Hibernate是一个开放源代码的对象关系映射（O/RMapping）框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以自动生成SQL语句，自动执行，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。

Hibernate，翻译过来是冬眠的意思，正好现在已经进入秋季，世间万物开始准备冬眠了。其实对于对象来说就是持久化。

持久化（Persistence），即把数据（如内存中的对象）保存到可永久保存的存储设备中（如磁盘）。持久化的主要应用是将内存中的对象存储在关系型的数据库中，当然也可以存储在磁盘文件中、XML数据文件中等等。

持久化是将程序数据在持久状态和瞬时状态间转换的机制。

JDBC就是一种持久化机制。文件IO也是一种持久化机制。

我们可以看出Hibernate六大核心接口，两个主要配置文件，以及他们直接的关系。Hibernate的所有内容都在这了。那我们从上到下简单的认识一下，每个接口进行一句话总结。

1、Configuration接口:负责配置并启动Hibernate

2、SessionFactory接口:负责初始化Hibernate

3、Session接口:负责持久化对象的CRUD操作

4、Transaction接口:负责事务

5、Query接口和Criteria接口:负责执行各种数据库查询

//读取hibernate.cfg.xml文件

Configuration cfg = new AnnotationConfiguration().configure();

//建立SessionFactory

SessionFactory factory =cfg.buildSessionFactory();

//取得session

Session session = null;

try{

//开启session

session = factory.openSession();

//开启事务

session.beginTransaction();

User user = new User();

user.setName("jiuqiyuliang");

user.setPassword("123456");

user.setCreateTime(new Date());

user.setExpireTime(new Date());

//保存User对象

session.save(user);

//提交事务

session.getTransaction().commit();

}catch(Exception e){

e.printStackTrace();

//回滚事务

session.getTransaction().rollback();

}finally{

if(session != null){

if(session.isOpen()){

//关闭session

session.close();

}

}

}

Hibernate的几种关联形式：

一对一关联（很少出现，出现也使用外键关联）：

1）单向外键关联：比如husband和wife，在husband中有wife属性，在wife属性的get方法上注解@OneToOne，该注解的一些可以设置的条件@JoinColumn（name=“。。外键名称。。”）可以设置在husband中设置外键的名称

2）双向外键关联：在husband和wife里面各自加上对面的属性，并且都是用@ OneToOne注解，这是两张表会产生两个外键，这是冗余的信息，所以需要额外设置，可以在wife的类里加上（此处怀疑是否在huaband里加也会有同样的效果，未做测试）@ OneToOne（mappedBy=”wife”）可以告诉谁做了映射（Husband类里的wife属性上做了映射），此时，wife表里面就没有外键了。一般双向关联必设mappedBy

3） 组件映射，对于两张一对一的表，可以将他们在数据库里合成一张表，可以在husband类里的wife属性前加入注解@Embedded，此时应当让husband和wife类的属性名都不一样，或者@Column修改表里的字段名

多对一关联

1）多对一单向关联：在多方加外键，就类似于一个组有很多人，那么关联的方法是在人的属性上加一个组名，而不是在组里面加人名（意思是一个组会被分成多条记录，每条记录的任命不一样，组名相同）此时使用注解@ManyToOne会自动在One的表格里添加Many\_Id的字段

2）一对多单向关联：在一方建立set<多方>去存储数据，，使用@OneToMany注解会生成中间表，被认为是多对多，此时需要在下面注解@JoinColumn（name=“。。。”）

！！！无论是一对多还是多对一，这两种注解在数据库里生成的表是一样的，都只会在多方加外键，所以一对多的在一方（Group）下面给set<User>加的注解@JoinColumn（name=“groupId”）中的字段名是groupId（相对于User和group来说）

3）一对多双向关联（等于多对一双向关联）：就是在Group和User里面分别设置@OneToMany和@ManyToOne，此时不需要给@OneToMany设置@JoinColumn，否则会产生冗余，但是需要在@OneToMany设置（mappedBy=group）来确立映射主体，这里的group是User里的group属性

4）多对多单向关联：比如学生和老师，单向关联是指老师知道要教多少学生，所以老师对象里面有一个set<Student>，而学生不需要知道老师是谁，没有老师的相关信息，此时应该在set,<Student>上注解@ManyToMany那么会生成一张关系表Teacher\_Student,里面会有两个字段Teacher\_id和students\_id,这里的students是Teacher的一个属性名，若要自定义这个表的一些名字，可以用

@JoinTable（name=””，

joinColumns{@JoinColumn（name=“teacher\_id“）},

inverseJoinColumns={@JoinColumn（name=“student\_id“）}

）

这里有JoinColumns{@JoinColumn（name=“teacher\_id“）}指定了关系表里参考了该对象id（代码所在的对象）的外键名称，而inverse则是说明了teacher对应的另一张表的一些情况

5）多对多双向关联：学生知道老师，老师也知道学生（一般情况下双向是很少见的）

这时候就需要在多对多单向的基础上在另一个类中也建立对应的set属性，这里是在Student类里面建立set<Teacher>，并且注解上@ManyToMany（mappedBy=“students“）这里的student是指Teacher类里的Set<Student> students属性