# Hibernate技术栈

Hibernate是一个开源的轻量级框架，所谓轻量级即不需要依赖其他。

JavaEE的三层架构

1. Web层，struts2框架；
2. Service层,Spring框架；
3. dao（data access object）层,Hibernate框架，对数据库进行crud操作；

## Hibernate入门

### 搭建hibernate环境

1. 创建项目，java项目和web项目都可以；
2. 导入jar包，lib文件夹里面的required文件夹，jpa文件夹（规范）和其他的日志jar包，这个日志jar包并不是Hibernate本身的，是外面的,包括log4j-1.2.16、slf4j-api-1.6.1和slf4j-log4j12-1.6.1。导入java的规范是先在项目下建一个文件夹，在右键选中build path,笔者经过多次的尝试，发现使用该方法导入jar包时，只能导入到referance library里面，程序运行时改文件夹下的jar包不能被自动加载。所以改规范不可行，最好的办法还是直接放在WEB-INF文件夹下咋ilb文件夹里面，程序自动把jar包放在Web App Libraries文件夹里面，在运行是实现自动加载。

### 创建实体类

1. 要求有一个属性是唯一的，比如ID；
2. 要有get和set的方法；
3. 使用hibernate时不需要手动建表，其可以自动帮我们创建；

### 配置实体类和数据库表的映射关系

1. 创建xml格式的配置文件，虽然配置文件的名称和位置没有固定要求，但是建议把xml文件放在实体类的包里面，名称和实体类的名称+“.hbm.xml”；
2. 配置时首先引入xml约束（如dtd/schema）,其实在下载包\project\hibernate-core\out\production\resources\org\hibernate里面的hibernate-mapping.dtd有的：

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

1. 配置映射关系，注意属性name的值要和实体类中的属性名称要一致

<hibernate-mapping>

<class name="com.lvhongbin.bean.User" table="t\_User">

<id name="id" column="id">

<generator class="native"></generator>

</id>

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

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

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

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

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

</class>

</hibernate-mapping>

### 创建Hibernate中的核心配置文件

1. 核心配置文件的格式为.xml，但是核心配置文件的名称和位置是固定的。

位置：必须在src下面

名称：必须为hibernate.cfg.xml

1. 配置时首先引入xml约束（如dtd/schema）,其实在下载包\project\hibernate-core\out\production\resources\org\hibernate里面的hibernate-configuration-3.0.dtd有的：

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

1. hibernate在操作过程中，只会加载核心的配置文件，其他的配置文件不会进行加载。
2. 配置数据库信息（可以参考property文件project\etc\hibernate.properties）,必须要有；
3. 配置hibernate信息，可以不写；
4. 引入映射文件，具体的代码如下：

<hibernate-configuration>

<session-factory>

<!-- 数据库驱动 -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<!-- 数据库连接的URL 数据库需要手动创建 -->

<property name="connection.url">jdbc:mysql://localhost:3306/jsp\_db?useSSL=true</property>

<!-- 数据库连接用户名 -->

<property name="connection.username">root</property>

<!-- 数据库连接密码 -->

<property name="connection.password">12345687</property>

<!-- Hibernate方言 旧版的方言为org.hibernate.dialect.MySQLDialect-->

<property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>

<!-- 打印SQL语句 -->

<property name="show\_sql">true</property>

<!-- 设置自动建表 -->

<property name="hbm2ddl.auto">update</property>

<!-- 映射文件 -->

<mapping resource="com/lvhongbin/bean/User.hbm.xml"/>

</session-factory>

</hibernate-configuration>

### 实现添加的功能

1. 加载hibernate核心配置文件；
2. 创建SessionFactory对象；
3. 使用SessionFactory创建session对象；
4. 开启事务；
5. 写具体的逻辑crud操作；
6. 提交事务；
7. 关闭资源；

代码如下：

public void Test() {

Configuration cfg=new Configuration();

cfg.configure();

SessionFactory sessionFactory=cfg.buildSessionFactory();

Session session=sessionFactory.openSession();

Transaction tx=session.beginTransaction();

User user=new User();

user.setName("lvhongbin");

user.setPassword("12345687");

user.setSex("male");

session.save(user);

tx.commit();

session.close();

sessionFactory.close();

