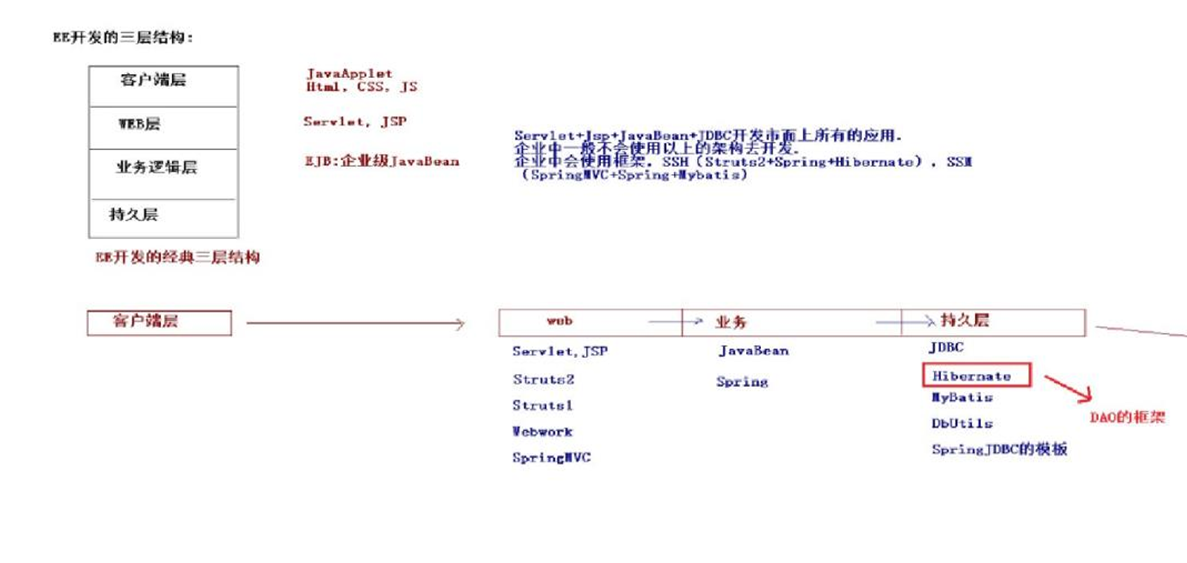
# Hibernate框架

JavaEE三层开发结构：



## ORM 规则

ORM 对象关系映射（英语：**(Object Relational Mapping**，简称**ORM**，或**O/RM**，或**O/R mapping**），是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换

\* O：面向对象领域的Object（JavaBean对象）

\* R：关系数据库领域的Relational（表的结构）

\* M：映射Mapping（XML的配置文件）

简单一句话：Hibernate使程序员通过操作对象的方式来操作数据库表记录

目标：

通过orm（对象关系映射）, 可以做到：

a. 不写sql，包对象直接保存到数据库中

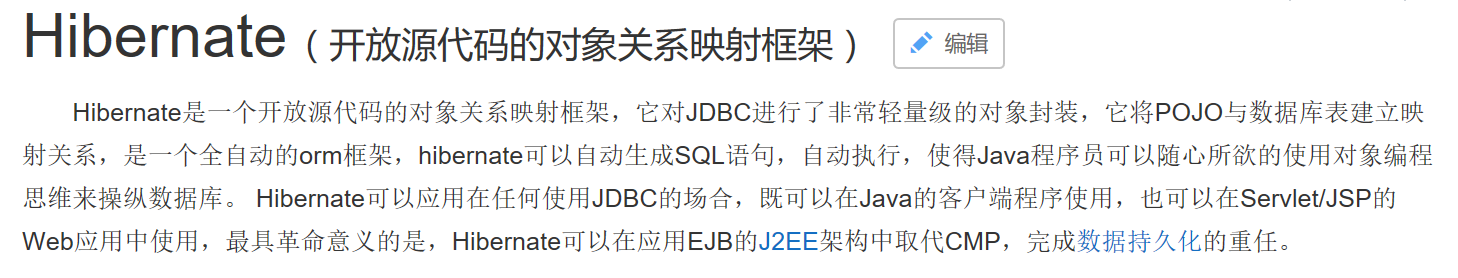
b. 不写sql， 从数据库中读取，直接获取的是封装好的对象！

ORM是一种思想，或者说是一种规则的定义， 基于orm的应用：

1. hibernate 框架

2. mybatis 框架

## Hibernate框架



### 概念

基于ORM的持久层框架，对jdbc操作进行了封装！

Hibernate与具体数据库无关的技术，即可以跨数据库平台！

**优点**

\* Hibernate对JDBC访问数据库的代码做了封装，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的性能非常好，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 种复杂关系

