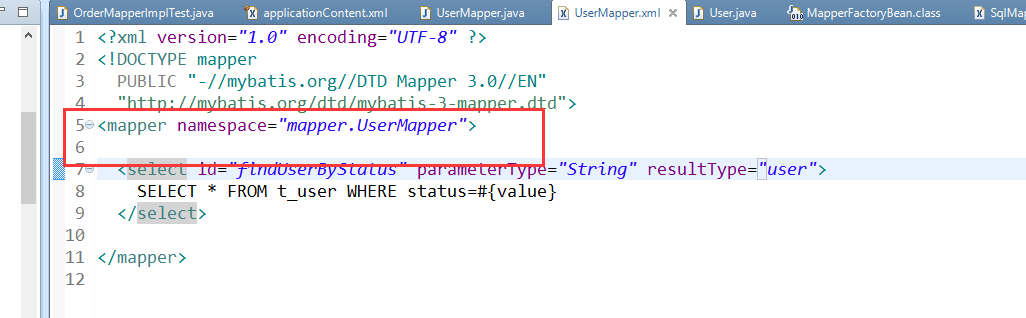


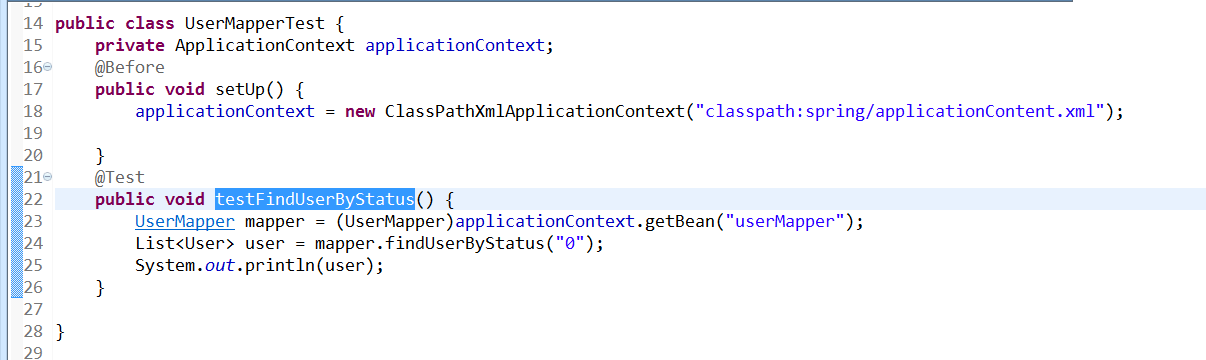


## Mapper代理









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

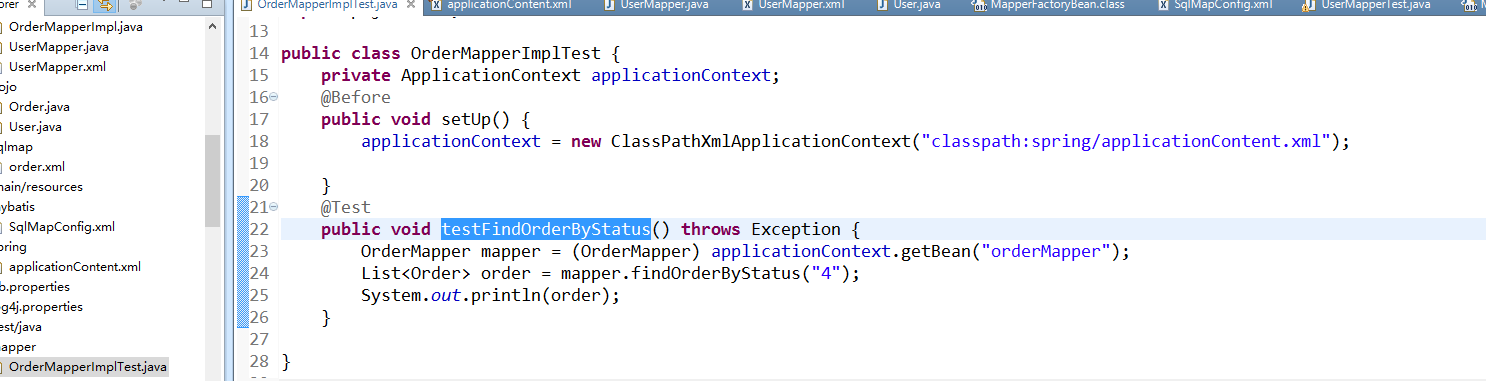
通过配置扫描器：MapperScannerConfigurer

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

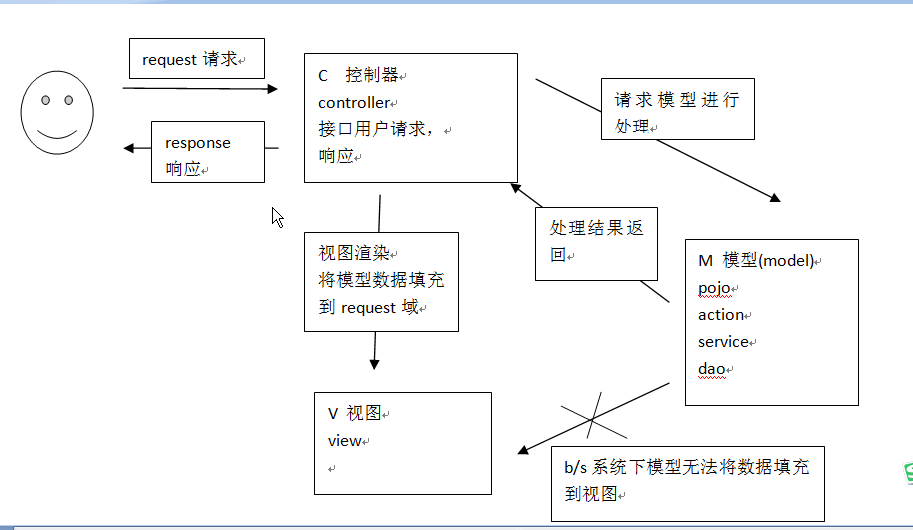