}

## Hibernate入门注意事项

### Hibernate配置文件详解

1. 映射文件的名称和位置没有固定，一般放在对应实体类的包里面；
2. 映射中的name属性的值要与对应实体类的属性相同，column属性与表中的字段名称相同；
3. 如果不写column值，则column值与name值一致；
4. property标签中的type属性，用来设置表字段的类型；
5. 核心配置文件的格式为.xml，但是核心配置文件的名称和位置是固定的：

位置：必须在src下面

名称：必须为hibernate.cfg.xml

1. 核心配置文件的数据库部分是必须的，hibernate部分时=是可选的，映射文件是必须的；

### Hibernate核心API的说明

1. Configuration

Configuration cfg = new Configuration();

cfg.configure(); //到src下面找到名称为hibernate.cfg.xml的配置文件，官方称“加载核心配置文件”

1. SessonFactory

使用Configuration对象创建SessonFactory对象，然后根据核心配置文件进行连接或者创建数据表。如果每次操作都进行连接或者创建，则特别耗资源，所以在hibernate操作中建议一个项目只创建一个SessonFactory对象。具体实现：写一个工具类（util），写静态代码块实现，代码如下：

public class HibernateUtil {

private final static Configuration cfg;

private final static SessionFactory sessionFactory;

static {

cfg = new Configuration();

cfg.configure();

sessionFactory=cfg.buildSessionFactory();

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

然后在事务操作中添加代码：

SessionFactory sessionFactory=HibernateUtil.getSessionFactory();

1. Session

类似与jdbc中的connection，调用session里面不同的方法实现crud操作：添加save方法，修改update方法，删除delete方法和跟据id查询get方法。session对象是单线程（不能共用，只能自己使用，注意进程和线程的区别）对象。

1. Transaction

事务对象，两个重要操作事务提交和事务回滚，tx.commit();和tx.rollback();事务有四个特性：原子性（一个事务操作要不全部成功，要么全部失败），一致性（数据操作前后总量不变），隔离性（多人同时操作不会相互影响）和持久性（数据在数据库提交后生效）。

### 解决配置文件中没有提示的问题

1. 可以上网；
2. 把约束文件引入项目，复制配置文件中的http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd，然后点击window -- Prefernce -- XML -- XML Catalog -- Add 在key中URI粘贴刚才复制的网址，然后在Location中寻找相应.dtd文件的路径；

### 实体类（持久化类）的编写规则

1. 属性私有
2. 私有属性需要有公开的get和set方法
3. 要求实体类中有一个属性作为唯一值（一般使用id值）
4. 实体类属性建议不使用基本数据类型，而使用基本数据类型对应的包装。8个基本数据类型,int对应的是integer，char对应的是Character，其他的都采用首字母大写，如string对应的是String，double对应的是Doube，其原因是基本数据类型没有null的值，而包装类有null值。

### 主键生成策略

1. increment 每次增加增量为1；
2. identity 条件是数据库支持自动增长，如Oracle就不支持自动增长；
3. sequence 条件是数据库支持序列，如MySql不支持序列；
4. nativ 跟据底层数据库对自动生成表示符的能力来选择identity，sequence，hilo三种生成器中的一种，适合跨数据库平台的开发，适用于代理主键；
5. uuid 实体类id属性类型必须为字符串类型

### 实体类操作

1. save()操作；

User user =new User();

user.setName(“lvhongbin”);

session.copy(user);

1. 跟据id查询

Session对象提供了两种对象装载的方法，分别是get()和load()。当开发人员不确定数据库是否有匹配的记录存在时，可以使用get()方法，如果没有匹配的记录存在，返回null。get()方法中含有两个参数，第一个是持久化对象，另一个是持久化对象的唯一标识值。其代码如下

User user =session.get(User.class, 1);

System.out.println(user);

load()方法返回对象的代理，只有在返回对象被调用的时候，Hibernate才会发出SQL语句去查询对象。这叫hibernate的延迟加载策略，即使用load()加载持久化对象时，它返回的是一个未初始化的代理（代理无须从数据库中抓取数据对象的数据）直到调用代理的某个方法时，hibernate才会访问数据库。当装载的对象长时间没有调用时，就会被垃圾回收器回收，在程序中合理使用延迟加载策略，可以优化系统的性能，节省内存空间，减少不必要的开支。

// sql语句 SELECT \* FROM t\_user t WHERE t.uid =1

User user =session.load(User.class, 1);

System.out.println(user);

或者在映射文件中的<property>标签的lazy属性改为“true”

<property lazy=“true”>

1. 修改

跟据id查询，返回对象

// UPDATE t-user SET username=?, address=? WHERE uid=?

