# 目录

目录

[目录 1](#_Toc22626)

[1.Mybatis应用场景 3](#_Toc32469)

[Mybatis 和hibernate 的区别和应用场景 3](#_Toc1845)

[课程 3](#_Toc29335)

[原生态的jdbc遇到的问题 4](#_Toc31709)

[2.Mybatis是什么 5](#_Toc13049)

[Mybatis结构搭建： 6](#_Toc13645)

[Mybatis全局配置SqlMapConfig.xml 7](#_Toc5642)

[sql语句配置文件 7](#_Toc24730)

[Mysql自增主键返回 8](#_Toc3780)

[Mysql非自增主键值返回。 9](#_Toc7720)

[通过oracle序列生成主键 9](#_Toc6150)

[原始Dao层书写 10](#_Toc4410)

[测试类 10](#_Toc5218)

[Mapper代理接口开发 11](#_Toc5394)

[Mybatis内部实现原理 11](#_Toc12603)

[Mybatis代理接口存在的问题 12](#_Toc3076)

[SqlMapConfig.xml 13](#_Toc4112)

[properties属性 13](#_Toc8230)

[Setting 全局配置参数 14](#_Toc20199)

[typeAliases 定义别名 14](#_Toc19079)

[Mabits默认支持的别名有 14](#_Toc4398)

[自定义别名 15](#_Toc23826)

[TypeHandlers类型处理器 15](#_Toc29841)

[Mapper加载映射文件 16](#_Toc27777)

[单个xml文件的加载 16](#_Toc31125)

[多个xml文件批量加载 16](#_Toc8986)

[输出映射 17](#_Toc26106)

[1.resultType：输出结果表中的列名和类属性名相同，则可以映射成功 17](#_Toc2560)

[2.resultMap：输出结果表中的列名和类属性名不相同，则可以通过resultMap进行高级映射 17](#_Toc5932)

[动态sql操作 17](#_Toc1632)

[1.If判断 不需要占位符 17](#_Toc11601)

[3. <where></where> 针对语句中的第一个and 多则删除，少则添加。 18](#_Toc28871)

[3.sql片段 18](#_Toc31670)

[定义sql片段。 18](#_Toc8290)

[引用sql片段 18](#_Toc20902)

[如果引用其他配置文件的sql片段，则需要加上namespace 18](#_Toc20723)

[4. foreach标签 18](#_Toc1774)

[关系映射 19](#_Toc16836)

[数据库分析 19](#_Toc20897)

[一对一查询，关联查询两个表 19](#_Toc7391)

[一对多 20](#_Toc22230)

[ResultMap的继承 21](#_Toc29008)

[多对多 21](#_Toc13940)

[resultMap编写 21](#_Toc8260)

[对应pojo类 22](#_Toc4684)

[结果 22](#_Toc23611)

[总结 22](#_Toc21262)

[延迟加载 23](#_Toc4773)

[1. 什么是延迟加载 23](#_Toc16862)

[2. 例子 23](#_Toc7081)

[两条语句： 24](#_Toc913)

[resultMap 24](#_Toc10474)

[全局配置 25](#_Toc10975)

[测试 25](#_Toc29556)

[3. 总结 25](#_Toc263)

[查询缓存 26](#_Toc27211)

[一级缓存 27](#_Toc24762)

[缓存测试 27](#_Toc28201)

[应用场景 28](#_Toc7505)

[二级缓存 28](#_Toc19567)

[开启二级缓存 29](#_Toc135)

[Pojo类进行序列化 29](#_Toc31316)

[测试 30](#_Toc31583)

[针对某条sql语句不使用二级缓存 30](#_Toc23253)

[清除缓存 30](#_Toc19164)

[其他设置 31](#_Toc27902)

[（二级缓存）Spring整合ehcache（分布式缓存） 31](#_Toc6055)

[应用场景 33](#_Toc28020)

[局限性 34](#_Toc4070)

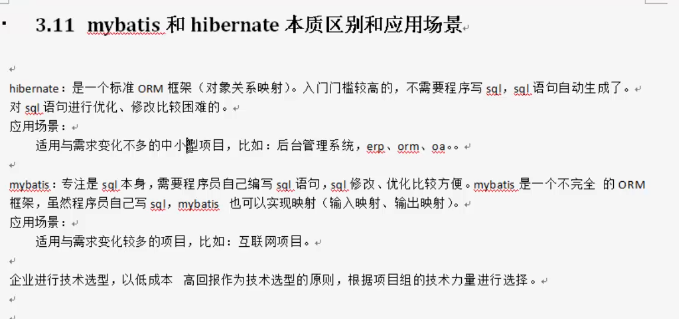
[Mybatis和spring整合 34](#_Toc3443)

[整合思想 34](#_Toc13519)

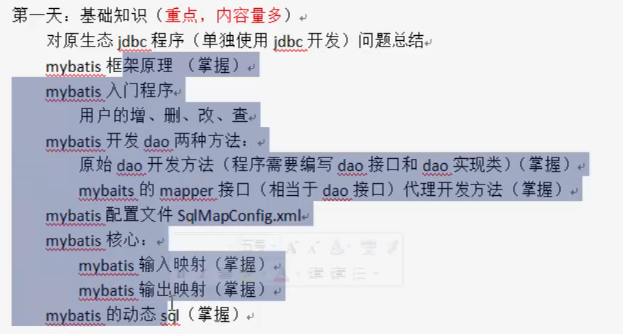
[步骤 34](#_Toc12958)