Json数据交互

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）

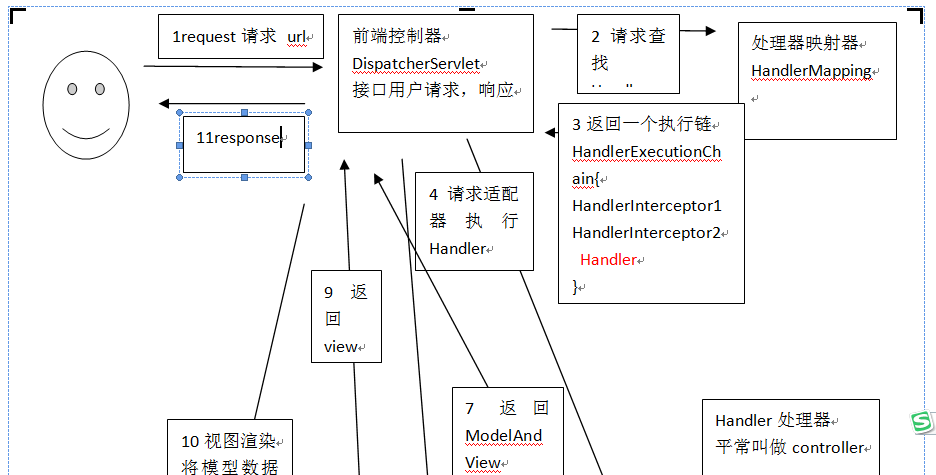
持久层的mapper都需要由spring进行管理

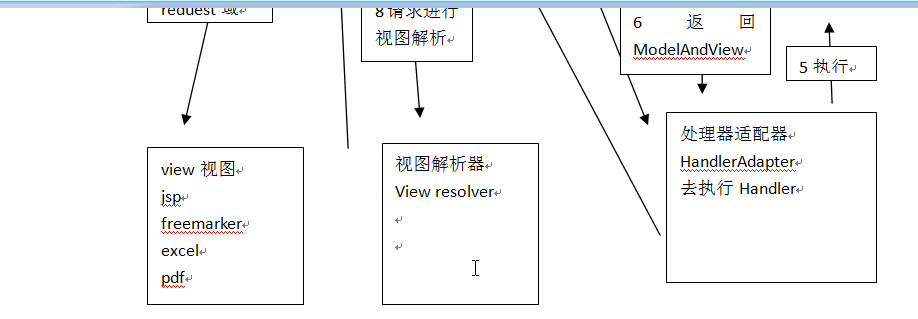
2整合环境

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

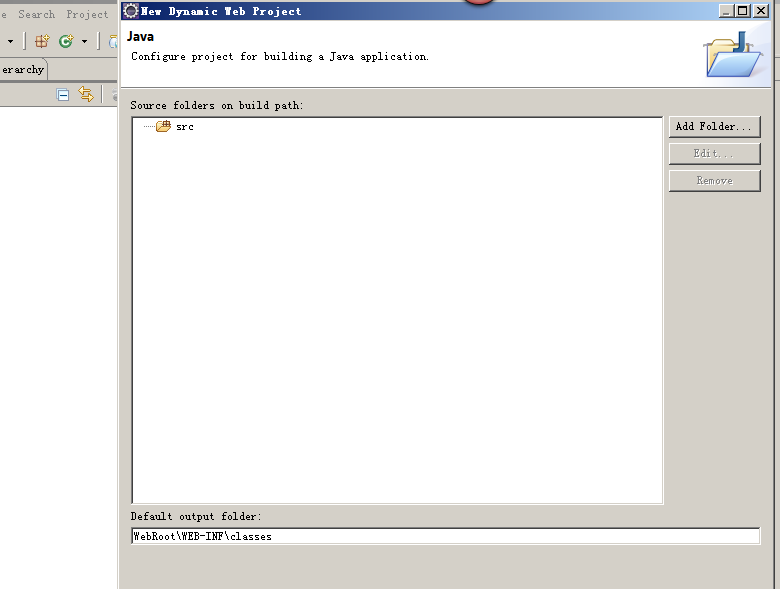
1. 处理器handler（需要程序员开发）
2. 视图解析器View resolver（不需要程序员开发

