使用的是面向对象的编程技术，而存放数据的是关系型数据库

编程使用的是对象模型 存放数据使用的是关系模型

对象模型和关系模型的阻抗不匹配性

对象关系映射Object Relation Mapping，简称ORM是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术，允许以操作对象的方式操作关系型数据库

JDBC

PreparedStatement ps=conn.prepareStatement(“insert into t\_users(username,password) values(?,?)”);

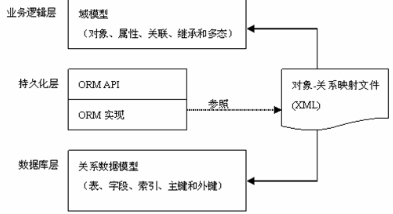
ps.setString(1,user.getUsername());

ps.setString(2,user.getPassword());

ps.executeUpdate();

将对对象的操作转换为sql语句【对关系的操作】

ORM原理



IUserServ public void save(UserBean user){….}

依赖于映射文件，将对user的操作转换为对关系的操作[sql]

ORM的优点

完全面向对象的编程思想，无SQL出现

减少代码编写，提高工作效率

提高访问数据库的性能，降低访问数据库的频率 [一般来说ORM技术的实现中一般都会采用缓存技术，从而实现减少数据库访问次数]

具有独立性，发生变化时，不会影响上层的实现

MyBatis是一种半自动化ORM工具，Hibernate是一种自动化ORM工具

**什么是Hibernate**

一种对象和关系之间映射ORM的轻量级框架【相对于EJB2而言】。它对JDBC进行了轻量级的对象封装，使得Java程序员可以使用对象编程思维来操作关系数据库

EJB：企业级JavaBean，是sun提出的一种企业级解决方案，目前最新是EJB3。 EJB可以分为3大类: SessionBean用于实现业务逻辑的封装，可以分为有状态和无状态两种.

EntityBean用于实现对持久化操作的封装,可以分为BMP和CMP容器管理的持久化

MessageDriven Bean消息驱动bean，用于实现异步数据访问

|  |  |
| --- | --- |
| OOP | DB |
| 类class | Table |
| 对象 | 一行数据 |
| 属性 | 列 |
| 标识属性 | 主键 |

Hibernate优点

开源并且免费，方便需要时研究，改写源码，进行功能的定制

简单，避免引用过多复杂问题，进行轻量级封装，容易调试

具有可扩展性，API开放，根据需要扩展

稳定的性能，开源社区，团体，公司和个人的广泛支持

文档丰富，齐全，目前依然不断更新，维护中

hibernate缺点

由于对持久层封装过于完整，导致开发人员无法对SQL进行优化，无法灵活使用JDBC的原生SQL，Hibernate封装了JDBC，所以没有JDBC直接访问数据库效率高。要使用数据库的特定优化机制的时候，不适合用Hibernate

框架中使用ORM原则，导致配置过于复杂，一旦遇到大型项目，比如300张表以上，配置文件和内容是非常庞大的，另外，DTO满天飞，性能和维护问题随之而来

如果项目中各个表中关系复杂，表之间的关系很多，在很多地方把lazy都设置false，会导致数据查询和加载很慢，尤其是级联查询的时候

Hibernate在批量数据处理时有弱势，对于批量的修改，删除，不适合用Hibernate,这也是ORM框架的弱点

为什么要用Hibernate

Hibernate对JDBC访问数据库的代码做了封装，动态产生SQL，大大简化了数据访问层繁琐的重复性代码

Hibernate是一个基于jdbc的主流持久化框架，是一个优秀的ORM实现，它很大程度的简化了dao层编码工作

Hibernate使用java的反射机制增强程序类实现透明性

Hibernate的性能非常好【缓存】，因为它是一个轻量级框架。映射的灵活性很出色。它支持很多关系型数据库，从一对一到多对多的各种复杂关系

持久化Java类必须遵循的原则

提供一个标识属性（identifier property）。如果没有该属性，一些功能不起作用，比如：级联更新（Cascaded updates）Session.saveOrUpdate()

