hibernate是一个ORM（object/relation mapping）对象/关系映射框架，是jdbc的轻量级封装，一般用于项目持久层，

hibernate的核心配置文件hibernate.cfg.xml

hibernate的基础是java的反射机制

使用hibernate数据库表相对应的java类一般称为pojo（plain ordinary java object）简单的java对象

对象关系映射文件指定domain（域）对象和数据库表的映射关系，文件名一般规范是domain对象名.hbm.xml，位置一般放在对应domain对象同一个文件中

对象关系映射文件配置需要引入指定格式的dtd文件：

<!DOCTYPE hibernate-mapping PUBLIC “-//Hibernate/Hibernate Mapping DTD 3.0//EN” “http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd”>

对象关系映射文件内容一般格式：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="类所在包名">

<class name="数据域对象名（类名）" table="数据库表名">

<id name="类成员属性名" column="对应数据库表列名" type="数据类型（全称）">

<generator class="产生主键值的策略">（默认assigned输入参数指定值）

<param name="产生主键值的策略">对应数据库中的策略名</param>

</generator>

</id>（id标签配置表主键字段，property标签配置普通字段）

[<property name="类成员属性名" type="数据类型（全称）">

<column name="对应数据库表列名" length="指定字段长度" not-null="字段是否允许为空（默认true允许）"></column>

</property>]

[<many-to-one name="类的对象属性名" column="数据库中关联的表外键名"></many-to-one>]（多对一关系关联）

[<set name="类的对象集合属性名" cascade="save-update（cascade表示级联操作：对数据对象进行操作的时候是否将其关联的数据对象也执行相应操作，save-update表示级联更新）">

<key column="数据库中关联表的外键名"></key>

<one-to-many class="类的对象集合属性的类名"/>

</set>]（一对多关系关联）

[<id name="类成员属性名" column="对应数据库表列名" type="数据类型（全称）">

<generator class="foreign">（一对一主键关联关系的外键生成策略）

<param name="property">类的对象属性名</param>

</generator>

</id>]（一对一主键关联关系的外键配置）

[<one-to-one name="类的对象属性名" constrained="关联是否生成外键，默认false"></one-to-one>]（一对一主键关系关联）

[<many-to-one name="类的对象属性名" unique="true"></many-to-one>]（一对一的外键关系关联hibernate会当成多对一标签的特殊情况来处理）

</class>

</hibernate-mapping>

hibernate.cfg.xml文件内容一般格式：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="connection.driver\_class">数据库驱动</property>

<property name="connection.username">用户名</property>

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

<property name="connection.url">连接数据库的url地址</property>

<property name="dialect">指定连接的数据库名称（相关信息在hibernate文件中的project\etc \hibernate.properties配置文件中）</property>

<property name="show\_sql">true</property>（对数据库操作时控制台输出显示hibernate对应的sql语句）

<property name="current\_session\_context\_class">thread/jta</property>（配置当前session与thread/jta绑定，thread表示本地事务，jta表示全局事务（即多个数据库的事务），可以使用getCurrentSession()方法获得session）

<property name="format\_sql">true</property>（对数据库操作时控制台输出显示hibernate对应的格式化后的sql语句）

<property name="hbm2ddl.auto">create/update/create-drop/validate</property>

（对象映射文件和相应的类指定完成时，hibernate会根据这些文件自动创建对应的数据库，create表示没有对应的表则创建对应的表，有则覆盖创建，update表示没有对应的表则创建对应的表，有表且表结构发生变化则改变表结构，没有变化则不改变表结构，create-drop表示创建新表，当sessionFactory设置关闭后将删除schema，validate表示每添加数据前会验证数据库表的结构与配置文件中的结构是否一致，一般建议使用update）

<mapping resource="对象映射文件的路径"/>（指定管理的对象映射文件）

</session-factory>

</hibernate-configuration>

Configuration 配置名=new Configuration().configure(“配置文件名（默认hibernate.cfg.xml）”); 读取配置文件，并完成初始化