作用：进行试图解析，根据逻辑视图名解析成真正的视图（view）

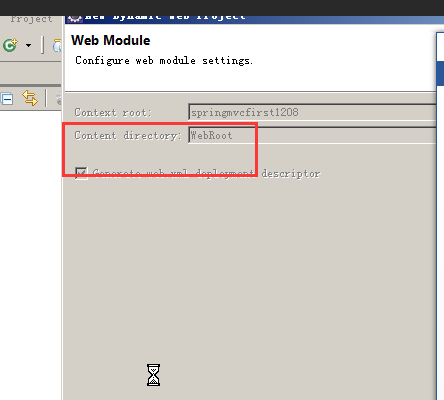
1. 视图View（需要程序员开发）

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

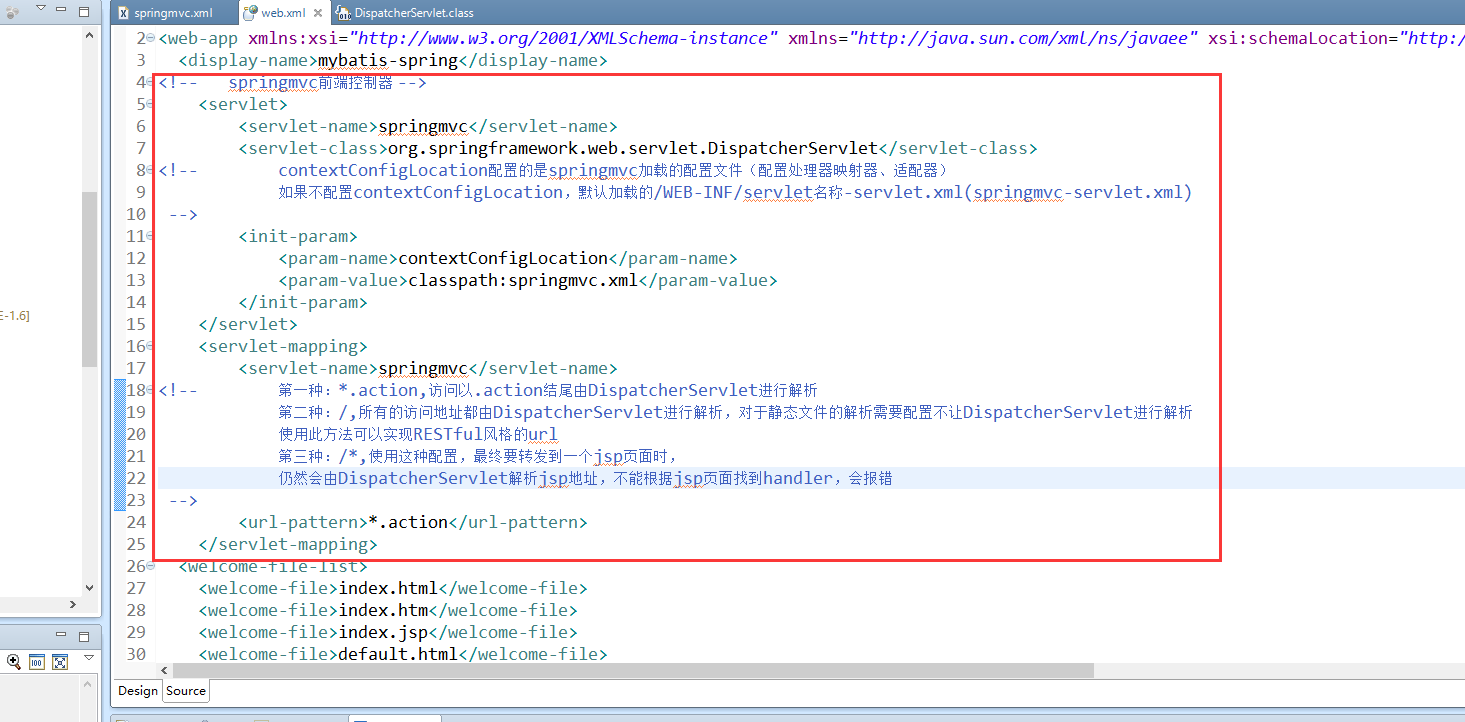
若是想要eclipse的程序能导在myeclipse中



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



