**Hibernate初探之单表映射**

## 第1章 Hibernate初识

### 1.2 什么是ORM

1.ORM

Objection/Relationship Mapping，即对象/关系映射。

2.背景介绍

利用面向对象思想编写的数据库应用程序最终都要把对象信息保存到关系型数据库中，于是要编写很多和底层相关的SQL语句，这不是面向对象编程思想锁需要的。另外，编写和底层相关的SQL语句也会有一定的麻烦，因为以下原因：

（1）不同的数据库使用的SQL语法版本不同，例如PL/SQL（Oracle数据库采用）和T/SQL（微软采用）；

（2）同样的功能在不同的数据库中有不同的实现方式，例如分页SQL；

（3）程序如果过于以来SQL语句，对程序的移植、扩展和维护将会带来很大麻烦。

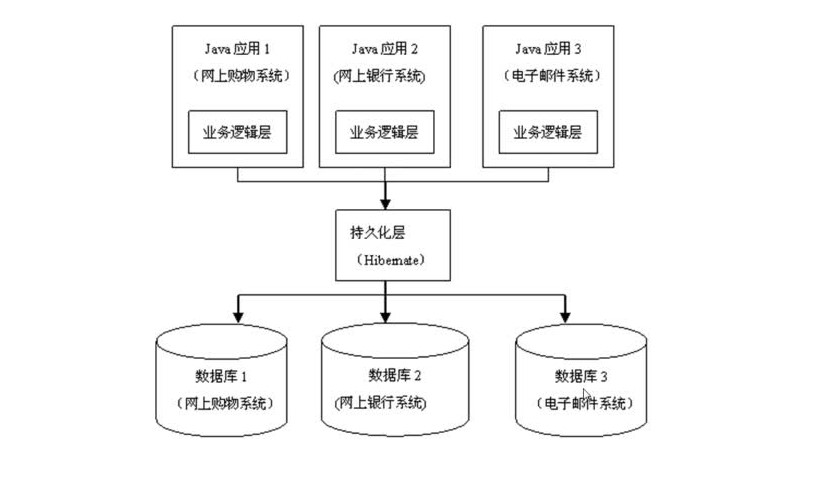
基于以上的原因分析，我们需要一种技术方案使得编写程序时抛弃SQL语句的书写，完全使用面向对象思想。

3.Hibernate是一种可行的可以解决上述问题的ORM框架技术。

### 1.3 Hibernate简介

1.Hibernate是Java领域的一款开源的ORM框架技术。Hibernate对JDBC进行了非常轻量级的对象封装。

2.Hibernate在程序中的作用



3.其他主流的ORM框架技术

（1）MyBatis：前身是iBatis；

（2）Toplink：后被Oracle收购，成为Oracle AS Toplink；

（3）EJB：本身是JavaEE的规范，重量级的框架技术，使用较少。

### 1.5 编写第一个Hibernate例子

1.Hibernate例子程序上需要的要素

（1）创建Hibernate的配置文件

配置文件名称是hibernate.cfg.xml。

（2）创建持久化类；

（3）创建ORM映射文件；

（4）通过Hibernate API编写访问数据库的代码

2.需要导入的包

（1）导入Hibernate必需的JAR包：

hibernate-release-4.2.4.Final\lib\required

（2）导入MySQL的JDBC驱动

mysql-connector-java-xxx-bin.jar

（3）导入Junit4的包（Junit是Java语言的单元测试框架）

junit-4.10.jar

### 1.6 创建Hibernate工程

1.为了避免以后新建工程都要重复导入所需JAR包，可以在Eclipse层面添加User Libraries。

※Windows—Preferences—Java—Build Path—User Libraries，先点击“New”来添加新的命名，然后点击“Add External JARs”来添加相应的JAR包。

添加完User Libraries后，就可以在项目层面上直接添加那些自定义的User Library，非常方便：

※在项目上右键—Properties—Java Build Path—Libraries选项卡—Add Library—User Library，然后选择上一步自定义的User Library即可。

### 1.7 Hibernate配置文档

1.右键新工程下的“src”包，New—Other—Hibernate—Hibernate Configuration File(cfg.xml)—Next，即可在src目录下生Hibernate成配置文档（文档名称默认是hibernate.cfg.xml）。