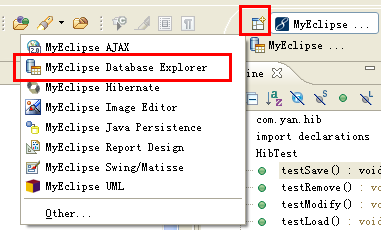
如果是集合类型的属性，它的类型必须定义为集合的接口。例如：List、Set

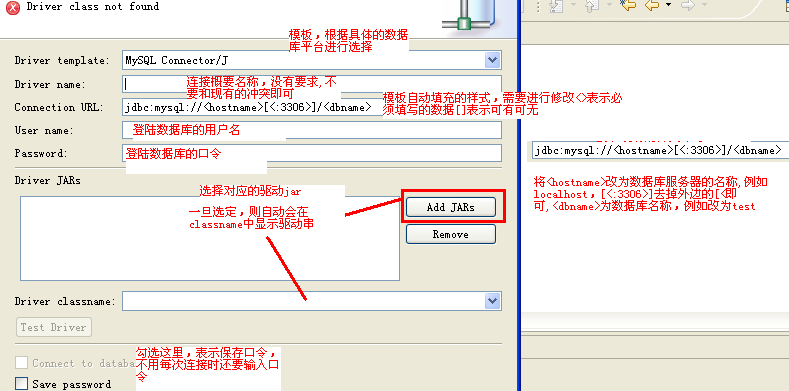
实现一个默认的构造方法（constructor）。这样的话Hibernate就可以使用Constructor.newInstance()来实例化它们

为类的持久化类字段申明访问方法（get/set）。Hibernate对JavaBeans风格的属性实行持久化

总结开发步骤:

