知识回顾：

1）动态SQL，mybatis中非常灵活的内容，根据业务需求，SQL语句动态变化。由一系列标签组成，常用<if test=”id!=null”>，动态去掉或者增加一部分SQL。

<where>去掉多余and或者or

<set>只能用在update标签中，它作用去掉最后一个逗号

<foreach>主要用于in子查询串open(，colse)，separator,，collection=array/list/ids，item=”id” #{id}

2）接口访问方式，特点：面向对象思考方式

UserMapper mapper = getMapper(UserMapper.class);

实现代理对象，无需我们实现这个UserMapper接口。动态代理技术，两种：$ProxyN（jdk动态代理）（只支持接口方式）、cglib动态代理（支持接口和实现类方式）（了解即可，广泛应用框架底层）

3）实现接口方式

a.写接口文件，路径：必须放在和映射文件同样路径中。

b.方法都怎么产生？安装UserMapper.xml映射文件中每个方法标签，按id方法名称，parameterType参数变成方法的参数，resultType返回值变成方法的返回值，List<User>

c. UserMapper mapper = getMapper(UserMapper.class);

d．List<User> userList = mapper.find(5);

本质，其实没有另外做一套访问数据库代码，而是利用原有的实现，只不过利用接口再去调用这个方法而已，最终selectList()。（了解源码）

看源码？

推荐：springmvc源码、mybatis源码、spring源码、Jdk HashMap源码，LinkList源码

User user=new User();

Spring geBean(\*);