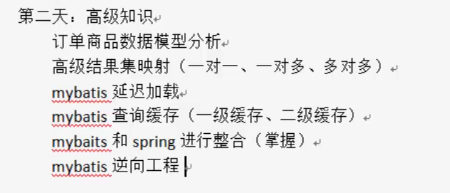
# 1.Mybatis应用场景

## Mybatis 和hibernate 的区别和应用场景



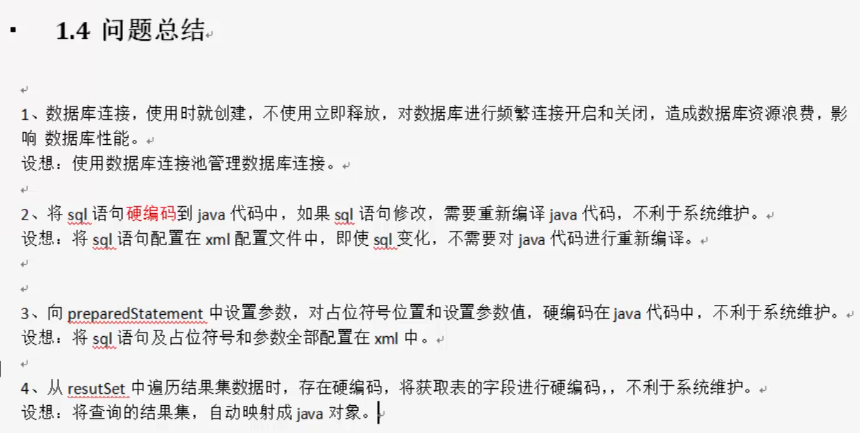
## 课程



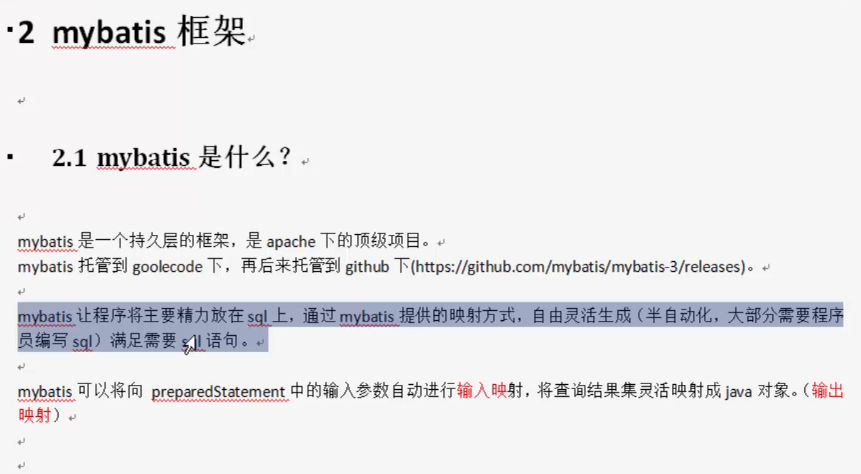


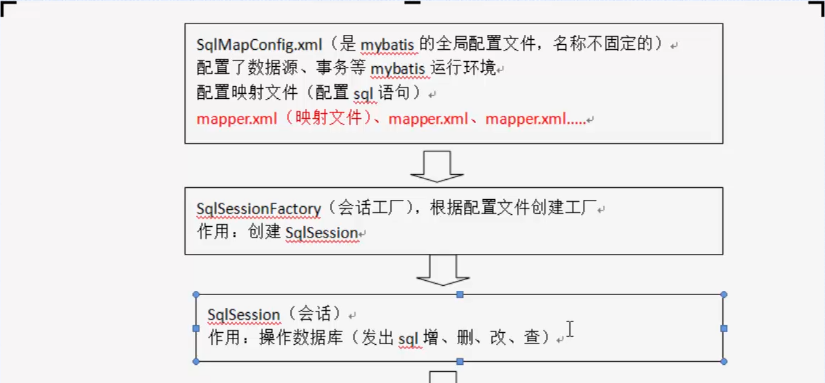
prepareStatment：预编译：statment发送一条sql语句，在数据库端进行编译，然后将编译的结果放在缓存中，如果再次请求的是同一条sql语句，则不会再次编译。提高了程序的性能

## 原生态的jdbc遇到的问题



# 2.Mybatis是什么







## Mybatis结构搭建：



### Mybatis全局配置SqlMapConfig.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <environments default="development">  <environment id="development">  <transactionManager type="JDBC"/>  <dataSource type="POOLED">  <property name="driver" value="com.mysql.jdbc.Driver"/>  <property name="url" value="jdbc:mysql://localhost:3306/springboottest"/>  <property name="username" value="root"/>  <property name="password" value="woshige2b"/>  </dataSource>  </environment>  </environments>  <mappers>  <mapper resource="org/mybatis/example/BlogMapper.xml"/>  </mappers>  </configuration> |