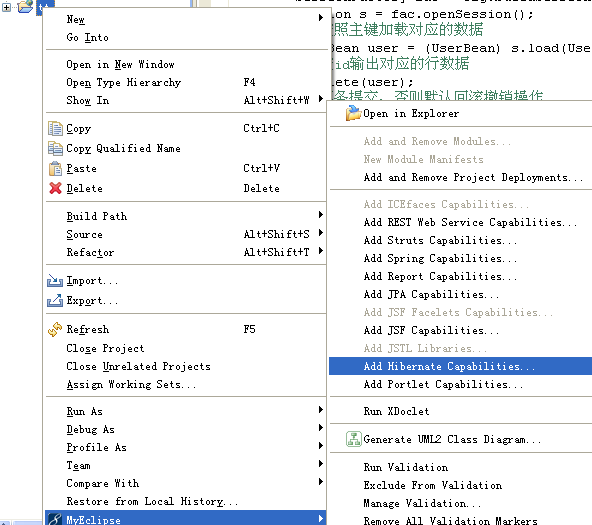
1\使用IDE中MyEclipse DB explorer创建连接概要

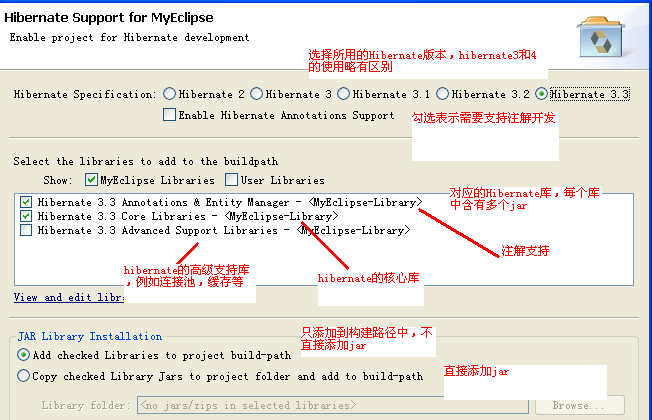


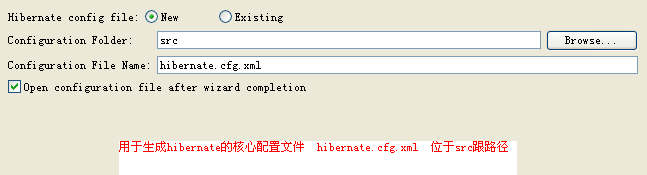


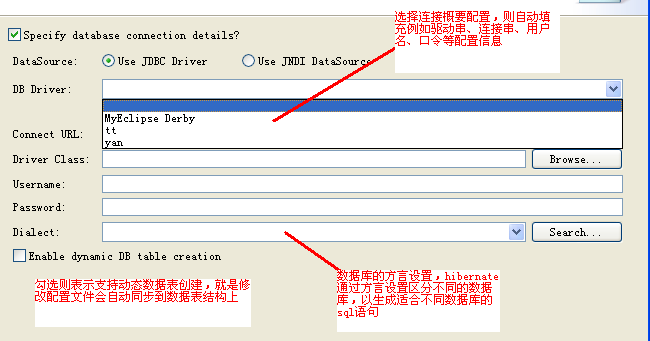
数据库和数据表的创建

2\使用IDE导入hibernate框架





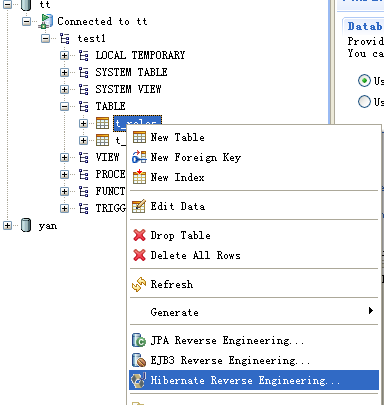


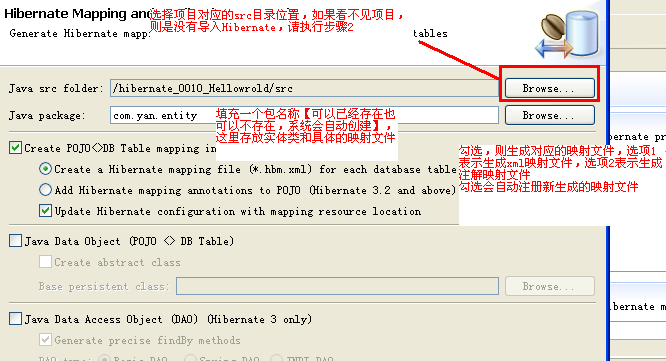


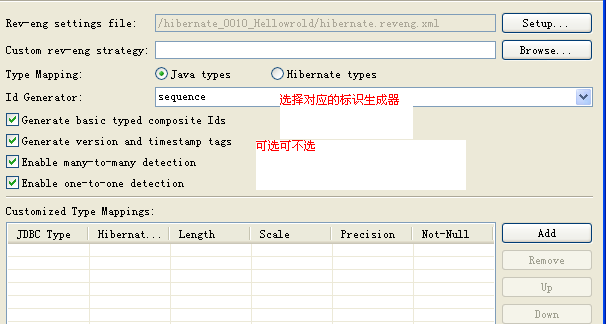
|  |
| --- |
| <?xml version=*'1.0'* encoding=*'UTF-8'*?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <!--hibernate的核心配置文件，主要包括3样配置信息： 数据库连接相关配置 hibernate的运行时常量 注册映射元文件-->  <hibernate-configuration>  <session-factory>  <!-- 数据库连接相关配置 -->  <property name=*"connection.url"*>jdbc:oracle:thin:@localhost:1521:ORCL</property>  <property name=*"connection.driver\_class"*>oracle.jdbc.OracleDriver</property>  <property name=*"connection.username"*>yanjun</property>  <property name=*"connection.password"*>123456</property>  <!-- 数据库的方言设置，hibernate根据方言设置区分不同数据库，从而针对不同数据生成对应的SQL语句 这个配置是必须-->  <property name=*"dialect"*>org.hibernate.dialect.Oracle9Dialect</property>    <!-- 定义hibernate的运行时常量，比如是不是使用缓存、连接池等 -->  <property name=*"hbm2ddl.auto"*>update</property><!-- 自动同步映射元文件和表结构 对映射元文件的修改会同步到表结构上 -->  <property name=*"myeclipse.connection.profile"*>yan</property><!-- 连接概要名称，可以有可以没有 -->  <property name=*"show\_sql"*>true</property><!-- 在控制台上输出所执行的sql语句，一般用于测试开发阶段，产品发布时必须修改 -->  <property name=*"format\_sql"*>true</property><!-- 在控制台上格式化输出sql -->    <!-- 注册映射元文件 -->  <mapping resource=*"com/yan/entity/UserBean.hbm.xml"* />  </session-factory>  </hibernate-configuration> |