3）resultType，resultMap

resultMap Mybatis最强大映射工具，字段映射 user\_name映射userName

对象关联，用到

今天任务：

1. 对象关联

一对一：qq登录（登录表+扩展信息），

一对多：组织机构管理（一个部门下有多个员工）

多对一：多个员工从属一个部门

多对多：讲师和学员；一个讲师可以有多个学员，一个学员有多个讲师

在mybatis简化这个关系：只有2种关系，

对一（association）和对多（collection）和ResultMap

1. SQL关联查询，一个查询获取多个表的数据 左关联left join万能
2. 小点（难度）懒加载、缓存（一级缓存、二级缓存-默认不开启）分布式内存数据库redis
3. 自动生成代码工具，自动生成SQL语句
4. 对象关联：对一

tb\_door门店表

tb\_order订单表

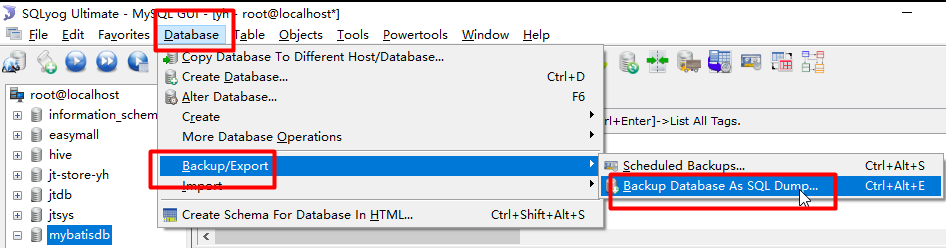
tb\_order\_detail详情表

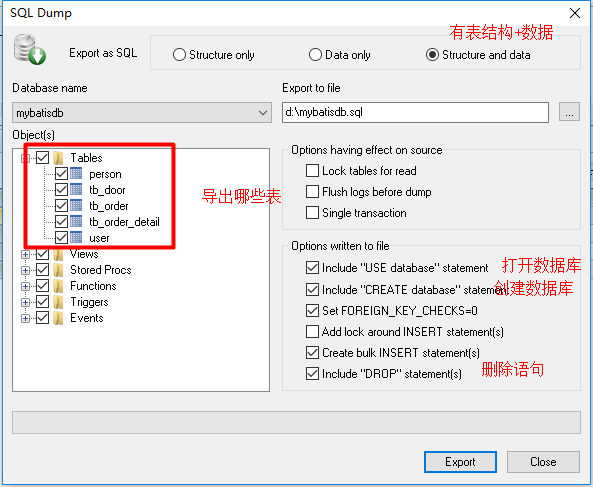
关系：

订单和门店什么关系？一个订单对应一个门店（对一）

订单和详情什么关系？一个订单下可以有多个商品（详情）（对多）

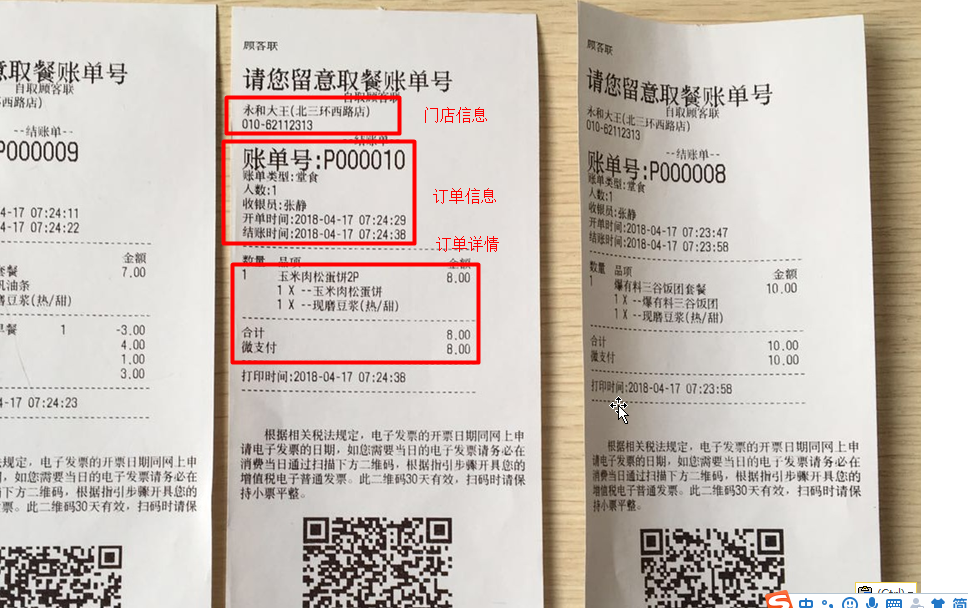
导出数据库





开发步骤：

1. 把数据库表导出，形成sql文件（创建表，数据）
2. 导入数据
3. 创建pojo对象，涉及对象关联 order直接获取door
4. 创建映射文件ResultMap结构
5. 测试类



1. 查询数据，形成一个ResultSet结果集
2. 利用这个结果集每个字段，order\_no，获取它去找pojo的setOrderNo()
3. 把数据库字段对应值通过set方法就放入pojo的属性中

对象怎么放？

order.door 先把对应door数据查询出来

SELECT \* FROM tb\_door

WHERE id = (SELECT door\_id FROM tb\_order WHERE id=1 )

子查询，查询中嵌套一个查询，称为：子查询

返回值，不能是多个，也不能是多条

ResultMap加上door字段信息，关联：对一

SELECT \* FROM tb\_door

WHERE id =

(

SELECT door\_id FROM tb\_order

WHERE id=1

)

SELECT \* FROM tb\_order WHERE id=1

Order

[

door=Door

[id=1, name=永和大王（北三环西路店）, tel=110-62112313],

id=1,

orderNo=P000011,

orderType=堂食,

personNum=1,

cashier=张静,

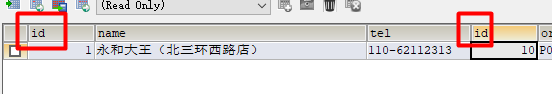
createTime=Thu Apr 26 14:49:07 CST 2018,

endTime=Tue Apr 17 07:24:38 CST 2018,

paymentType=微支付,

price=8.0

]