### sql语句配置文件

|  |
| --- |
| ParameterType：如果输入类型是简单类型且只有一个，则占位符内的标识符可以自拟。  resultType：只允许为po对象或者是简单类型。不存在list等  #{} 和${} 的区别：1.是否防sql注入 2.是否存在单引号  视频讲的：在只有一个简单类型参数的前提下， ${}内的标识符只能用value #{}内的标识符可以任意  传入为单个简单类型参数，${}内的标识符只能用value #{}内的标识符可以任意 。  传入多个简单类型参数，需要用@param定义接口上方法的参数，指定属性名  传入复杂类型，  如pojo类，则用属性名称表示 如User类中有属性那么，用#{name}  如list类，collection=”list” 或者用@param可以自定义collection的值。  如map类，使用key ，如map.put(“key”,”value”); 则用#{key} |

最后，解释下，什么叫“在build path中”？你可以这么理解，eclipse中，在build path下的类可以被编译运行，在build path下的配置文件可以被类以相对路径直接读写。

### Mysql自增主键返回

原理：Insert语句之后，返回自增主键的值。

需求：存在外键联系的表，在插入主表数据后，需要返回插入数据的主键的值（数据库通过自增的形式），然后将主键充当外键和其他信息存入到其他表中。

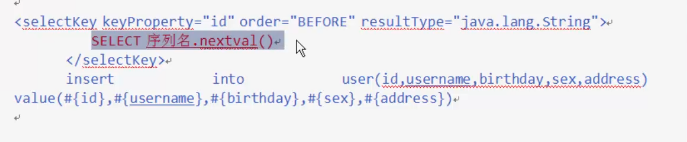
|  |
| --- |
| Sql语句 |
| 得到的主键值保存在传入的po对象内， 在此例子中，主键值保存在传入参数为User类型的对象中。 |

### Mysql非自增主键值返回。

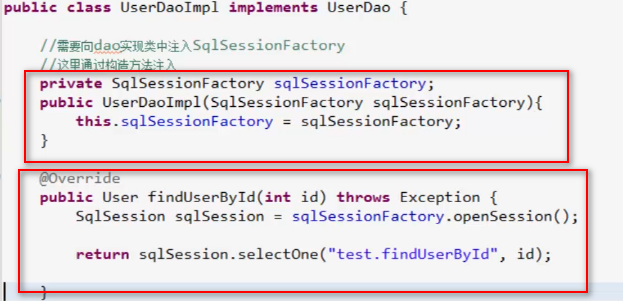
前提：使用的是mysql的uuid，且主键类型为 String 长度为35。

|  |
| --- |
|  |

### 通过oracle序列生成主键



## 原始Dao层书写



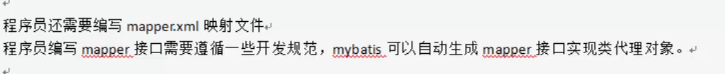
### 测试类

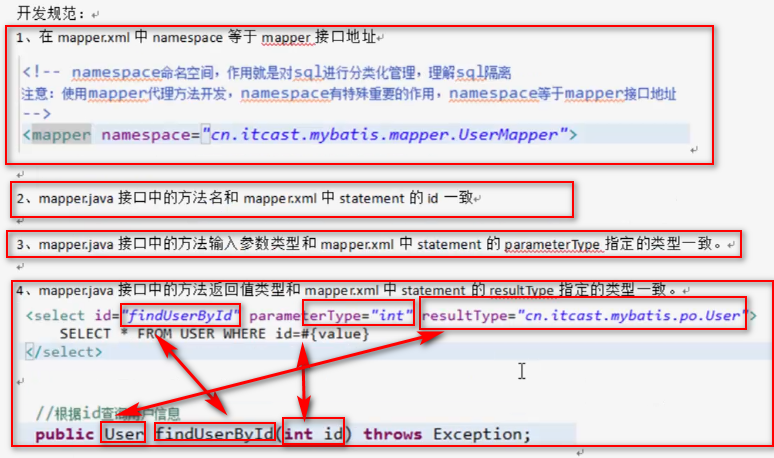


原始Dao开发的总结

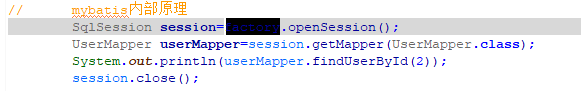


## Mapper代理接口开发





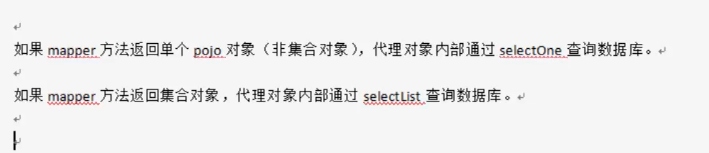
### Mybatis内部实现原理



通过session实现mapper接口，再进行操作。

### Mybatis代理接口存在的问题

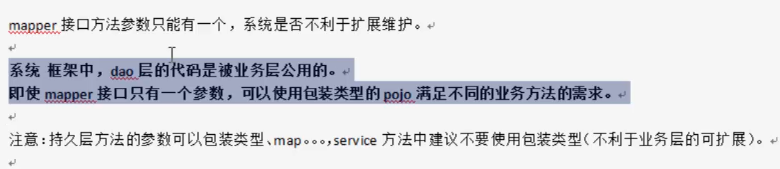
#### 1.select问题，怎么知道mybatis内部调用的是selectOne还是selectList。