1. Web.xml的配置（前端控制器的配置）



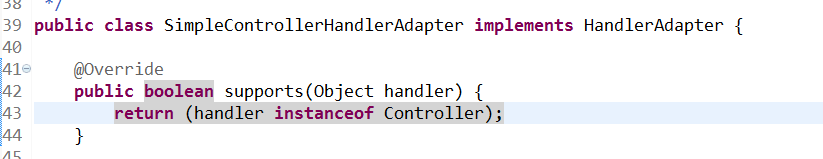
2、配置处理器映射器

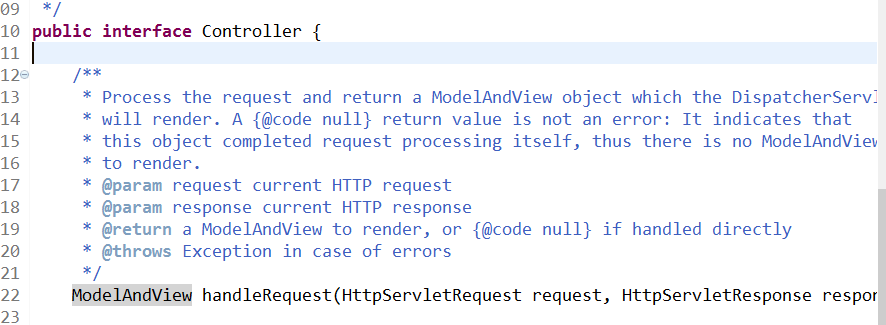
在springmvc.xml

3、配置处理器适配器

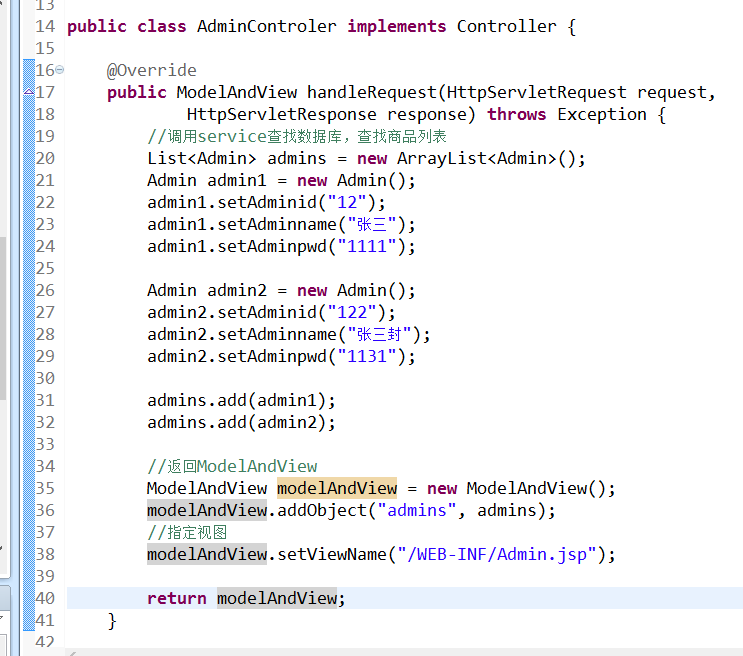
4、配置视图解析器

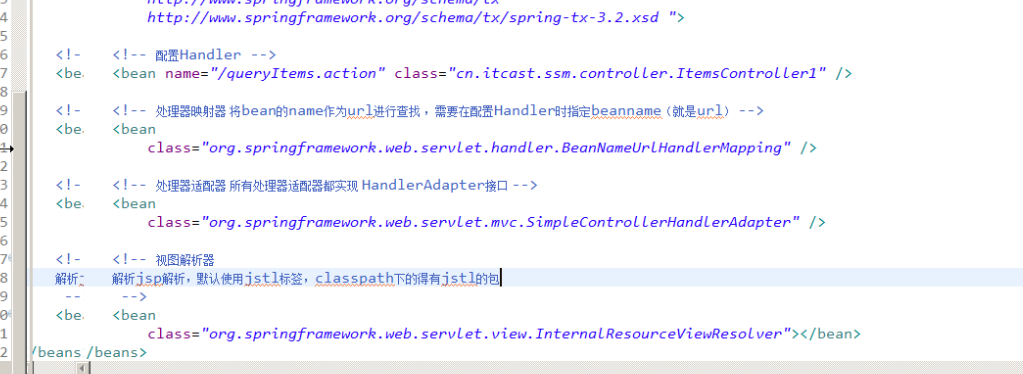
在springmvc.xml中进行配置查看源码

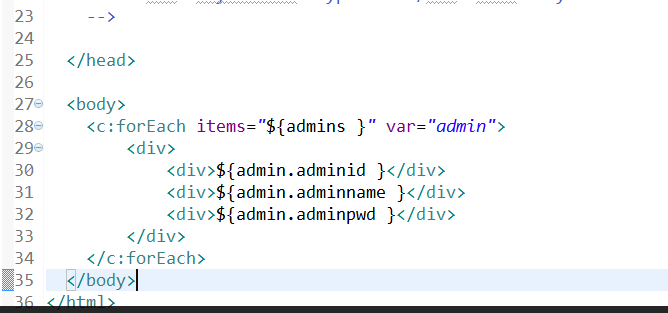