Mybatis中对表映射，还是对结果集？结果集

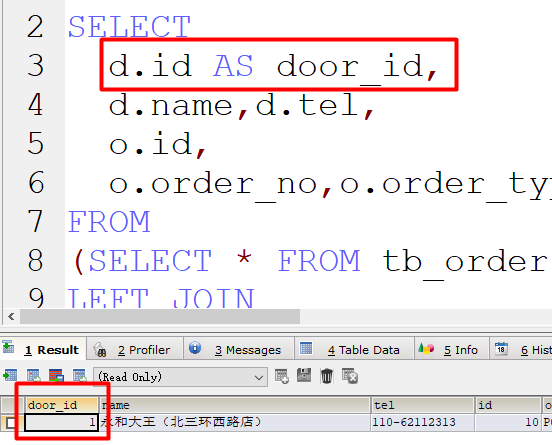
上面的问题就是字段名称重复了，构建大的结果集中字段重名。

重名对SQL结果是不影响，它是影响，但是mybatis在映射时，找不到对应关系了？

Mybatis。获取到字段id，setId(id)。有两个setId()，它获取了第一个，第二个根本没有。

解决方案：

结果集字段同名，改名！



Order [door=Door [id=1, name=永和大王（北三环西路店）, tel=110-62112313], id=10

注意：在mybatis中有“bug”，不允许结果集中字段同名。如果同名。Mybatis不出错。

结果逻辑数据正确吗？不正确。

对一：

<association property=*"door"*

javaType=*"Door"* column=*"door\_id"*>

1. 对多
2. 构建一个大大结果集，结果集中要出现所有字段（订单、门店、详情）

SELECT

od.id AS order\_detail\_id,

od.order\_id,

od.num,od.item,od.price as order\_detail\_price,

o.door\_id,

o.name,o.tel,

o.id,

o.order\_no,o.order\_type,o.person\_num,o.cashier,o.create\_time,o.end\_time,o.payment\_type,o.price

FROM

(

SELECT

d.id AS door\_id,

d.name,d.tel,

o.id,

o.order\_no,o.order\_type,o.person\_num,o.cashier,o.create\_time,o.end\_time,o.payment\_type,o.price

FROM

(SELECT \* FROM tb\_order WHERE id=#{id}) o

LEFT JOIN

(

SELECT \* FROM tb\_door

WHERE id = (SELECT door\_id FROM tb\_order WHERE id=#{id})

) d

ON o.door\_id=d.id

) o

LEFT JOIN

(SELECT \* FROM tb\_order\_detail WHERE order\_id =#{id}) od

ON o.id=od.order\_id

1. 订单和详情，对象关系：一对多（pojo）；

public class Order {

//关联对象

//一个订单从属一个门店，一对一（对一）

private Door door; //外键形式改成对象关联形式

//一个订单有多个商品，一对多（对多）

private List<OrderDetail> orderDetails;

1. 映射文件（对多）ResultMap

<!-- ResultMap映射，字段和属性不匹配 -->

<resultMap type="Order" id="orderRM">

<!-- 主键 -->

<id property="id" column="id"/>

<!-- 普通属性 -->

<result property="orderNo" column="order\_no"/>

<result property="orderType" column="order\_type"/>

<result property="personNum" column="person\_num"/>

<result property="cashier" column="cashier"/>

<result property="createTime" column="create\_time"/>

<result property="endTime" column="end\_time"/>

<result property="paymentType" column="payment\_type"/>

<result property="price" column="price"/>

<!-- 关联对象：对一

property就是对象中关联实例名称

javaType代表存放数据的对象类型

column代表外键值

-->

<association property="door"

javaType="Door" column="door\_id">

<id property="id" column="door\_id"/>

<result property="name" column="name"/>

<result property="tel" column="tel"/>

</association>

<!-- 关联对象：对多，固定搭配 collection对应ofType -->

<collection property="orderDetails"