#### parameterType 只有一个，传入参数只能为一个？

##### 解决方法：

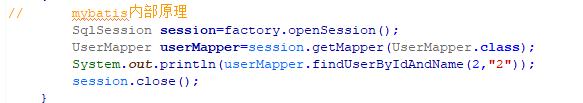
##### 1.定义包装类型，缺点：需要定义个类或者map对象



##### 2.采用mapper接口中的方法参数采用@param的注解，同时不写parameterType属性



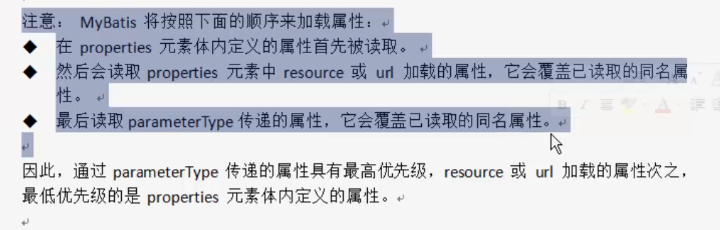


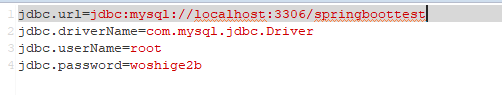


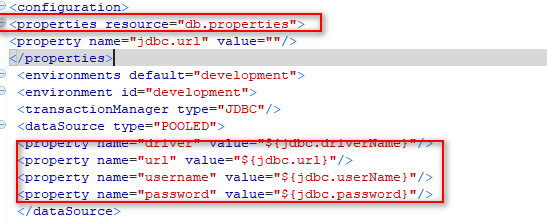
# SqlMapConfig.xml

## properties属性

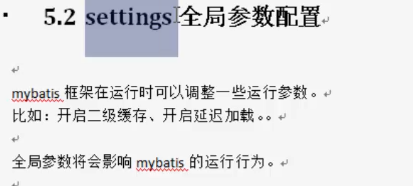
将连接数据库信息放到properties文件中







## Setting 全局配置参数



## typeAliases 定义别名

### Mabits默认支持的别名有

|  |  |
| --- | --- |
| 别名 | 映射的类型 |
| \_byte | byte |
| \_long | long |
| \_short | short |
| \_int | int |
| \_integer | int |
| \_double | double |
| \_float | float |
| \_boolean | boolean |
| string | String |
| byte | Byte |
| long | Long |
| short | Short |
| int | Integer |
| integer | Integer |
| double | Double |
| float | Float |
| boolean | Boolean |
| date | Date |
| decimal | BigDecimal |
| bigdecimal | BigDecimal |
| object | Object |
| map | Map |
| hashmap | HashMap |
| list | List |
| arraylist | ArrayList |
| collection | Collection |
| iterator | Iterator |

### 自定义别名

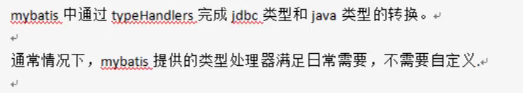
#### 单个类别名设置



#### **批量类别名设置**



## TypeHandlers类型处理器

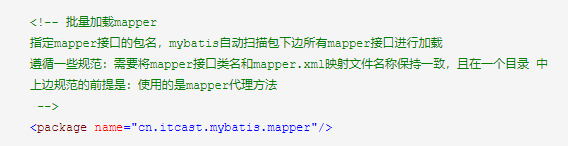


## Mapper加载映射文件

### 单个xml文件的加载



### 多个xml文件批量加载



要求和<mapper class=”” /> 是一样的,同样是需要mapper代理方法

# 输出映射

## 1.resultType：输出结果表中的列名和类属性名相同，则可以映射成功

## 2.resultMap：输出结果表中的列名和类属性名不相同，则可以通过resultMap进行高级映射

<resultMap type=”a类” id=””>

<id property=”” column=”” /> 如果多个列组成唯一标识id，那么配多个<id />

<result property=”” column=””/>

<associate property=”” javaType=””>

<id property=”” column=”” />

<result property=”” column=””>

</result></associate> //a类中的其他对象

</resultMap>

Associate标签，用于一对一查询，即一个对象中的某个成员变量为一个java对象



# 动态sql操作

## 1.If判断 不需要占位符

<if test=”user!=null”> 连接用and

User=#{user}

</if>

## <where></where> 针对语句中的第一个and 多则删除，少则添加。

## 3.sql片段

### 定义sql片段。

<sql id=””>

</sql>

### 引用sql片段

<include refid=” ” />

### 如果引用其他配置文件的sql片段，则需要加上namespace

例子：

<include refid=”com.hjy.mapper.UserMappser.conditionInTbuser” />

com.hjy.mapper.UserMappser为sql片段所在的mapper配置文件的namespace

conditionInTbuser为sql片段的id

同时要注意配置文件已经注册

## foreach标签

<foreach collection=””>

</foreach>

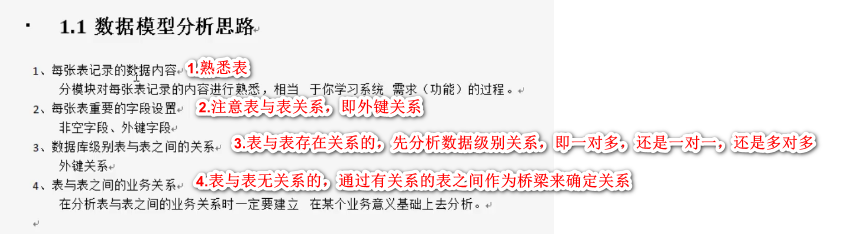
1.假如传入参数一个pojo或者map 则值为pojo类中的list类型的属性名，或者map的value为list类型的key值。