Controller



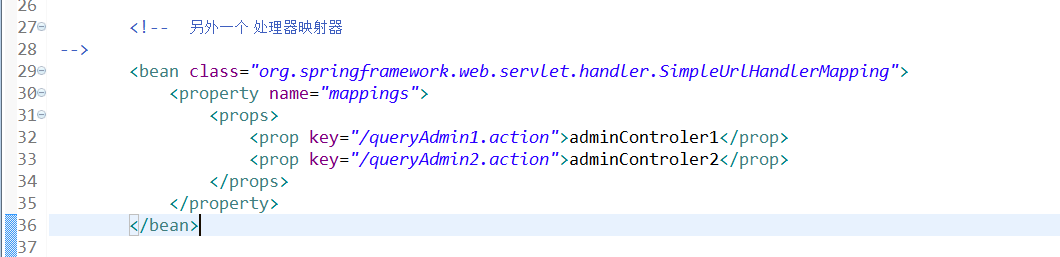




注意要是出现：说明视图指定的位置找不到



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



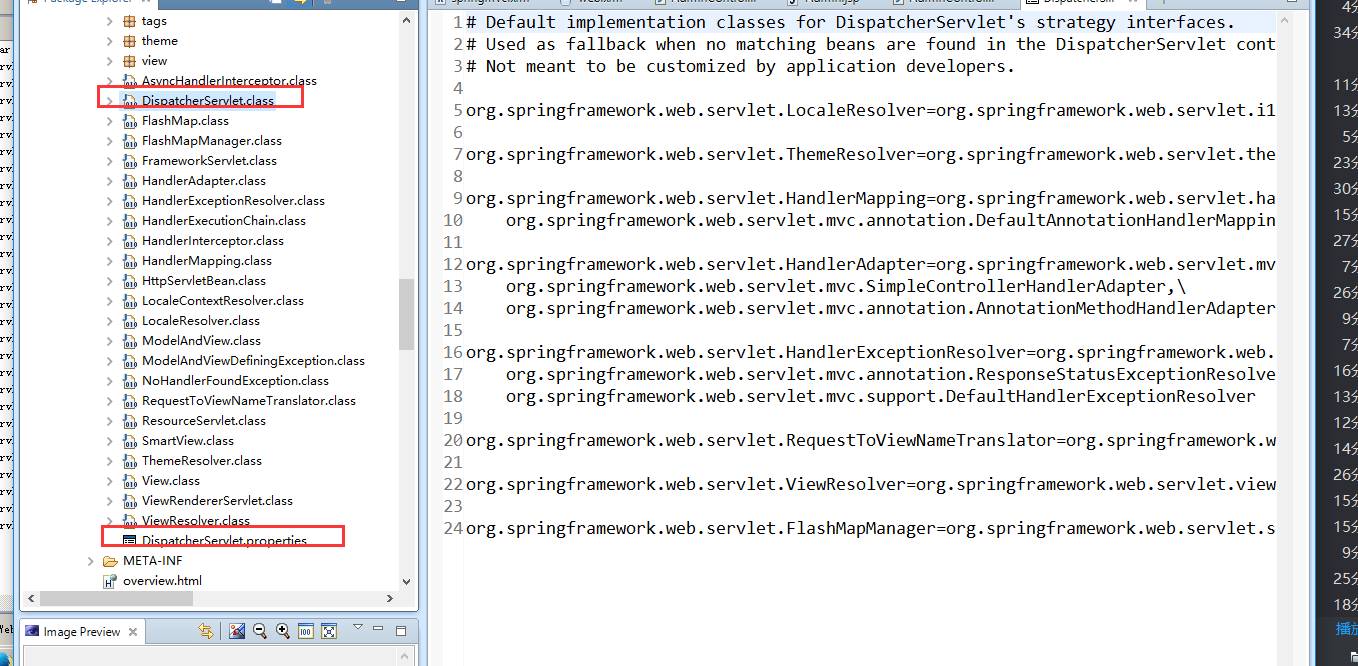




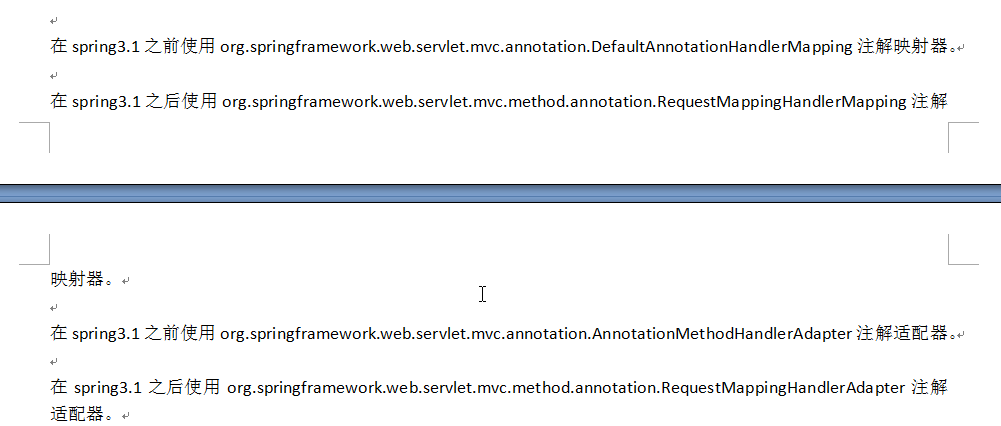
这个处理器适配器的好处是可以通过response来设置响应的格式如：json等