SessionFactory 会话工厂名=configuration.buildSessionFactory(); 创建会话工厂对象，一般一个会话工厂对象对应一个数据库，通常设置会话工厂为单态的，

Session 会话名=会话工厂名.openSession(); 创建会话对象，相当于Connection对象

会话名.save(对象名); 将对象信息通过会话保存到对象对应的数据库表中即对应的数据库表插入一条相应对象信息的记录

会话名.close(); 关闭会话

hibernate强制统一要求对数据库的除查询以外的操作为事务，需要进行事务提交

Transaction 事务名=会话名.beginTransaction(); 创建会话对应的事务

事务名.commit(); 提交事务

hibernate数据库表对应的类对象一般规范需要序列化，实现Serializable接口即class 类名 implements java.io.Serializable

类名 记录对象名=(类名)会话.load(类名.class, 主键值); 通过主键值获取对应数据库表中记录的对象信息

类名 记录对象名=(类名)会话.get(类名.class, 主键值); 通过主键值获取对应数据库表中记录的对象信息

类名.class获取类的class对象

记录对象名.设置函数("值"); 通过load加载后的记录对象可以通过设置对象的属性值对数据库表中对应的记录进行修改，hibernate默认修改是对对象所有成员属性进行修改，未通过设置函数修改的属性默认修改为原属性值

会话名.delete(记录对象名); 删除记录对象对应的数据库表中的记录

hql语句（hibernate query language）



ORM框架除了hibernate外还有Apache OJB，Cayenne，Jaxor，iBatis，jRelationalFramework ，mirage，SMYLE，TopLink，ejb cmp

hibernate的核心配置文件hibernate.cfg.xml文件可以用hibernate.properties配置文件来代替

hibernate框架的configuration类负责管理hibernate的配置信息，读取加载hibernate.cfg.xml配置文件中的信息已经配置的驱动，管理\*.hbm.xml对象关系文件

SessionFactory会话工厂可以缓存sql语句和数据（称为session级缓存）

通过SessionFactory获取Session的两个方法的区别：

openSession()是获取一个新的session，需要进行session资源关闭操作

getCurrentSession()是获取和当前进程绑定的session，即在同一个进程中的session是同一个，这样有利于事务控制，并且在事务提交或者回滚后系统会自动关闭session资源，但是对数据库的任何操作都要封装成事务包括查询操作

Session默认线程不同步

Session的两个对数据库查询操作的方法get()和load()的区别

get()查询数据会先去缓存（session缓存/二级缓存）查找，未查到就立即向数据库发出对应的sql执行语句，查不到数据会返回null

load()查询数据会先去缓存（session缓存/二级缓存）查找，未查到会返回一个代理对象，当使用到查询结果内容的时候才会向数据库发出对应的sql执行语句（这种现象称为延时加载，可以通过修改配置文件中<class name="数据域对象名（类名）" lazy="true/false" table="数据库表名">的lazy属性值（默认true）修改加载方式），查询不到数据会报错ObjectNotFoundException

建议确定数据库中有要查找的数据时用load()方法，不确定时用get()方法

ThreadLocal<类型> 名称 定义线程局部模式，将指定类型的对象与线程绑定即在线程作用域中指定数据类型的对象时同一个

Session 会话名=threadLocal.get(); 从线程中获得绑定的会话对象

threadLocal.set(会话名); 将会话绑定到线程中

HQL（Hibernate Query Language）是面向对象的查询语言，与SQL不同，对象名java类和属性区分大小写其他不区分大小写，查询针对数据库表映射的数据域类不是数据库表，支持多态，语句主要通过Query来实现

List<domain类名> 队列名=会话名.createQuery("（hql查询语句）from（数据域类名）Employee where （条件）id=3").list(); 查询结果会自动封装成对应的domain对象存入队列中，Query是对数据对象进行操作不是对数据库表，查询的数据域类名是数据库表对应的类名，条件中可以是数据库表对应的字段名也可以是类名对应的属性名