2.在配置文档中的<session-factory>节点中添加如下配置：

|  |
| --- |
| <property name=*"connection.username"*>scott</property>  <property name=*"connection.password"*>tiger</property>    <property name=*"connection.driver\_class"*>com.mysql.jdbc.Driver</property>  <property name=*"connection.url"*>jdbc:mysql://127.0.0.1:3306/hibernate?useUnicode=true&amp;  characterEncoding=UTF-8</property>  <property name=*"dialect"*>org.hibernate.dialect.MySQLDialect</property>  <property name=*"show\_sql"*>true</property>  <property name=*"format\_sql"*>true</property>  <property name=*"hbm2ddl.auto"*>create</property> |

【说明】

（1）在这里配置的是对数据库的连接信息，包括连接数据库的用户名、密码、驱动类、连接地址、使用Unicode字符集（防止中文乱码）、方言等。

连接地址中，MySQL服务如果是本机默认的地址和端口，则也可以使用：

jdbc:mysql:///hibernate?useUnicode=true&amp;characterEncoding=UTF-8

### 1.8 创建持久化类

1.创建持久化类Students类

持久化类的设计原则要符合JavaBeans的设计原则，原则如下：

（1）公有的类；

（2）提供公有的不带参数的默认的构造方法；

（3）属性私有；

（4）属性使用getter/setter封装。

### 1.9 创建对象-关系映射文件和数据库

1. 右键新工程下的“src”包，New—Other—Hibernate—Hibernate XML Mapping file(hbm.xml)—Next，即可在src目录下生成对象-关系映射文件（默认为Students.hbm.xml，根据持久化类而得名）。

在映射文件中，对持久化类的类名和属性映射到数据库的表和字段，之后建立数据库可根据此映射文件来建立。

2.创建完对象-关系映射文件后，我们需要将此文件添加到Hibernate配置文件。

在hibernate.cfg.xml文件的<session-factory></session-factory>中继续添加：

|  |
| --- |
| ***<mapping resource="Students.hbm.xml"/>*** |

3.根据对象关系映射文件建立数据库（这里只建立数据库，在后面的实验中，数据表和字段Hibernate会自动建立）。

### 1.10 使用JUnit进行单元测试

1.JUnit中三个常用注解：

（1）@Test：表示被注解的是个测试方法；

（2）@Before：表示被注解的是初始化方法，即在执行测试方法之前会执行该方法；

（3）@After：表示被注解的方法用来释放资源，执行完测试方法后来执行此方法。

2.在工程目录下新建test目录（类型是Source Folder，表示源程序文件夹），并新建测试类StudentTest.java，使用JUnit进行单元测试。

|  |
| --- |
| **StudentTest.java** |
| **public** **class** StudentsTest {  @Before  **public** **void** init(){    }    @After  **public** **void** destroy(){    }    @Test  **public** **void** testSaveStudent(){    }  } |

### 1.11 通过Hibernate API编写访问数据库的代码

1.测试类的代码示例

|  |
| --- |
| //测试类，使用JUnit进行单元测试  **public** **class** StudentsTest {  **private** SessionFactory sessionFactory;  **private** Session session;  **private** Transaction transaction;  @Before  **public** **void** init(){  //创建配置对象  Configuration config = **new** Configuration().configure();  //创建服务注册对象  ServiceRegistry serviceRegistry = **new** ServiceRegistryBuilder().applySettings(config.getProperties()).buildServiceRegistry();  //创建会话工厂对象  sessionFactory = config.buildSessionFactory(serviceRegistry);  //创建会话对象  session = sessionFactory.openSession();  //开启事务  transaction = session.beginTransaction();  }    @After  **public** **void** destroy(){  transaction.commit();//提交事务  session.close();//关闭会话  sessionFactory.close();//关闭会话工厂  }    @Test  **public** **void** testSaveStudent(){  Students s = **new** Students();  s.setSid(1);  s.setName("Tom");  s.setGender("男");  **try** {  s.setBirthday(**new** SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss").parse("1994年06月13日 05:00:00"));//日期时间格式的转换  } **catch** (ParseException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  s.setAddress("上海市闵行区");  //保存到数据库中  session.save(s);  }  } |

