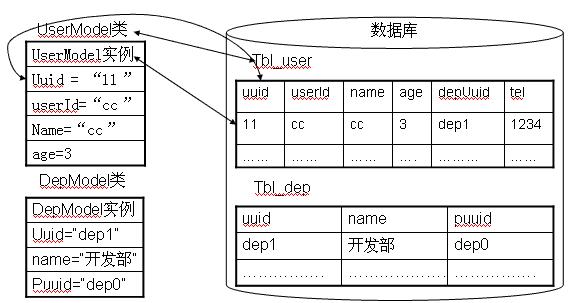
## **一、什么是Hibernate？**

　　Hibernate是一个轻量级的ORMapping框架

　　ORMapping原理（Object Relational Mapping）



ORMapping基本对应规则：

1：类跟表相对应

2：类的属性跟表的字段相对应

3：类的实例与表中具体的一条记录相对应

4：一个类可以对应多个表，一个表也可以对应对个类

5：DB中的表可以没有主键，但是Object中必须设置主键字段

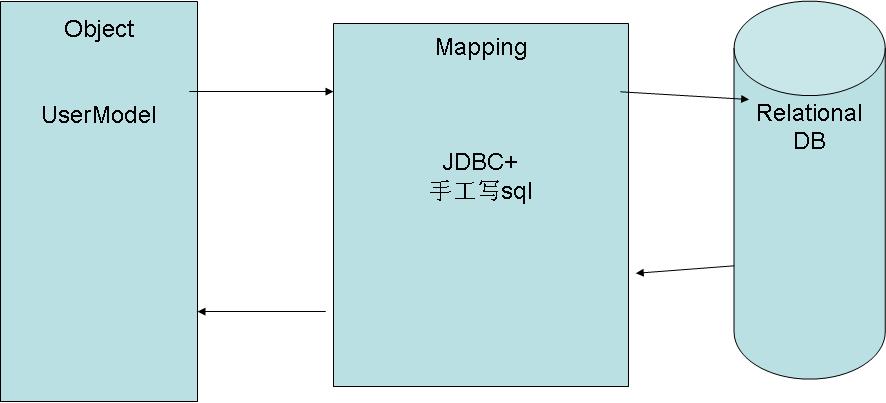
6：DB中表与表之间的关系（如：外键）映射成为Object之间的关系

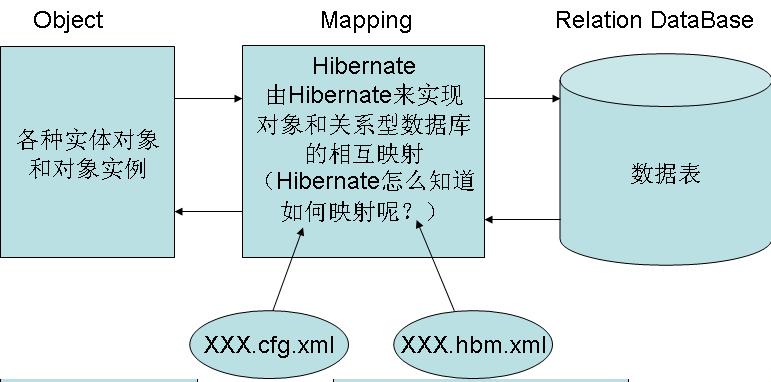
7：Object中属性的个数和名称可以和表中定义的字段个数和名称不一样

ORMapping的基本实现方式：

使用JDBC，用SQL来操作数据库，只是看动态生成还是人工写代码来实现。

大家想想，我们实现过ORMapping吗？



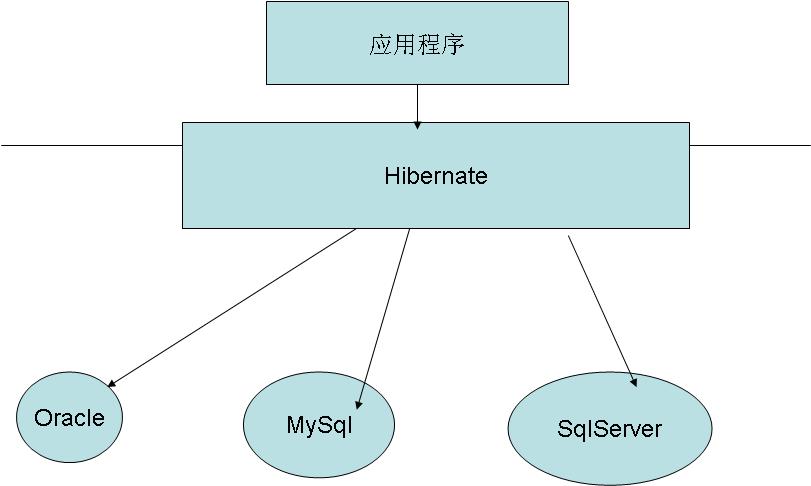


## **二、Hibernate能干什么：**

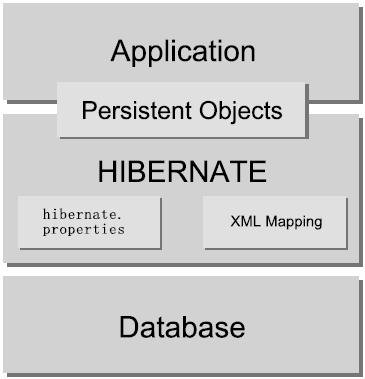
    Hibernate主要用来实现Java对象和表之间的映射，除此之外还提供还提供数据查询和获取数据的方法，可以大幅度减少开发时人工使用SQL和JDBC处理数据的时间。

Hibernate的目标是对于开发者通常的数据持久化相关的编程任务，解放其中的95%。对于以数据为中心的程序来说,它们往往只在数据库中使用存储过程来实现商业逻辑,Hibernate可能不是最好的解决方案;对于那些在基于Java的中间层应用中，它们实现面向对象的业务模型和商业逻辑的应用，Hibernate是最有用的。

    Hibernate可以帮助你消除或者包装那些针对特定厂商的SQL代码，并且帮你把结果集从表格式的表示形式转换到一系列的对象去