2.传入参数为list类型，@param可以自定义collection的值。或者用Collection=”list”

注意@param和使用Collection=”list”不可以兼容

# 关系映射

## 数据库分析



user：用户信息表

Order：订单表

Orerdetail 订单明细，一个订单表有多个订单明细，一个明细行数据只存放一个商品，所以，一个订单表中一行数据，订单明细表则可能会有多行数据。

Item ：商品表

有关联的表

User--order

Order--orderdetail

Orderdetail--item

其余user--orderdetail和item等无直接联系

User--》order 一对多

Order--》orerdetail 一对多

Orderdetail--》item 一对一 （订单明细中的商品对应唯一一件商品）

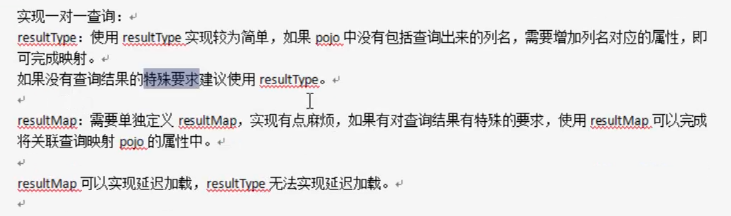
Item--》orderdetail 一对多

Orderdetail--》order 一对一 （订单明细表中有订单号，唯一对应一个订单）

Order--》user 一对一 （订单表有用户id所有唯一对应一个user）

## 一对一查询，关联查询两个表

两个表表示有两个pojo类。



用resultType，需要写一个扩展类，再扩展类继承某个pojo类（查询结果包含字段较多的），并再其中中添加查询结果的其他字段。

用resultMap，在一个pojo类中添加另一个pojo类为成员对象，用resultmap映射。

## 一对多



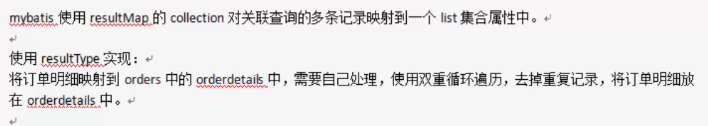
查询 用户信息（user）和订单信息（order），关联订单详细（orerdetail）。可知有四条记录，因为用户信息和订单信息查询结果为两条，但一个订单信息对应订单详细有多条。

即存在一对多的关系。

如果用resulttype 则结果也为四条。

如果用resultmap 接收pojo类为 class order 中包含成员变量 user和list<orderditail> 类型，则结果只有两条。





## ResultMap的继承

<resultMap extends=”其他resultmap的id”>

## 多对多

User --》item 一对多

Item--》user 一对多

所以 user和item 的关系是多对多

查询用户购买的商品的信息

User--》order--》orderdetail--》item

### resultMap编写

<resultMap id=”” type=””>

<id column=”” propety=”” />

<result column=”” property=””/>

<collection property=”order” ofType=”order表对应的pojo类”>

<id column=”” propety=”” />

<result column=”” property=””/>

<collection property=”orderdetail” ofType=”orderdetail表对应的pojo类”>

<id column=”” propety=”” />

<result column=”” property=””/>

<association property=”” javaType=””>

<id column=”” propety=”” />

<result column=”” property=””/>

</association>

</collection>

</collection>

</resultMap>

### 对应pojo类

共有四个类user order orderdetail item。

User类含有成员 List<order>

order类中含有成员类型List<orderdetail>

Orderdetail类含有成员类型 item

### 结果

类似这个表



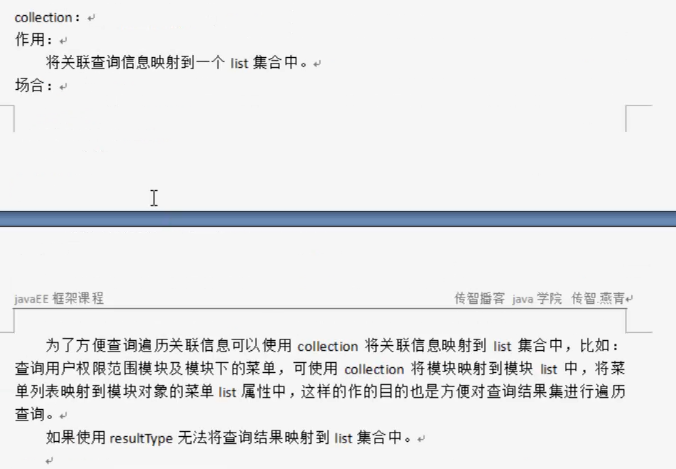
数据库中共有4条记录，使用resultmap查询后，只显示一条封装在user类中。