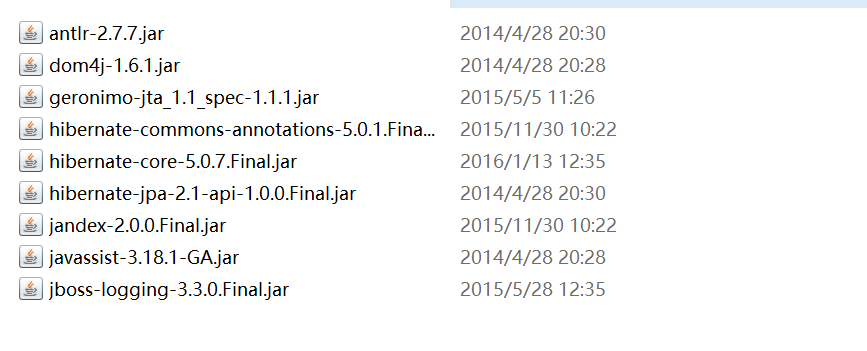
### Hibernate开发环境搭建

Hibernate版本：hibernate-release-5.0.7.Final

整体步骤：

1. 下载源码，引入jar文件





Log4j日志的三个包



数据库驱动包：



2. 写配置

\* hibernate.cfg.xml 主配置

\* Users.hbm.xml 映射配置

3.Api

Configuration

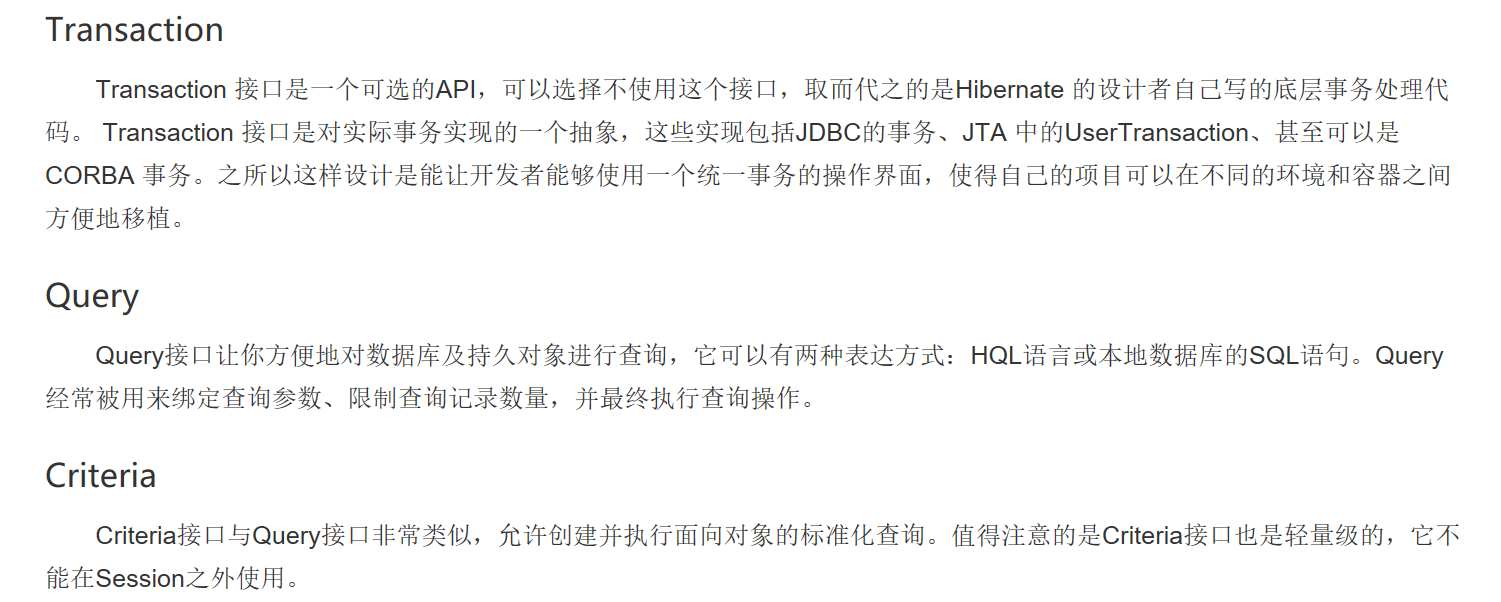
SessionFactory

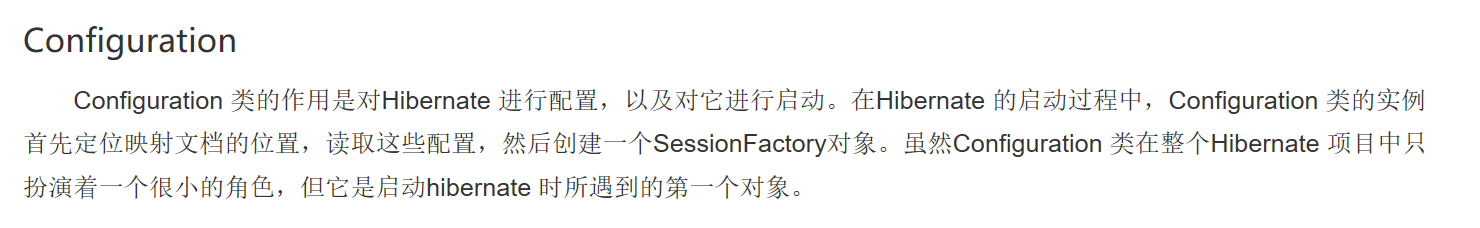
Transaction

Session

Query







## Hibernate快速入门

### 第一步：下载Hibernate5的运行环境

1. 下载相应的jar包等

http://sourceforge.net/projects/hibernate/files/hibernate-orm/5.0.7.Final/hibernate-release-5.0.7.Final.zip/download

2. 解压后对目录结构有一定的了解

### 第二步：创建表结构

1. 建表语句如下

|  |
| --- |
| Create database hibernate\_day01;  Use hibernate\_day01;  CREATE TABLE `cst\_customer` (  `cust\_id` bigint(32) NOT NULL AUTO\_INCREMENT COMMENT '客户编号(主键)',  `cust\_name` varchar(32) NOT NULL COMMENT '客户名称(公司名称)',  `cust\_user\_id` bigint(32) DEFAULT NULL COMMENT '负责人id',  `cust\_create\_id` bigint(32) DEFAULT NULL COMMENT '创建人id',  `cust\_source` varchar(32) DEFAULT NULL COMMENT '客户信息来源',  `cust\_industry` varchar(32) DEFAULT NULL COMMENT '客户所属行业',  `cust\_level` varchar(32) DEFAULT NULL COMMENT '客户级别',  `cust\_linkman` varchar(64) DEFAULT NULL COMMENT '联系人',  `cust\_phone` varchar(64) DEFAULT NULL COMMENT '固定电话',  `cust\_mobile` varchar(16) DEFAULT NULL COMMENT '移动电话',  PRIMARY KEY (`cust\_id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=94 DEFAULT CHARSET=utf8; |

第三步：搭建Hibernate的开发环境

1. 创建WEB工程，引入Hibernate开发所需要的jar包

a. MySQL的驱动jar包

b.Hibernate开发需要的jar包（资料/hibernate-release-5.0.7.Final/lib/required/所有jar包）

c. 日志jar包（资料/jar包/log4j/所有jar包）

### 第三步：编写JavaBean实体类

1. Customer类的代码如下：

public class Customer {

private Long cust\_id;

private String cust\_name;

private Long cust\_user\_id;

private Long cust\_create\_id;

private String cust\_source;

private String cust\_industry;

private String cust\_level;

private String cust\_linkman;

private String cust\_phone;

private String cust\_mobile;

// 省略get和set方法

}

### 第四步：创建类与表结构的映射

**1. 在JavaBean所在的包下创建映射的配置文件**

\* 默认的命名规则为：实体类名.hbm.xml

\* 在xml配置文件中引入约束（引入的是hibernate3.0的dtd约束，不要引入4的约束）

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

**2. 如果不能上网，编写配置文件是没有提示的，需要自己来配置**

先复制http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd --> window --> preferences --> 搜索xml --> 选择xml catalog --> 点击add --> 现在URI --> 粘贴复制的地址 --> 选择location，选择本地的DTD的路径