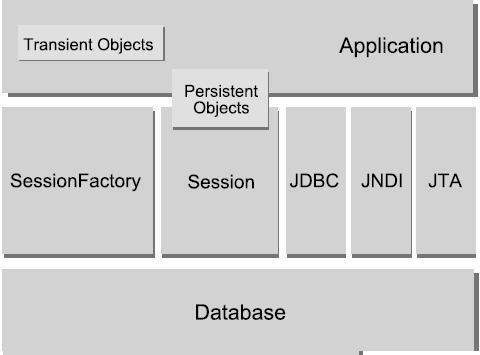


一个非常简要的Hibernate体系结构的高层概要图

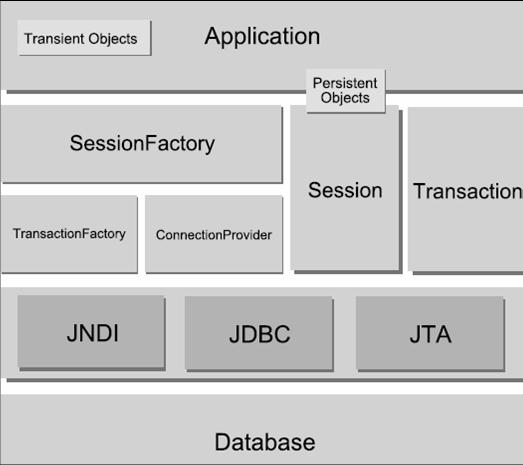


Hibernate运行时体系结构

“最小”的体系结构方案，要求应用程序提供自己的 JDBC 连接并管理自己的事务。这种方案使用了Hibernate API 的最小子集.



“全面解决”的体系结构方案，将应用层从底层的JDBC/JTA API中抽象出来，而让Hibernate来处理这些细节。



## **（三）Hibernate中的对象**

SessionFactory (org.hibernate.SessionFactory)

针对单个数据库映射关系经过编译后的内存镜像，是线程安全的（不可变）。 它是生成的工厂，本身要用到。

Session (org.hibernate.Session)

表示应用程序与持久储存层之间交互操作的一个单线程对象，此对象生存期很短，隐藏了连接，也是的工厂。

Transaction (org.hibernate.Transaction)

应用程序用来指定原子操作单元范围的对象，它是单线程的，生命周期很短。它通过抽象将应用从底层具体的、以及事务隔离开。

ConnectionProvider (org.hibernate.connection.ConnectionProvider)

生成连接的工厂（有连接池的作用）。它通过抽象将应用从底层的或隔离开。仅供开发者扩展/实现用，并不暴露给应用程序使用。

TransactionFactory (org.hibernate.TransactionFactory)

生成对象实例的工厂。仅供开发者扩展/实现用，并不暴露给应用程序使用。

示例如下：

数据库表：Student.sql

[IMG_263](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

CREATE TABLE "STUDENT"