3\使用IDE工具进行反向映射用于生成对应的映射元文件

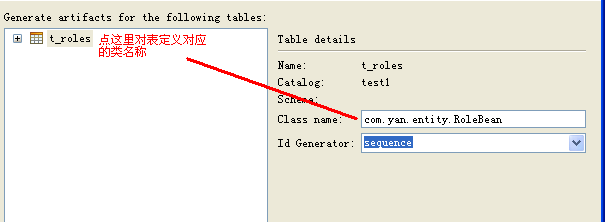
在MyEclipse DB Explorer中







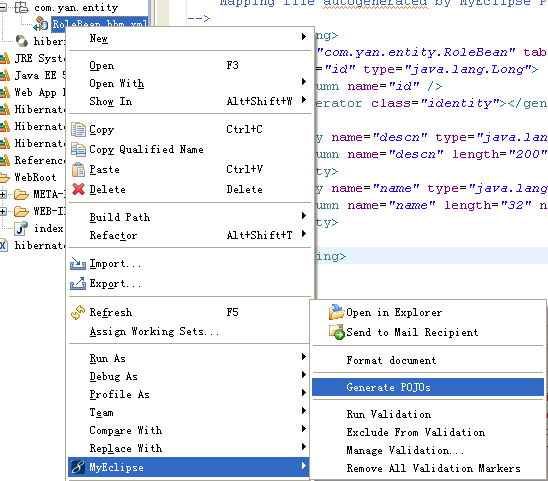
如果使用的mysql中的auto\_increment, 这里需要选择identity； 如果使用Oracle这里选择使用sequence，并在数据库中create sequence。



|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <!--  映射元文件用于说明类和表之间的对应关系 <class> name="实体类的全名" table="对应的表名" **schema="模式"建议删除**  -->  <hibernate-mapping>  <class name=*"com.yan.entity.UserBean"* table=*"T\_USRES"*>  <!--  用户说明类中的标识属性和主键之间的对应关系  <id>用户说明标识属性 name="属性名称" type="属性类型"  <column>用于说明所对应的列 name="列名称" 后面还有一些约束说明  例如 presicion="总位宽" scale="小数位数"  -->  <id name=*"id"* type=*"java.lang.Long"*>  <column name=*"ID"* precision=*"18"* scale=*"0"* />  <!-- 标识生成器 class="sequence"表示标识值是由序列生成的 默认序列名称为hibernate\_sequence 可以配置序列名称 -->  <generator class=*"sequence"*>  <param name=*"sequence"*>seq\_users</param>  </generator>  </id>  <!--  property用于说明非键属性 name="属性名" type="类型"  <column>用于说明该属性对应的列 name="列名称" 后面带有一些约束说明  比如 length="字串长度" not-null='true'该列非空 unique="true"有唯一性约束  -->  <property name=*"username"* type=*"java.lang.String"*>  <column name=*"USERNAME"* length=*"20"* not-null=*"true"* unique=*"true"* />  </property>  </class>  </hibernate-mapping> |

如果需要，手动修改生成的映射元文件

修改完成后，右键点击映射元文件生成对应的实体类



POJO意思是普通的古老的Java对象

私有属性，共有的get/set方法

无参构造器

具体操作

Configuration cfg = **new** Configuration().configure();用于读取并解析配置文件,充当SessionFactory的工厂

SessionFactory fac = cfg.buildSessionFactory(); Session的工厂

Session s = fac.openSession();充当实体管理器的功能，提供了最基本的”C增R删U改D删除”操作

Transaction tx = s.beginTransaction();hibernate默认事务回滚，如果需要修改生效，则需要通过tx进行手动提交

s.save(obj)存储 s.delete(obj)删除 s.update(obj)修改

s.load(UserBean.class,id):Object /get 按id查询

Query q=s.createQuery(“hql”);

List list=q.list();返回多行数据

**作业： 熟悉以上的操作步骤**

hibernate开发方式的三种方式:

1\直接编写实体类，再向类上追加注解配置，再由实体类生成数据表 ORM

2\直接编写xml映射，再生成实体类和表

**3\先创建表，然后通过反向工程工具生成配置和实体类[具体使用较多]**

EJB—JPA:java持久化API 是sun针对各种ORM框架提出的一套规范,目前主要用于EJB3的实体Bean中

提供JPA的实现:

Hibernate（开源免费） TopLink(收费) EclipseLink OpenJPA

Hibernate中使用注解开发实际上就是使用JPA中定义的注解