**3. 编写映射的配置文件**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">    <hibernate-mapping>  <class name="com.qf.model.Customer" table="cst\_customer">  <id name="cust\_id" column="cust\_id">  <generator class="native"/>  </id>  <property name="cust\_name" column="cust\_name"/>  <property name="cust\_user\_id" column="cust\_user\_id"/>  <property name="cust\_create\_id" column="cust\_create\_id"/>  <property name="cust\_source" column="cust\_source"/>  <property name="cust\_industry" column="cust\_industry"/>  <property name="cust\_level" column="cust\_level"/>  <property name="cust\_linkman" column="cust\_linkman"/>  <property name="cust\_phone" column="cust\_phone"/>  <property name="cust\_mobile" column="cust\_mobile"/>  </class>  </hibernate-mapping> |

### 第五步：编写Hibernate核心的配置文件

**1. 在src目录下，创建名称为hibernate.cfg.xml的配置文件**

**2. 在XML中引入DTD约束**

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

**3. 打开：资料/hibernate-release-5.0.7.Final/project/etc/hibernate.properties，可以查看具体的配置信息**

a.必须配置的4大参数

#hibernate.connection.driver\_class com.mysql.jdbc.Driver

#hibernate.connection.url jdbc:mysql:///test

#hibernate.connection.username gavin

#hibernate.connection.password

b.数据库的方言（必须配置的）

#hibernate.dialect org.hibernate.dialect.MySQLDialect

c. 可选的配置

#hibernate.show\_sql true

#hibernate.format\_sql true

#hibernate.hbm2ddl.auto update

d.引入映射配置文件（一定要注意，要引入映射文件，框架需要加载映射文件）

\* <mapping resource="com/qf/ model /Customer.hbm.xml"/>

**4. 具体的配置如下**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">    <hibernate-configuration>  <session-factory>  <property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>  <property name="hibernate.connection.url">jdbc:mysql:///hibernate\_day01</property>  <property name="hibernate.connection.username">root</property>  <property name="hibernate.connection.password">root</property>  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>    <mapping resource="com/qf/model/Customer.hbm.xml"/>  </session-factory>  </hibernate-configuration> |

### 第六步：编写Hibernate入门代码

1. 具体的代码如下

|  |
| --- |
| /\*\*  \* 测试保存客户  \*/  @Test  public void testSave(){  // 先加载配置文件  Configuration config = new Configuration();  // 默认加载src目录下的配置文件  config.configure();  // 创建SessionFactory对象  SessionFactory factory = config.buildSessionFactory();  // 创建session对象  Session session = factory.openSession();  // 开启事务  Transaction tr = session.beginTransaction();  // 编写保存代码  Customer c = new Customer();  // c.setCust\_id(cust\_id); 已经自动递增  c.setCust\_name("测试名称");  c.setCust\_mobile("110");  // 保存客户  session.save(c);  // 提交事务  tr.commit();  // 释放资源  session.close();  factory.close();  } |

### 总结快速入门之6步骤

1. 下载Hibernate框架的开发包，创建WEB的项目，导入了开发的jar包

MySQL驱动包、Hibernate开发的必须要有的jar包、日志的jar包

2. 编写数据库和表结构

3. 编写JavaBean，以后不使用基本数据类型，使用包装类

4. 编写映射的配置文件（核心），先导入开发的约束，里面正常配置标签

5. 编写hibernate的核心的配置文件，里面的内容是固定的

6. 编写代码，使用的类和方法

## 技术分析之：Hibernate常用的配置文件

### Hibernate配置文件之映射配置文件

1. 映射文件，即Stu.hbm.xml的配置文件

<hibernate-mapping package="model" >

如果配置 package，name的取值可以是简单类名 name="Person"

\* <class>标签 -- 用来将类与数据库表建立映射关系

\* name -- 类的全路径

\* table -- 表名.(类名与表名一致,那么table属性也可以省略)

\* catalog -- 数据库的名称，基本上都会省略不写

\*dynamic-insert="false" 是否支持动态生成insert语句

\*dynamic-update="false" 是否支持动态生成update语句

如果设置true，hibernate底层将判断提供数据是否为null，如果为null，insert或update语句将没有此项。

\* <id>标签 -- 用来将类中的属性与表中的主键建立映射，id标签就是用来配置主键的。

\* name -- 类中属性名

\* column -- 表中的字段名.(如果类中的属性名与表中的字段名一致,那么column可以省略.)