## 总结

使用resultType还是resultMap根据需求

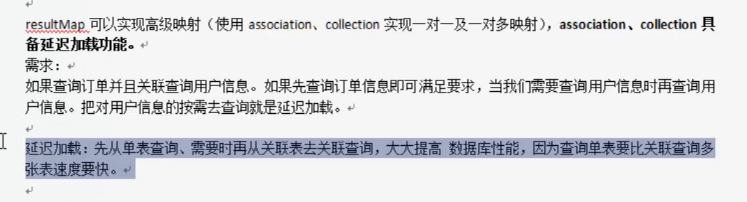
应用resultMap特殊场合





# 延迟加载

## 什么是延迟加载

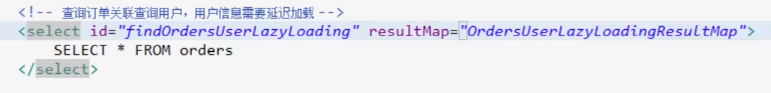


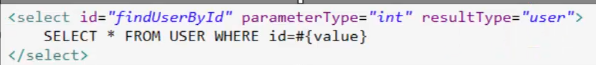
## 例子

通过订单查询关联用户信息

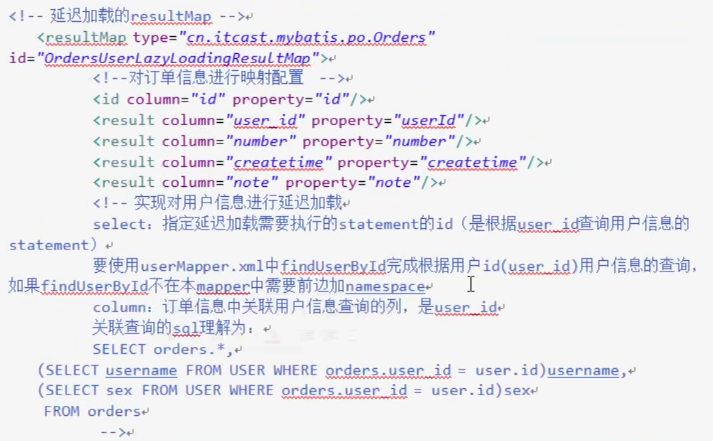
先查询订单信息，再查询用户信息。

### 两条语句：

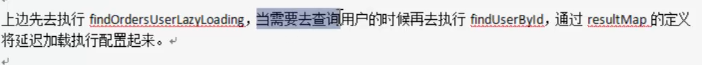




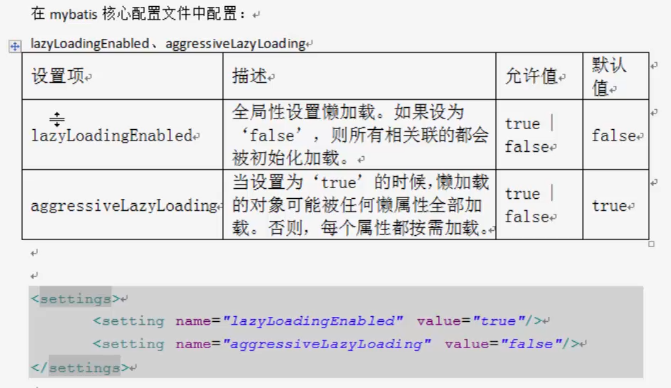
### resultMap







### 全局配置

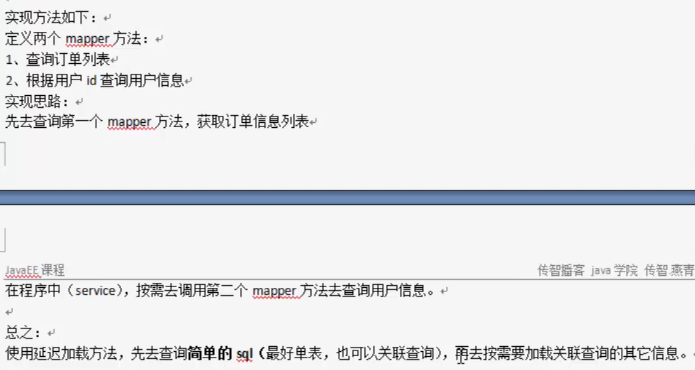