ofType="OrderDetail" column="order\_id">

<id property="id" column="order\_detail\_id"/>

<result property="num" column="num"/>

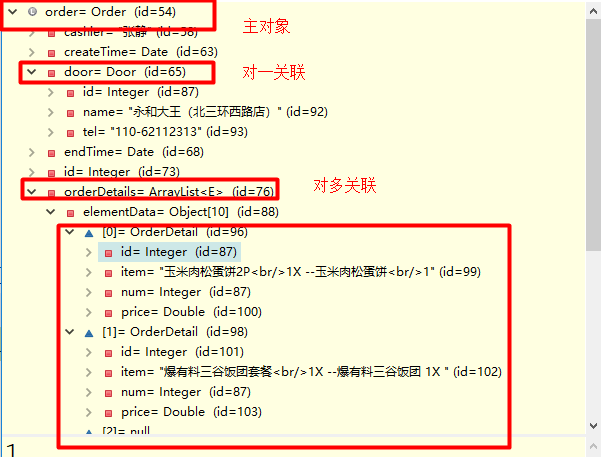
<result property="item" column="item"/>

<result property="price" column="order\_detail\_price"/>

</collection>

</resultMap>

1. 测试



懒加载：

Hibernate 把数据库数据映射pojo Java对象中。

Order

Door

List<OrderDetail>

如果假设我们的业务只需要orderNo。这时立即加载（传统方式）

会把所有的数据都加载。

如果是懒加载，它不会加载Door和List<OrderDetail>的信息

只到用户调用户door.getName()这时会自动触发新的查询，返回信息。

立即加载用户无用数据也会直接返回。会慢。

懒加载，性能高，缺点：多次查询

Mybatis懒加载机制，默认是立即加载

1. 在核心配置文件中配置setting，配置2个值

<!-- 全局配置 -->

<settings>

<!-- 默认未开启懒加载 -->

<setting name="lazyLoadingEnabled" value="true"/>

<!-- 设置为不积极的加载 -->

<setting name="aggressiveLazyLoading" value="false"/>

</settings>

1. 改变映射文件的形式，每个查询单独的构建

SELECT \* FROM tb\_order WHERE id=10

--子查询参数都可以由上面的结果集中获取

SELECT \* FROM tb\_door WHERE id=1

SELECT \* FROM tb\_order\_detail

WHERE order\_id=10

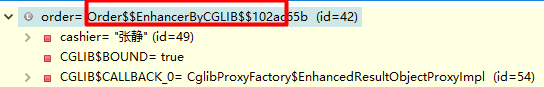
3）改变ResultMap

单独构建查询，关联中引入select属性，指向这个查询

上面的改造，实现懒加载机制

sqlMapConfig.xml中元素是有先后顺序的要求：

configuration" 的内容必须匹配 "(properties?,settings?,typeAliases?,typeHandlers?,objectFactory?,objectWrapperFactory?,plugins?,environments?,databaseIdProvider?,mappers?)



这个类使用CGLIB动态代理方式创建

实际当中，例如电商，都是单表操作！！！！

1. 关联查询，leftjoin低
2. 懒加载（了解）面试

数据库有缓存吗？

数据库有缓存，它把sql语句作为key，map(k,v)。如果k存在，就直接返回（缓存）；如果k不存在，查询数据库，组织数据库，然后把数据放到缓存中，最后在返回调用中。

Mybatis缓存

1. 一级缓存sqlSession级别缓存（默认开启，而且不能配置）

@Test //一级缓存存在，如果缓存中没有数据，发出SQL语句到数据库查询，如果缓存中有数据就不查询数据库，直接返回

public void cacheLevel11() throws IOException{

InputStream is = Resources.getResourceAsStream("sqlMapConfig.xml");

SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(is);

//一级缓存SqlSession

SqlSession session = factory.openSession();

Person p = session.selectOne("cn.tedu.mybatis.pojo.PersonMapper.get",1);

System.out.println(p);

Person p1 = session.selectOne("cn.tedu.mybatis.pojo.PersonMapper.get",1);

System.out.println(p1);

}

1. 二级缓存sqlSessionFactory级别缓存，范围大，可以在多个sqlSession共享。（默认不开启，可以配置开启缓存）

一级用户无法操作，mybatis自身管理的，二级可以配置带都被redis替代

<mapper namespace="cn.tedu.mybatis.pojo.PersonMapper">

<!-- 开启二级缓存 -->

<cache/>

@Test //二级缓存SqlSessionFactory

public void cacheLevel2() throws IOException{

InputStream is = Resources.getResourceAsStream("sqlMapConfig.xml");

SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(is);

SqlSession session = factory.openSession();

Person p = session.selectOne("cn.tedu.mybatis.pojo.PersonMapper.get", 1);

session.close();

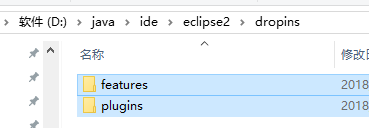
SqlSession session1 = factory.openSession();

Person p1 = session1.selectOne("cn.tedu.mybatis.pojo.PersonMapper.get", 1);