\* length -- 字段的程度，如果数据库已经创建好了，那么length可以不写。如果没有创建好，生成表结构时，length最好指定。

\* <property> -- 用来将类中的普通属性与表中的字段建立映射.

\* name -- 类中属性名

\* column -- 表中的字段名.(如果类中的属性名与表中的字段名一致,那么column可以省略.)

\* length -- 数据长度

\* type -- 表中列的类型。默认hibernate自己通过getter获得类型，一般情况不用设置

\* Hibernate的数据类型 type="string"

\* Java的数据类型 type="java.lang.String"

\* 数据库字段的数据类型 <column name="name" sql-type="varchar"/>

#### 基本数据与包装类型

* 基本数据类型和包装类型对应hibernate的映射类型相同
* 基本类型无法表达null、数字类型的默认值为0。
* 包装类默认值是null。当对于默认值有业务意义的时候需要使用包装类。

## 类型对应

|  |  |  |
| --- | --- | --- |
| Java数据类型 | Hibernate数据类型 | 标准SQL数据类型 (PS:对于不同的DB可能有所差异) |
| byte、java.lang.Byte | byte | TINYINT |
| short、java.lang.Short | short | SMALLINT |
| int、java.lang.Integer | integer | INGEGER |
| long、java.lang.Long | long | BIGINT |
| float、java.lang.Float | float | FLOAT |
| double、java.lang.Double | double | DOUBLE |
| java.math.BigDecimal | big\_decimal | NUMERIC |
| char、java.lang.Character | character | CHAR(1) |
| boolean、java.lang.Boolean | boolean | BIT |
| java.lang.String | string | VARCHAR |
| boolean、java.lang.Boolean | yes\_no | CHAR(1)('Y'或'N') |
| boolean、java.lang.Boolean | true\_false | CHAR(1)('Y'或'N') |
| java.util.Date、java.sql.Date | date | DATE |
| java.util.Date、java.sql.Time | time | TIME |
| java.util.Date、java.sql.Timestamp | timestamp | TIMESTAMP |
| java.util.Calendar | calendar | TIMESTAMP |
| java.util.Calendar | calendar\_date | DATE |
| byte[] | binary | VARBINARY、BLOB |
| java.lang.String | text | CLOB |
| java.io.Serializable | serializable | VARBINARY、BLOB |
| java.sql.Clob | clob | CLOB |
| java.sql.Blob | blob | BLOB |
| java.lang.Class | class | VARCHAR |
| java.util.Locale | locale | VARCHAR |
| java.util.TimeZone | timezone | VARCHAR |
| java.util.Currency | currency | VARCHAR |

### Hibernate配置文件之核心配置文件

#### 1. 核心配置文件的两种方式

a. 第一种方式是属性文件的形式，即properties的配置文件

hibernate.properties

hibernate.connection.driver\_class=com.mysql.jdbc.Driver

缺点

不能配置加载映射的配置文件，需要手动编写代码去加载

b．第二种方式是XML文件的形式，开发都会选择这种方式(使用)

hibernate.cfg.xml

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

优点

格式比较清晰

编写有提示

可以在该配置文件中加载映射的配置文件（最主要的）

#### 2. 关于hibernate.cfg.xml的配置文件方式

必须有的配置

数据库连接信息:

hibernate.connection.driver\_class -- 连接数据库驱动程序

hibernate.connection.url -- 连接数据库URL

hibernate.connection.username -- 数据库用户名

hibernate.connection.password -- 数据库密码

方言:

hibernate.dialect -- 操作数据库方言

可选的配置

hibernate.show\_sql -- 显示SQL

hibernate.format\_sql -- 格式化SQL

hibernate.hbm2ddl.auto -- 通过映射转成DDL语句

create -- 每次都会创建一个新的表.---测试的时候

create-drop -- 每次都会创建一个新的表,当执行结束之后,将创建的这个表删除.---测试的时候

update -- 如果有表,使用原来的表.没有表,创建一个新的表.同时更新表结构.

validate -- 如果有表,使用原来的表.同时校验映射文件与表中字段是否一致如果不一致就会报错.