### 测试



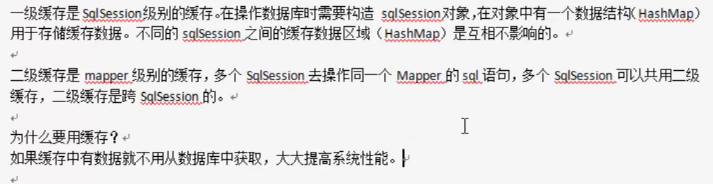
## 总结

不适用mybatis的collection和associaction用延迟加载

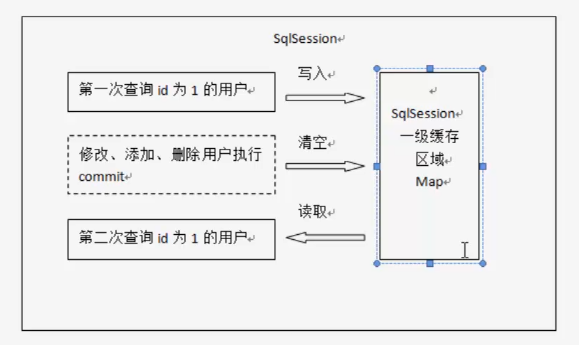


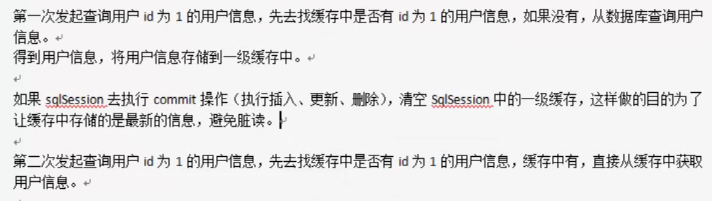
# 查询缓存





## 一级缓存

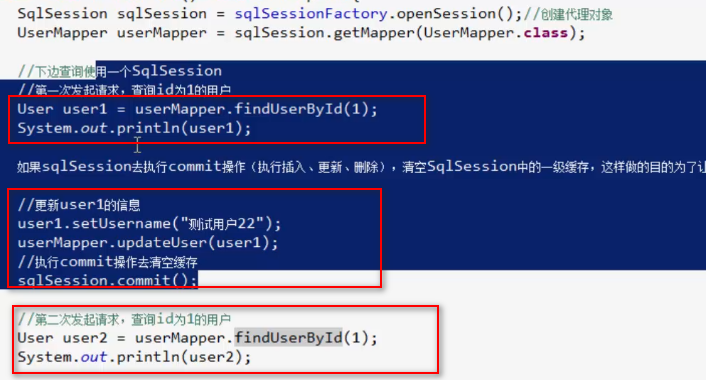




### 缓存测试

Mybatis默认支持一级缓存

前提，同一个sqlsession下。

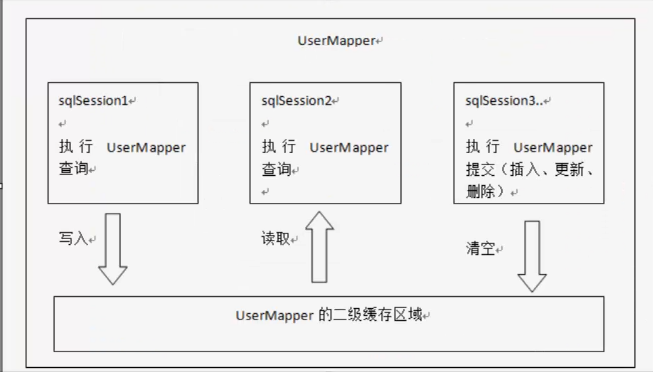


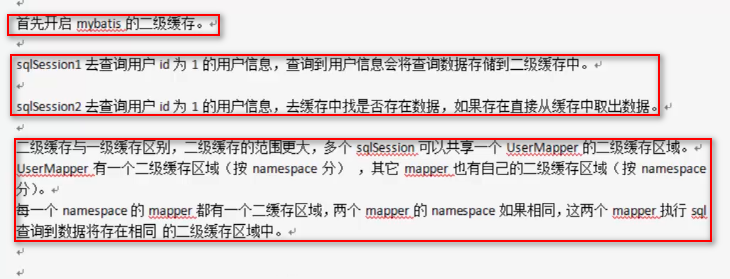
### 应用场景

Sevice中开始事务，就会创建sqlsession，调用service方法结束，sqlsession会关闭，则缓存消失。即controller两次调用sevice的查询方法时，拿不到缓存。