(

"STUNO" VARCHAR2(20),

"STUNAME" VARCHAR2(20),

"STUPASS" VARCHAR2(20),

"STUSEX" VARCHAR2(2),

"MOBILE" VARCHAR2(20),

"EMAIL" VARCHAR2(20),

"ADDRESS" VARCHAR2(50),

"STUAGE" NUMBER

)

[IMG_264](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

配置文件：hibernate.cfg.xml

[IMG_265](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

<?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>

<!-- 数据库URL -->

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:orcl</property>

<!-- 数据库用户名 -->

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

<!-- 数据库密码 -->

<property name="connection.password">123123</property>

<!-- JDBC驱动 -->

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<!-- 每个数据库都有对应的Dialect以匹配其平台特性 -->

<property name="dialect">org.hibernate.dialect.Oracle10gDialect</property>

<!-- 指定当前session范围和上下文 -->

<property name="current\_session\_context\_class">thread</property>

<!-- 指定运行期生成的SQL输出到日志以供调试 -->

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

<!-- 是否格式化sql -->

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

<!-- 映射文件 -->

<mapping resource="Student.hbm.xml" />

</session-factory></hibernate-configuration>

[IMG_266](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

1：与被描述的类同名 ，如：Student.hbm.xml

2：存放位置与所描述类存放在同一文件夹下

3：主要有如下四部分配置 ：

（1）类和表的映射

（2）主键的映射

（3）类的属性和DB中字段的映射

（4）关系的映射

4：配置的时候可以到hibernate发行包里面找个例子，比如可以用“\project\hibernate-core\src\test\java\org\hibernate\test\cid”下面的Customer.hbm.xml作例子

映射文件：Student.hbm.xml

[IMG_267](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

<?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>

<class name="hibDemo1.entity.Student" table="STUDENT">

<id name="stuNo" type="java.lang.String" column="STUNO">

<!--generator的class类型

assigned:主键的状态 assigned表示程序生成

sequence：Oracle中的序列

identity：Sql中的自动编号

increment：先查询最大的编号再增1

uuid：生成32位长的字符串

native：根据数据库自动生成

-->

<generator class="assigned" />

</id>

<!-- 直接使用property属性设置 -->

<property name="stuName" type="java.lang.String" column="STUNAME" length="50" not-null="true" />

<!-- 使用column设置 -->

<property name="stuPass" type="java.lang.String" column="STUPASS">

<column name="STUPASS" length="50" not-null="true"></column>

</property>

<property name="stuSex" type="java.lang.String" column="STUSEX" />

<property name="stuAge" type="java.lang.Integer" column="STUAGE" />

<property name="Mobile" type="java.lang.String" column="MOBILE" />

<property name="Email" type="java.lang.String" column="EMAIL" />

<property name="Address" type="java.lang.String" column="ADDRESS" />

</class></hibernate-mapping>

[IMG_268](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

客户端文件：

java代码：StudentService.java

[IMG_269](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

package hibDemo1.dao;

import hibDemo1.entity.Student;

import java.util.List;

import org.hibernate.HibernateException;import org.hibernate.Query;import org.hibernate.Session;import org.hibernate.SessionFactory;import org.hibernate.Transaction;import org.hibernate.cfg.Configuration;

public class StudentService {

private static Configuration conf;

private static SessionFactory sf;

private static Transaction tx;

static {

try {

conf = new Configuration().configure();

sf = conf.buildSessionFactory();

} catch (HibernateException e) {

e.printStackTrace();

}

}

public static Session getSession() {

return sf.openSession(); //过去老的方法，不需要使用事务

//return sf.getCurrentSession(); // 新的方法，需要和事务一起使用，可以保证每个用户创建的session独立，需要在配置文件中配置

//<property name="current\_session\_context\_class">thread</property> }

/\*\*

\* 获取所有学生列表

\*

\* @return

\*/

public List<Student> GetAllStudent() {

List<Student> list = null;

Session session = getSession();

if (session != null) {

try {

String hql = "from Student";

Query query = session.createQuery(hql);

list = query.list();

} catch (HibernateException e) {

e.printStackTrace();

} finally {

session.close();

}

}

return list;

}

/\*\*

\* 获取单个学生信息

\*

\* @param stuNo

\* @return

\*/

public Student GetStudentBystuNo(String stuNo) {

Student stu = null;

Session session = getSession();

if (session != null) {

try {

// get如果没有查询到数据，则返回null

// stu = (Student) session.get(Student.class, stuNo);

// load如果没有查询到数据，则抛出异常

stu = (Student) session.load(Student.class, stuNo);

} catch (HibernateException e) {

e.printStackTrace();

} finally {

session.close();

}

}

return stu;