}

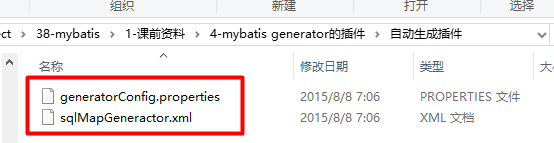
Eclispe自动生成mybatis代码的工具

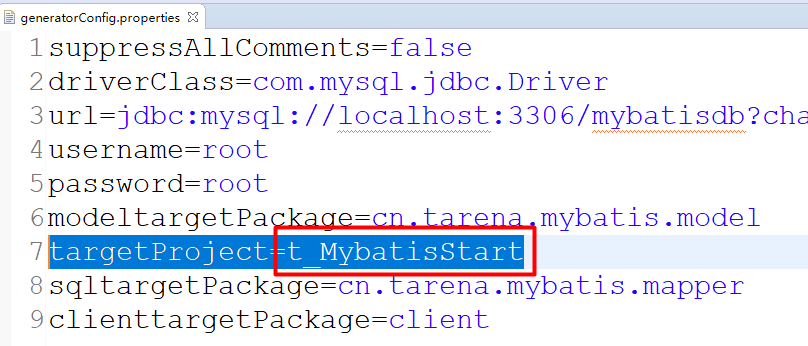
1. 把eclipse插件安装



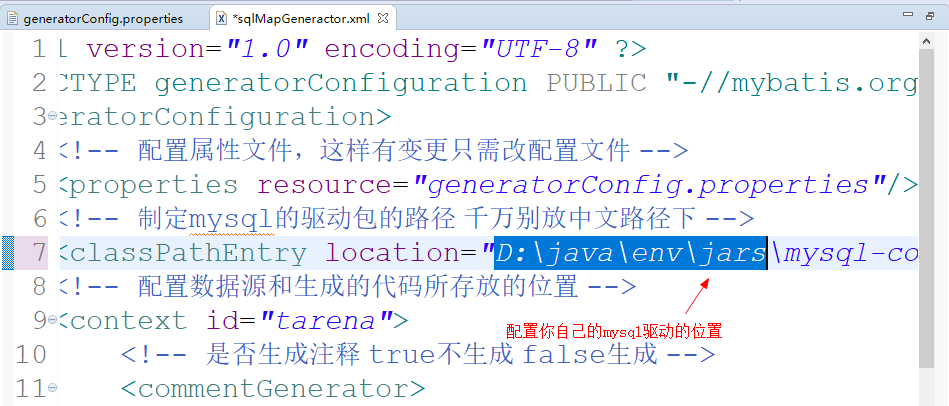
Eclipse重启时，自动扫描这个插件目录，如果有插件会自动添加

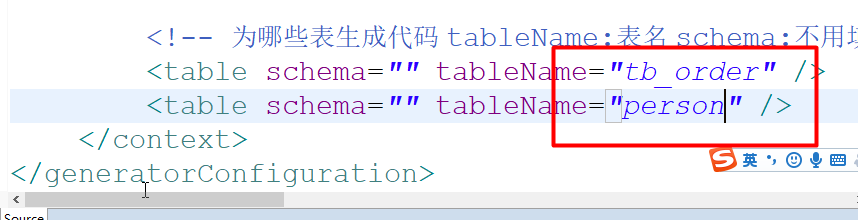
1. 修改配置文件，自动生成mybatis代码





修改指向自己的项目





小结：

1. 关联对象pojo

private Door door; 对一

private List<OrderDetail> orderDetails； 对多

1. 映射文件

resultType无能为力

必须使用resultMap

<resultMap type=”Order” id=”orderRM”>

<id property=”id” column=”id”/>

<result property=”orderNo” column=”order\_no”/>

。。。

对一

<association property=”door” javaType=”Door” column=”door\_id”>

<id>

<result>

。。

</association>

对多

<collection property=”orderDetails” ofType=”OrderDetail” column=”id”>

。。。

1. 懒加载

一般都是使用立即加载（多个关联对象）

懒加载，只查询主对象，关联的对象都不执行。

优点：每次加载数据量小，性能高。

如果访问子对象的数据，它会自动发起sql请求，查询子对象

缺点：多次访问数据库

懒加载mybatis写法简单，3条查询sql搞定；

立即加载，大大结果集，要包含所有的对象，left join实现。

1. 缓存，分两级

一级缓存，sqlSession级别，共享一个session，mybatis自己控制不需用户动

二级缓存，sqlSessionFactory级别，共享一个factory，默认不开启，在自己的映射文件中要开启二级缓存支持<cache/>，支持第三方ehcache被redis替代。

1. 插件

把插件复制到eclipse路径中

把两个文件复制到任一一个项目中，它可以生成代码放到另外一个项目

配置，生成到哪个项目中，配置jar数据库驱动，配置要生成数据库表

如果插件成功，xml配置文件右键多了一个菜单