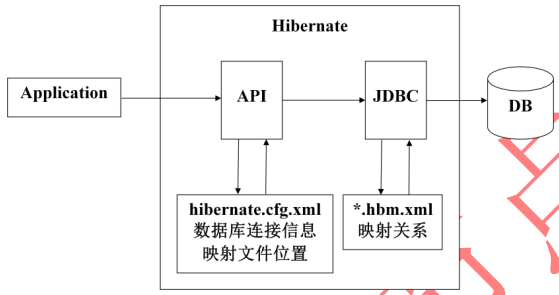
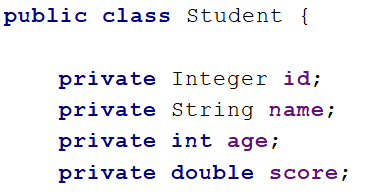
## 1、工作原理



## 2、简单示例

### 1）定义持久化对象

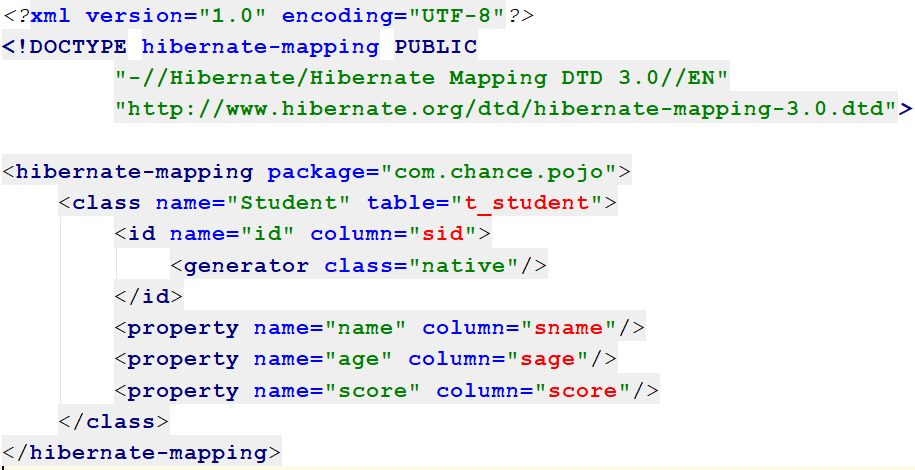


### 2）配置映射文件

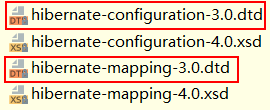
配置两个关系：

实体类——>数据库中的表

属性——>表中字段



Hibernate主要涉及两个配置文件。这两个配置文件的约束文件在hibernate-core-5.0.1.Final.jar中的or.hibernate包中。



hibernate.cfg.xml主配置文件的约束文件：

**<!DOCTYPE hibernate-configuration PUBLIC  
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd"*>***

Xxx.hbm.xml映射配置文件的约束文件：

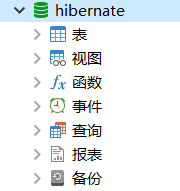
**<!DOCTYPE hibernate-mapping PUBLIC  
 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd"*>***

### 3）配置主配置文件

* 配置DB连接四要素与方言
* 配置数据源（不指定则使用Hibernate内置的数据源）
* 配置当前Session上下文（指当前Session上下文为线程，即在同一线程内所使用的Session为同一Session）
* 配置自动建表
* 配置控制台SQL输出
* 注册映射文件