Criteria是比HQL更面向对象的查询方式

List<domain类名> 队列名=会话名.createCriteria(查询条件可以是类名.class查询整个类对应的数据库表所有内容).setMaxResults(显示最大结果个数).add(过滤条件).list(); 通过设置相应信息查询相应结果自动封装成对应的数据对象存入队列中

sql语句一般建议查询语句只查需要的字段，hql语句一般建议查询所有，因为hql语句的优势是面向对象的，查询结果会被封装成对象，方便关联查询等操作

from 类名 [where 条件] hql语句查询所有属性

select 属性名[,属性名] from 类名 [where 条件] hql语句查询部分属性

类名 对象名=会话名.createQuery(“hql语句”).uniqueResult; 确定要查询的数据最多只有一条记录存在，指定查询语句查询出一条记录即可停止查询，减少查询时间提高效率，存在多条记录会报错

hql语句类属性名前加distinct表示对该属性值进行过滤去重

hql语句条件（属性名 between 值 and 值）表示查询指定两个值之间的的属性名的属性值

hql语句条件（属性名 in/not in）表示查询在/不在指定值内的属性名的属性值

hql语句条件（group by 属性名）表示查询结果按照属性名进行分组

属性名 as 别名 hql语句指定属性名的别名

hql语句的having条件和聚合函数与数据库中用法相同，但是对类和属性名操作的

hql语句查询一列数据时返回的是对象不是对象数组

hql语句类名.属性名表示对应数据库表.字段名

List<domain类名> 队列名=会话名.createQuery("hql语句").setFirstResult(开始记录数（第一条记录数值为0）).setMaxResult(显示记录个数); 查询结果分页显示从指定开始记录数开始显示指定个数条记录

Query 查询名=会话名.createQuery("hql语句（输入参数用?/:参数名表示）"); 通过输入参数的方式防止sql注入漏洞

查询名.setString(参数序号/”参数名”,参数值); 通过输入参数的方式防止sql注入漏洞

hql查询语句也可以配置到对象关系映射文件中：

<query name="查询名"><![CDATA[hql语句]]></query>  
List<类名> 队列名=会话名.getNameQuery("查询名").list(); 程序中调用对象关系映射文件中的hql查询语句

List<类名> list=会话名.createCriteria(类名.class).add(Restrictions.gt("属性名",指定数值)); 查询属性值对应的字段大于指定数值的数据记录

hibernate数据对象的三种状态：

瞬时态（transient）：数据库中没有与之相对应的数据存在，超出作用域会被java虚拟机回收器回收，一般是数据对象被创建但没有与session关联的状态

持久态（persistent）：数据库中有与之相对应的数据存在，数据对象已经与session关联且session未关闭的状态，hibernate会自动检测持久态下的数据对象中的数据发生的变化相应的在提交事务时会改变数据库中对应的数据发生对应的变化

游离态（detached）：据库中有与之相对应的数据存在，数据对象没有与session关联或者session已经关闭的状态，游离态下的数据对象中的数据发生变化不会影响数据库中与之对应的数据发生相应变化

由于延时加载原因，关联查询关联数据时，hibernate会向数据库发送两次查询，第一次返回结果是没有实际的关联数据的代理对象，当使用到关联数据时才会发出第二次查询关联数据，当使用关联数据在session（会话）结束时会报错误，原因是hibernate发出第二次查询时session已经关闭无法查询

解决延时加载的一种方法：通过修改配置文件中<class name="数据域对象名（类名）" lazy="true/false" table="数据库表名">的lazy属性值（默认true）修改加载方式

解决hibernate延时加载的方法：

在对象映射文件中配置关闭延时加载

Hibernate.initialize(查询的对象名.get关联的类名());初始化代理对象

在session未关闭是，使用关联对象的数据，强制数据库进行访问查询操作

Web项目既解决延时加载又解决hibernate发送多余SQL语句的方法：openSessionInView在web接受请求的位置设置一个过滤器，该过滤器创建session并且在过滤器结束的时候再关闭session，请求响应过程中使用同一个session（过滤器创建的session）并且不进行关闭。