【说明】

此处的测试类单元测试时会自动在数据库中生成数据表并插入数据。

## 第2章 Hibernate进阶

### 2.2 Hibernate常用配置

1.在1.7节中我们配置了Hibernate配置文档，除了必需的对数据库的连接的配置外，还有以下常用的配置。

|  |  |  |
| --- | --- | --- |
| 属性名字 | 含义 | 补充说明 |
| hibernate.show\_sql | 是否把Hibernate运行时的SQL语句输出到控制台，在编码阶段方便调测 |  |
| hibernate.format\_sql | 输出到控制台的SQL语句是否进行排版，方便阅读 |  |
| hbm2ddl.auto | 可以帮助由Java代码生成数据库脚本，进而生成具体的表结构。可选参数：  create|update|create-drop|validate | create：新建表  update：更新表  create-drop：先创建后删除  validate：若现在的表结构与原表不同，则不创建 |
| hibernate.default\_schema | 默认的数据库前缀 | 在创建数据表的时候添加前缀 |
| hibernate.dialect | 配置Hibernate数据库方言，Hibernate可针对特殊的数据库进行优化 |  |

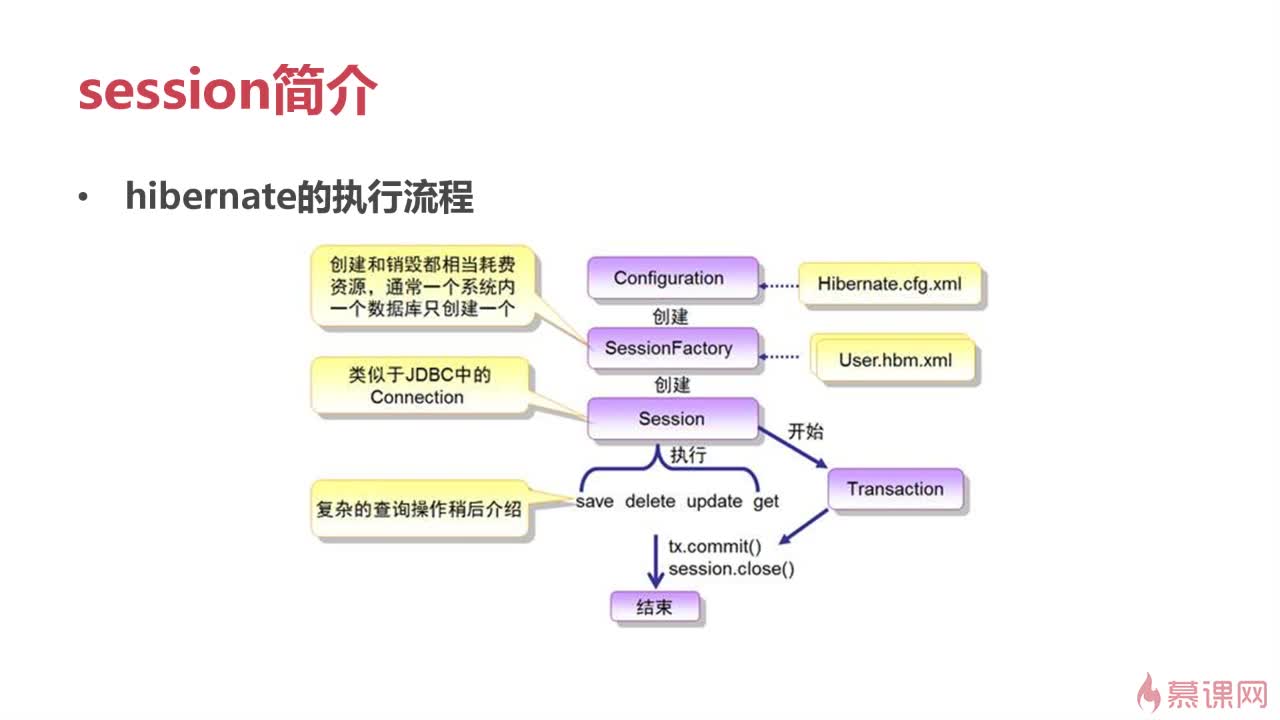
【说明】

①属性名字里的“hibernate”可以省略。

②在1.11节中进行单元测试时，每一次会自动在数据库中生成数据表并插入数据，这是因为Hibernate配置文档中的表生成策略hbm2ddl.auto配置为“create”导致的，可以更改为“update”，则可以保存原数据并只更新数据。

### 2.3 Session简介

1.Hibernate执行流程



【说明】

在执行Session对象的方法（对数据库进行增删改查）时，需要开启一个事务，即需要将方法封装在事务中。方法执行结束后，提交事务，再关闭Session会话。

2.Session简介

Hibernate是对JDBC的封装，所以不建议直接使用JDBC的connection对象操作数据库，而是通过Session的对象操作数据库，所以可以理解为Session的对象是操作数据库的对象，在对数据库执行操作之前就应该获得一个Session的实例对象。