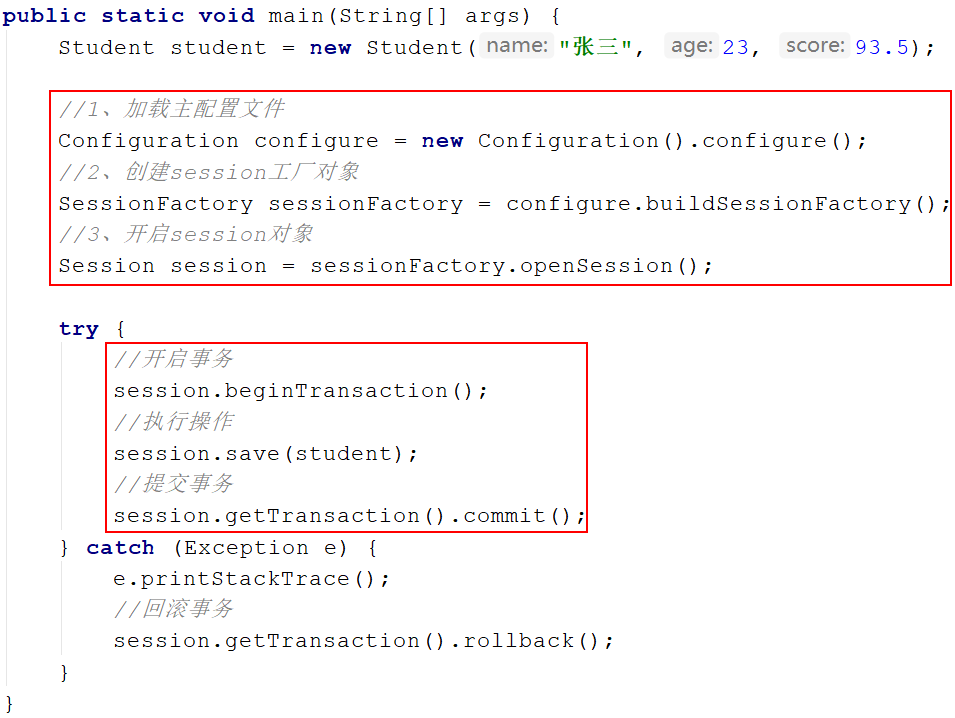


### 4）创建数据库hibernate



### 5）添加日志控制文件

### 6）测试



## 3、CURD操作

### 1）工具类创建

**public class** HibernateUtil {

**private static** SessionFactory *sessionFactory*;  
  
 */\*\*  
 \* 获取Session对象  
 \*  
 \** ***@param:*** *[]  
 \** ***@return:*** *org.hibernate.Session  
 \*/* **public static** Session getSession() {  
 **return** *getSessionFactory*().getCurrentSession();  
 }  
  
 */\*\*  
 \* 获取Session工厂  
 \*  
 \** ***@param:*** *[]  
 \** ***@return:*** *org.hibernate.SessionFactory  
 \*/* **public static** SessionFactory getSessionFactory() {  
 **if** (*sessionFactory* == **null** || *sessionFactory*.isClosed()) {  
 *sessionFactory* = **new** Configuration().configure().buildSessionFactory();  
 }  
 **return** *sessionFactory*;  
 }  
  
}

### 2）save()

持久化操作，save()为Hibernate的API。

### 3）persist()

同样为持久化操作，persist()为JPA的API。

### 4）delete()

删除，其底层是根据id进行删除，所以指定的删除对象必须要有id属性。

@Test

**public void** testDelete() {  
 Student student = **new** Student();  
 *//指定要删除的student的id* student.setId(1);  
  
 Session session = HibernateUtil.*getSession*();  
 **try** {  
 session.beginTransaction();  
  
 session.delete(student);  
  
 session.getTransaction().commit();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 session.getTransaction().rollback();  
 }  
}

### 5）update()

修改，其底层是根据id进行修改，所以必须要有id属性。

### 6）saveOrUpdate()

通过参数对象是否具有id值来判断是执行update()操作还是save()操作。若参数具有id，但该id在DB中不存在，则会抛出异常。

@Test

**public void** testSaveOrUpdate() {  
 Student student = **new** Student(**"李四"**, 24, 95.5);  
 *//设置要修改的对象的id。该语句决定执行save()还是update()* student.setId(14);  
  
 Session session = HibernateUtil.*getSession*();  
 **try** {  
 session.beginTransaction();  
  
 session.saveOrUpdate(student);  
  
 session.getTransaction().commit();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 session.getTransaction().rollback();  
 }  
}

### 7）get()、load()

get()在查询无果时，会返回null，不会抛出异常。

load()在查询无果时，会抛出异常。

### 8）对于增、改、删的底层SQL执行顺序

对于不同操作对象的增、改、删操作，无论其代码的书写顺序是怎样的，底层SQL执行顺序都是insert、update、delete。

若需要修改底层的执行顺序，可通过Sessinon的flush()方法刷新Session完成。flush()使其前后分隔为两部分，各部分会按增、改、删顺序执行。