基础接口的参数类型无法确定的情况可以用Serializable接口这个类型来代替大部分常用参数类型，因为大部分常用参数类型都继承了Serializable接口

Web应用一般用<%=this.getServletContext().getContextPath() %>或者${pageContext.request.contextPath}代替web应用名称的路径，提高代码的灵活性

Hibernate实现延时加载需要重写数据域（domain）对象，所以实现延时加载数据域（domain）对象必须是非final对象

Hibernate会向一级缓存中放入数据的数据库操作：save，update，saveOrUpdate，load，get，list，iterate，lock

Hibernate会首先从一级缓存中取出数据的数据库操作： load，get

Session.evict(数据对象名); 清楚hibernate指定数据对象名的一级缓存（Session级缓存）

Session.clear(数据对象名); 清楚hibernate所有一级缓存（session级缓存）

Hibernate一级缓存不需要配置但不能控制缓存数量，缓存大批量数据时会有内存溢出的可能，一级缓存的生命周期与session关联，session关闭时一级缓存会自动清空

Hibernate由于一级缓存缓存的数量有限，所以有二级缓存（SessionFactory缓存），hibernate二级缓存是交给第三方处理的，所以二级缓存需要在hibernate的配置文件hibernate.cfg.xml中配置并且引入第三方的包和相应的缓存配置文件才能使用，常用的二级缓存有Hashtable，EHCache，OSCache

Hibernate二级缓存配置：

<session-factory>

<property name="cache.use\_second\_level\_cache">true</property>（指定使用二级缓存）

<property name="cache.provider\_class">二级缓存的名称（包含包名）</property>（指定使用的二级缓存的名称）

<property name="hibernate.generate\_statistics">true</property>（指定是否使用hibernate二级缓存统计信息）

<class-cache usage="read-only/read-write/nonstrict-read-write/transactional" class="数据域类名（包含包名）"/>（指定使用二级缓存的策略（只读，读写，不严格读写，事务）和使用二级缓存的数据域对象，这个配置可以在对应的对象映射文件中配置，标签名为cache）

</session-factory>

Statistics 名称=SessionFactory.getStatistics(); 获得hibernate二级缓存所有统计信息

主键值产生策略：

increment：自动递增，增量固定为1不能修改，适用于所有数据库系统，不适用于多进程，主键必须为数值类型

identity：适用于支持identity数据库系统，主键必须为数值类型

sequence：适用于支持sequence数据库系统，主键必须为数值类型

native：会根据数据库系统类型自动从identity、sequence和hilo中选择适合的策略，主键必须为数值类型

hilo：由hibernate按照high/low算法产生值，从指定的数据库表和字段中获取值（需要配置相应的表和字段），适用于所有数据库系统，主键必须为数值类型

uuid：由hibernate基于128位bit唯一值产生算法，根据当前设备ip、时间、jvm启动时间、内部自增量4个参数生产16进制数值，一般主键使用字符串String类型

assigned：自定义设计指定主键值

映射复合主键：

<composite-id name="复合主键名" class="复合主键生成类名（包含包名）">（指定复合主键的名称以及复合后的类名）

[<key-property name="数据域属性名" type="数据域属性类型">

<column name="数据域属性对应的数据库字段名"></column>

</key-property>]（指定需要复合的主键）

</composite-id>

foreign：主键值与另一张表的主键相关联

主键类型是数值类型的oracle数据库一般生成策略是sequence、mysql数据库一般是increment/assigned、sql server数据库一般是identity/native/assigned

主键类型使字符串String类型一般生成策略是uuid/assigned

Hibernate不适合OLAP（On-Line Analytical Processing联机分析处理）以查询分析数据为主的系统，适合OLTP（On-Line Transaction Processing联机事务处理），hibernate批量操作数据的效率不高，适合数据量不大，性能要求不是很高，事务操作比较多的系统