Session和Connection是多对一的关系。每个Session都有一个与之对应的Connection，一个Connection可以同时提供多个Session使用。

3.把对象保存到关系数据库中，需要调用Session的方法，例如：

save()，update()，delete()，createQuery()等。

### 2.4 Transaction简介

1.Hibernate对数据库的操作都是封装在事务当中，并且默认是非自动提交的。所以例如用Session保存对象时，如果不开启事务，以及手动提交事务，那么对象不会真正保存到数据库中（***？？这是指对数据库的所有操作还是针对保存对象这一操作？？***）。

2.若想让Hibernate像JDBC那样自动提交，需要调用Session的doWork()方法获得JDBC的Connection后，将其设置为自动提交模式（**注意：不建议这样做**）。

操作如下：

|  |
| --- |
| //若不使用事务，设置session的自动提交模式  //生成Work实现类的匿名实例对象  session.doWork(**new** Work(){  **public** **void** execute(Connection connection) **throws** SQLException {  // **TODO** Auto-generated method stub  connection.setAutoCommit(**true**);  }    });  session.save(s);  //强制刷新  session.flush(); |

### 2.5 Session详解（上）

1.如何获得Session的对象？

可使用SessionFactory的两个方法均可获得：

（1）openSession()；

（2）getCurrentSession()；

2.

使用openSession()方法已经在前面的代码示例中展示了；

使用getCurrentSession()方法时，使SessionFactory的对象调用getCurrentSession()方法；另外需要在hibernate.cfg.xml配置文件中进行如下配置：

（1）若是本地事务（又称JDBC事务）：

***<property name="hibernate.current\_session\_context\_class">***

***thread***

***</property>***

（2）若是全局事务（又称jta事务）：

***<property name="hibernate.current\_session\_context\_class">***

***jta***

***</property>***

### 2.6 Session详解（下）

1.openSession和getCurrentSession的区别

（1）getCurrentSession在事务提交或回滚后会自动关闭，而openSession不会自动关闭，需要手动关闭，若使用openSession而没有关闭，多次之后会导致连接池溢出。

（2）openSession每次创建新的Session对象，而getCurrentSession使用现有的Session对象（类似单例模式，使用同一个对象）。

【说明】

根据上述两点区别，getCurrentSession会自动关闭session，释放掉对数据库的连接，那么下次取得session时，会重新使用之前释放掉的连接，这样资源消耗相对低很多。而使用openSession是需要手动关闭session的，这样问题也不大，可能产生问题的地方在于若是没有手动关闭，那么每次都会创建新的session，并使用新的数据库连接（可使用connection.hashCode()来比较不同的Connection连接对象），那么总会将资源耗尽的。

### 2.7 hbm配置文档

1.<hibernate-mapping />有哪些属性

|  |
| --- |
| ***<hibernate-mapping***  ***schema=”schema\_name”//设置模式名称***  ***catalog=”catalog\_name”//设置目录名称***  ***default-cascade=”cascade\_style”//默认的级联风格***  ***default-access=”field|property|ClassName”//默认的访问策略***  ***default-lazy=”true|false”//加载策略***  ***package=”package\_name”//设置默认的包名***  ***/>*** |

2.<class />有哪些属性

|  |
| --- |
| ***<class***  ***name=”class\_name”//对象关系映射中映射到哪个类***  ***table=”table\_name”//这个类映射到哪个表***  ***batch-size=”N”//抓取策略，一次可以抓取多少条记录***  ***where=”condition”//设置抓取记录的条件***  ***entity-name=”EntityName”//表示一个类映射到多张表***  ***/>*** |

3.<id></id>有哪些属性

|  |
| --- |
| ***<id name=”property\_name”//表示映射的属性***  ***type=”type\_name”//上面属性的数据类型***  ***>***  ***column=”column\_name”//映射到表的字段***  ***length=”length”//指定其长度***  ***<generator class=”generator\_class” />//主键生成策略***  ***</id>*** |

【说明】

<id>标签是指在数据表中是主键的字段（以及对应属性）。

4.**主键生成策略**

使用MySQL数据库时，主要有native和assigned两种主键生成策略，其中应用更多的是native策略（自动选择），assigned是一种手动生成的策略。