# 重点

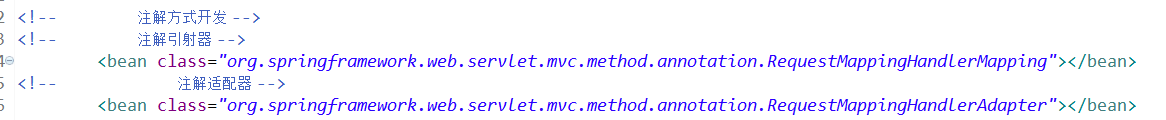
注解方式的映射器和适配器



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）

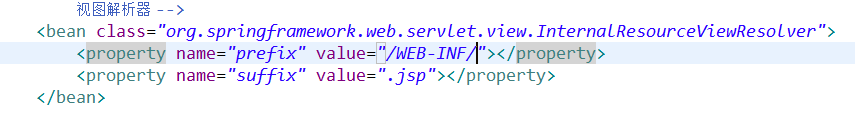


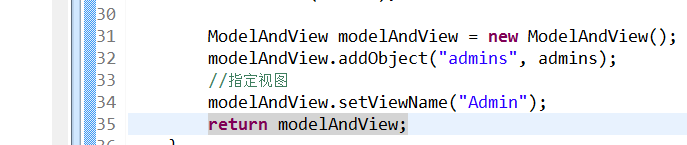
注解驱动 <mvc:annotation-driven></mvc:annotation-driven>



注解映射器的扫描器







不用指定前缀和后缀