只有在同一个方法中调用两次查询方法，才能取得缓存。

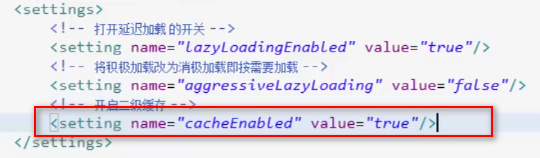
## 二级缓存





### 开启二级缓存

1. 在全局配置文件中开启



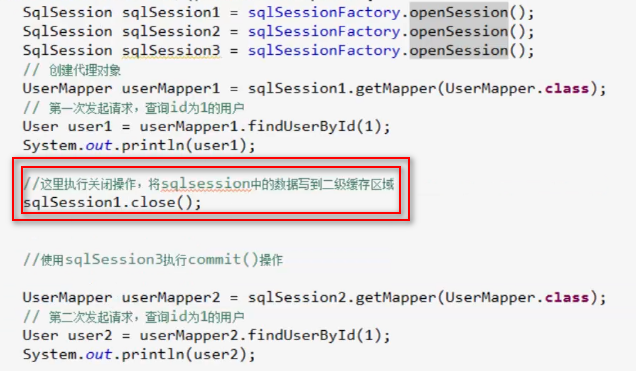
1. 在要开启二级缓存的mapper中开启



### Pojo类进行序列化

封装的类实现serializable

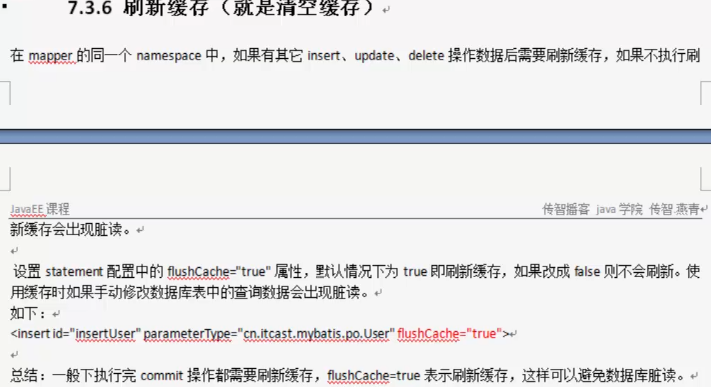
### 测试



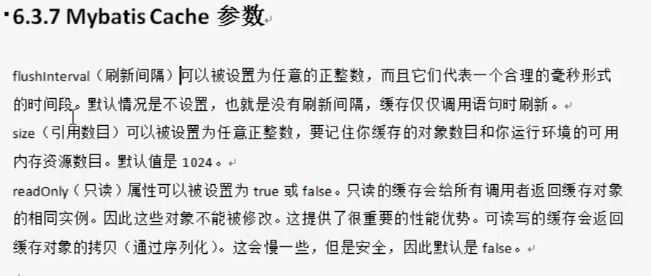
### 针对某条sql语句不使用二级缓存



### 清除缓存



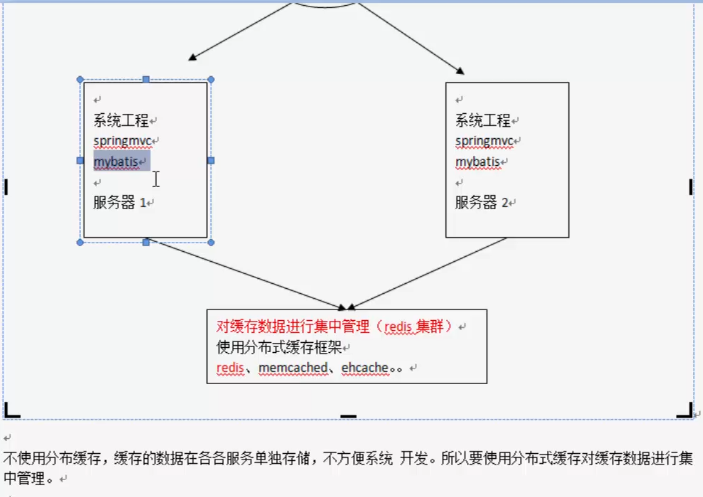
### 其他设置



### （二级缓存）Spring整合ehcache（分布式缓存）

Mybatis无法实现分布式缓存，需要和其他分布式缓存框架整合。



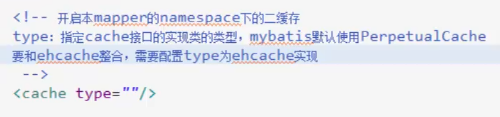


#### 整合原理：

Mybatis提供一个Cache接口，加入整合包后，存在一个cache的实现类。

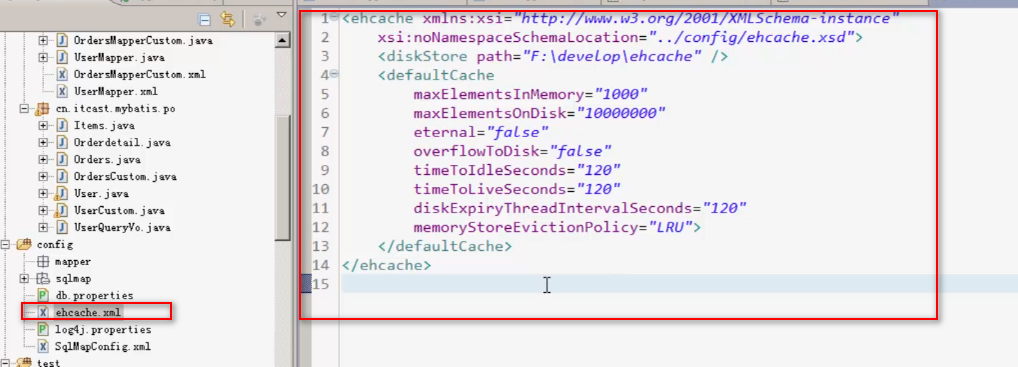
可以整合redis或者ehcach等分布式缓存框架。

#### 步骤1.



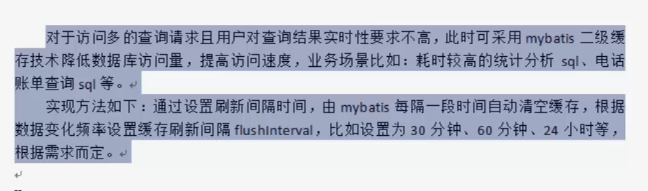


#### 步骤2.加入ehcache的配置文件

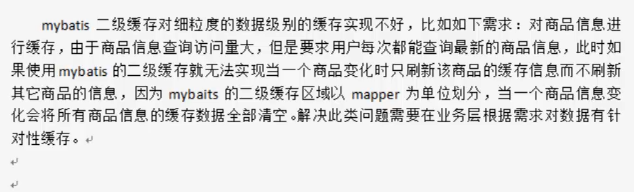


结束

### 应用场景

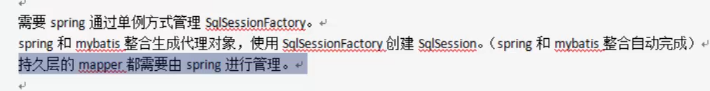


### 局限性



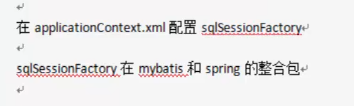
# Mybatis和spring整合

## 整合思想



## 步骤

1. 
2. 配置sqlsessionfactory



1. Mapper代理