## 第3章 Hibernate单表操作

### 3.2 单一主键

1.单一主键是指由表中的某一列充当主键。与其对应的是复合主键。

2.在MySQL中，有常见的两种主键生成策略：

（1）assigned：由Java应用程序负责生成（也可以理解为“手工赋值”）

这种策略需要开发者在程序中手动赋值，才能保证避免主键的冲突。

（2）native：由底层数据库自动生成标示符（其实是自动生成主键的值），其生成方法是和底层数据库相关的，如果是MySQL，则生成AUTO\_INCREMENT类型的，如果是Oracle则生成SEQUENCE类型的。

【注意】

①当使用assigned策略时，若未手动赋值主键，那么程序只会根据属性的数据类型给主键赋该数据类型的默认值（如属性student\_id对应主键，数据类型是int型，未手动赋值主键时程序只会赋值为0）。

②使用native时，若同时手动赋值主键，是不会产生效果的。

### 3.3 基本类型

1.在对象关系映射中的基本数据类型的对应关系



在对象关系映射配置文件（hbm.xml）中，有属性的映射标签（包括<id><property>），有**type属性**，即是指基本数据类型，但是我们通常要么写表中的“Hibernate映射类型”，要么写“Java类型”，而不写“标准SQL类型”。

2.从上表中选择容易出错的日期时间类型来说明。



例如在第1章的Hibernate案例中，属性birthday的映射关系如下：

|  |
| --- |
| ***<property name="birthday" type="java.util.Date">***  ***<column name="BIRTHDAY" />***  ***</property>*** |

因为数据类型写的是“***java.util.Date***”（是Java类型中的），所以最终插入到数据库中的总是日期、时间都有；如果只想插入日期，可以直接写Hibernate类型中的“***date***”。即改为：

|  |
| --- |
| ***<property name="birthday" type="date">***  ***<column name="BIRTHDAY" />***  ***</property>*** |

### 3.4 对象类型

1. 在对象关系映射中的对象类型的对应关系



【说明】

①Hibernate映射类型中，text和clob表示大文本数据类型，blob表示二进制数据类型，比如视频、音频、图片等。

②MySQL不支持标准SQL类型中的CLOB数据类型，在MySQL中用TEXT，MIDUMTEXT以及LONGTEXT类型来表示长度超过255的长文本数据。

### 3.5 组件属性

1.**组件属性**是指实体化类（数据持久化类）中的某个属性是用户自定义的类的对象。

2.组件属性需要在hbm.xml配置文件中作如下配置：

|  |
| --- |
| ***<component name="address" class="Address">***  ***<property name="postcode" column="POSTCODE"></property>***  ***<property name="phone" column="PHONE"></property>***  ***<property name="address" column="ADDRESS"></property>***  ***</component>*** |

①***<component>***标签即表示是组件属性，该属性对应的类是Address类，其内部也包含其他的属性。

### 3.6 单表操作CRUD

1.Hibernate执行单表操作需要使用**Session对象**调用以下方法执行操作。

|  |  |
| --- | --- |
| 方法 | 说明 |
| save(Object object) | 插入记录 |
| get(Class clazz, Serializable id) | 查询单条记录 |
| load(Class clazz, Serializable id) | 查询单条记录 |
| update(Object object) | 更新记录 |
| delete(Object object) | 删除记录 |

【说明】

Serializable是指序列化id，在get与load方法中需要使用记录中的主键id。（由此可见也许这是一种效率较低的查询方式）

2.get与load的**两点区别**

（1）在不考虑缓存的情况下，get方法会在调用之后立即向数据库发出SQL语句，并返回持久化对象。

load方法在调用后会返回一个代理对象，该代理对象只保存了实体对象的id，直到使用对象的非主键属性时才会发出SQL语句。

（2）查询数据库中不存在的数据时，get会返回null；

load方法会抛出异常：***org.hibernate.ObjectNotFoundException***。

## One More Thing

1. Hibernate Tools是JBoss推出的一个Eclipse综合开发工具插件，该工具可以简化ORM框架Hibernate、JBoss Seam以及EJB3等的开发工作。
2. **全局事务**：资源管理器管理和协调的事务，可以跨越多个数据库和进程。

**本地事务**：在单个EIS或数据库的本地并且限制在单个进程内的事务。本地事务不涉及多个数据来源。

3. 数据表中AUTO\_INCREMENT的字段是从1开始增长的，而不是0。

4. Hibernate在连接到数据库后在执行数据库操作之前，就会根据表的生成策略（***hbm2ddl.auto***）来更新数据表。