加载映射

如果XML方式：<mapping resource="cn/qf/hibernate/model/User.hbm.xml" />

## 技术分析之Hibernate常用的接口和类

### Configuration类和作用

1. Configuration类

Configuration对象用于配置并且启动Hibernate。

Hibernate应用通过该对象来获得对象-关系映射文件中的元数据，以及动态配置Hibernate的属性，然后创建SessionFactory对象。

简单一句话：加载Hibernate的配置文件，可以获取SessionFactory对象。

**2. Configuration类的其他应用（了解）**

加载配置文件的种类，Hibernate支持xml和properties类型的配置文件，在开发中基本都使用XML配置文件的方式。

如果采用的是properties的配置文件，那么通过Configuration configuration = new Configuration();就可以加载配置文件

但是需要自己手动加载映射文件

例如：config.addResource("cn/qf/model/Student.hbm.xml");

如果采用的XML的配置文件，通过Configuration configuration = new Configuration().configure();加载配置文件

### SessionFactory：重要

（二级缓存 被redis缓存数据库给代替了。）

**1. 是工厂类，是生成Session对象的工厂类**

**2. SessionFactory类的特点**

由Configuration通过加载配置文件创建该对象。

SessionFactory对象中保存了当前的数据库配置信息和所有映射关系以及预定义的SQL语句。同时，SessionFactory还负责维护Hibernate的二级缓存。

预定义SQL语句

使用Configuration类创建了SessionFactory对象是，已经在SessionFacotry对象中缓存了一些SQL语句

常见的SQL语句是增删改查（通过主键来查询）

这样做的目的是效率更高

一个SessionFactory实例对应一个数据库，应用从该对象中获得Session实例。

SessionFactory是线程安全的，意味着它的一个实例可以被应用的多个线程共享。

SessionFactory是重量级的，意味着不能随意创建或销毁它的实例。如果只访问一个数据库，只需要创建一个SessionFactory实例，且在应用初始化的时候完成。

SessionFactory需要一个较大的缓存，用来存放预定义的SQL语句及实体的映射信息。另外可以配置一个缓存插件，这个插件被称之为Hibernate的二级缓存，被多线程所共享

**3. 总结**

一般应用使用一个SessionFactory,最好是应用启动时就完成初始化。

**编写HibernateUtil的工具类**

1. 具体代码如下

|  |
| --- |
| public class HibernateUtil {  private static final Configuration cfg;  private static final SessionFactory factory;  static{  // 给常量赋值  // 加载配置文件  cfg = new Configuration().configure();  // 生成factory对象  factory = cfg.buildSessionFactory();  }  // 获取Session对象  public static Session openSession(){  return factory.openSession();  }  } |

### Session接口

**1. 概述**

Session是在Hibernate中使用最频繁的接口。也被称之为持久化管理器。它提供了和持久化有关的操作，比如添加、修改、删除、加载和查询实体对象

Session 是应用程序与数据库之间交互操作的一个单线程对象，是 Hibernate 运作的中心

Session是线程不安全的

所有持久化对象必须在 session 的管理下才可以进行持久化操作

Session 对象有一个一级缓存，显式执行 flush 之前，所有的持久化操作的数据都缓存在 session 对象处

持久化类与 Session 关联起来后就具有了持久化的能力

**2. 特点**

不是线程安全的。应避免多个线程使用同一个Session实例

Session是轻量级的，它的创建和销毁不会消耗太多的资源。应为每次客户请求分配独立的Session实例

Session有一个缓存，被称之为Hibernate的一级缓存。每个Session实例都有自己的缓存

**3. 常用的方法**

save(obj)保存一个对象

delete(obj) 删除一个对象

get(Class,id) 根据id查询

update(obj) 更新对象

saveOrUpdate(obj) -- 保存或者修改（如果没有数据，保存数据。如果有，修改数据）

createQuery() -- HQL语句的查询的方式

### Transaction接口

**1. Transaction是事务的接口**

**2. 常用的方法**

commit() -- 提交事务

rollback() -- 回滚事务

**3. 特点**

Hibernate框架默认情况下事务不自动提交.需要手动提交事务

如果没有开启事务，那么每个Session的操作，都相当于一个独立的事务