第一种方法(建议)

User user =session.get(User.class, 1);

user.setName(“lvhongbin”);

session.update(user);

第二种方法(不建议,因为他会影响到别的参数，如果别的参数不修改的话会变成空值)

User user =new User();

user.setName(“lvhongbin”);

session.update (user);

第三种方法 强制刷新提交。session的刷出（flush）过程是指Session执行的一些必须的SQL语句来把内存中的对象状态同步到JDBC中，刷出会在某些查询之前执行，在事物提交时执行，或者在程序中直接调用flush（）时执行。

User user =new User();

user.setName(“lvhongbin”);

session.flush();

1. 删除

只有当对象在持久化状态时才能执行，所以在删除数据之前，首先将对象的状态转化为持久化状态。

//DELETE FROM t\_user WHERE uid=?

第一种方法(建议)

User user =session.get(User.class, 1);

session.delete(user);

第二种方法

User user = new User();

User.setUid(3);

session.delete(user);

1. saveOrUpdate()

当实体类对象为瞬时态时作save()的操作，当实体类对象为托管态或者持久态时作update()的操作；

### 实体类对象的状态

1. 瞬时态：对象里面没有id值，对象与session没有关联。只是通过new关键字开辟内存空间创建Java对象，或者通过删除操作删除数据库表某一对象时，没有纳入session的管理里面，如

User user =new User();

user.setName(“lvhongbin”);

或者：

User user =session.get(User.class, 1);

session.delete(user);

1. 持久态：在对象里面有id值，它总和会话状态（session）和事务（transaction）关联在一起。当持久化对象发生改动时并不会立即执行数据库操作，只有当事务结束时，才会更新数据库，以便保证Hibernate的持久化对象和数据库操作的同步性，如插入，查询和更新的操作：

User user =session.get(User.class, 1);

1. 托管状态：对象里面有id值，对象与session没有关联，close(),clear(), evict()

User user =new User();

user.setUid(3);

### Hibernate的一级缓存

1. 数据存到数据库里面，数据库本身是文件系统，使用流方式操作文件本身效率并不高，把数据存到内存里面，不需要使用流的方式，可以直接读取内存里面的数据，提高读取效率。
2. Hibernate框架中有很多优化的手段，其中缓存是其中一个手段，Hibernate有一级缓存和二级缓存，一级缓存是默认打开的，属于session级缓存，所以它的生命周期与session相同，随session创建而创建，随session销毁而销毁，而且存储数据必须是持久化数据。过程：首先，查询一级缓存发现没数据，才会去查询数据，返回查询对象（持久化对象），然后，把持久化对象的值放到一级缓存中；若查询一级缓存中存在相同的数据，直接返回。
3. 一级缓存的更新特性：持久态自动更新数据库（不需要写update方法）。过程：创建session对象时，自动创建一级缓存对象和快照区（副本），当持久态对象被创建时，持久态被放到一级缓存及其对应的快照区中

User user =session.get(User.class, 1);

user.setName(“lvhongbin”);

tx.commit(); //修改user对象里面的值，修改持久态对象的值的同时，修改一级缓存中的内容，但是不会修改对照区中的内容。最后提交事务的时候，比较一级缓存和快照区的内容是否相同，如果不相同，则把一级缓存中的内容更新到数据库里面，如果相同，则不更新。

1. Hibernate的二级缓存，目前已经不用了，替代技术为redis，默认是不打开的，需要配置，属于sessionFaction级缓存。

验证一级缓存的存在， 跟据uid=1查询，返回对象；再次根据uid=1查询，返回对象，通过断点debug查看。

### Hibernate的事务操作

1. 不考虑隔离产生的问题：脏读，不可重复读和虚读。

### Hibernate的其他api

JavaBean,更正确的叫法为实体类，拥有get和set的方法；