}

/\*\*

\* 添加一个学生

\*

\* @param stu

\* @author Administrator

\*/

public boolean AddStudent(Student stu) {

boolean b = false;

Session session = getSession();

if (session != null) {

try {

// 开启一个事务

tx = session.beginTransaction();

// 保存 session.save(stu);

// 提交事务 tx.commit();

return true;

} catch (HibernateException e) {

e.printStackTrace();

tx.rollback();

} finally {

session.close();

}

}

return b;

}

/\*\*

\* 更新一个学生

\*

\* @param stu

\* @author Administrator

\*/

public boolean UpdateStudent(String stuNo, String newName) {

boolean b = false;

Session session = getSession();

if (session != null) {

try {

// 开启一个事务

tx = session.beginTransaction();

// 获取一个学生对象

Student stu = (Student) session.load(Student.class, stuNo);

// 更新某个属性 stu.setStuName(newName);

// 提交事务 tx.commit();

return true;

} catch (HibernateException e) {

e.printStackTrace();

tx.rollback();

} finally {

session.close();

}

}

return b;

}

/\*\*

\* 更新一个学生

\*

\* @param stu

\* @author Administrator

\*/

public boolean DeleteStudent(String stuNo) {

boolean b = false;

Session session = getSession();

if (session != null) {

try {

// 开启一个事务

tx = session.beginTransaction();

// 获取一个学生对象

Student stu = (Student) session.load(Student.class, stuNo);

// 删除操作 session.delete(stu);

// 提交事务 tx.commit();

return true;

} catch (HibernateException e) {

e.printStackTrace();

tx.rollback();

} finally {

session.close();

}

}

return b;

}

}

[IMG_270](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

test.java

[IMG_271](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

package hibDemo.test;

import hibDemo1.dao.StudentService;import hibDemo1.entity.Student;

import java.util.List;import java.util.Scanner;

public class test {

public static void main(String[] args) {

// 添加学生信息

// AddStudent();

// 显示所有学生信息 ShowAll();

// 显示单个学生信息

// ShowOne();

//更新学生信息

//Update();

//删除学生信息

//Delete();

ShowAll();

}

public static void ShowAll() {

StudentService service = new StudentService();

List<Student> list = service.GetAllStudent();

for (Student student : list) {

System.out.println(student.getStuNo() + " " + student.getStuName());

}

}

public static void ShowOne() {

String no = "A004";

StudentService service = new StudentService();

Student student = service.GetStudentBystuNo(no);

if (student != null) {

System.out.println(student.getStuNo() + " " + student.getStuName());

} else {

System.out.println("no data");

}

}

public static void AddStudent() {

Scanner input = new Scanner(System.in);

Student stu = new Student();

System.out.print("请输入学生编号：(A001)");

stu.setStuNo(input.next());

System.out.print("请输入学生姓名：(A001)");

stu.setStuName(input.next());

stu.setStuPass("888888");

System.out.print("请输入学生年龄：(0-100)");

stu.setStuAge(input.nextInt());

System.out.print("请输入学生手机号：(A001)");

stu.setMobile(input.next());

System.out.print("请输入学生邮箱：(A001)");

stu.setEmail(input.next());

System.out.print("请输入学生地址：(A001)");

stu.setAddress(input.next());

StudentService service = new StudentService();

service.AddStudent(stu);

}

public static void Update() {

Scanner input = new Scanner(System.in);

System.out.print("输入要修改的学号：");

String stuNo = input.next();

System.out.print("输入要修改的姓名：");

String newName = input.next();

StudentService service = new StudentService();

service.UpdateStudent(stuNo, newName);

}

public static void Delete() {

Scanner input = new Scanner(System.in);

System.out.print("输入要修改的学号：");

String stuNo = input.next();

StudentService service = new StudentService();

service.DeleteStudent(stuNo);

}

}

[IMG_272](http://www.cnblogs.com/zhengcheng/p/javascript:void(0);" \o "复制代码)

**说明：**

**1：**SessionFactory sf = new Configuration().configure().buildSessionFactory();这句话的意思是读取hibernate.cfg.xml，创建Session工厂，是线程安全的。

默认是”hibernate.cfg.xml”，不用写出来，如果文件名不是”hibernate.cfg.xml”，那么需要显示指定，如下：

SessionFactory sf = new Configuration(). **configure(** **“javass.cfg.xml”** **).**buildSessionFactory();

**2：**Session是应用程序主要使用的Hibernate接口，约相当于JDBC的Connection+Statement/PreparedStatement的功能，是线程不安全的

3：在Hibernate4里面，已经不推荐使用Configuration类了，而改为使用 ServiceRegistryBuilder和MetadataSources来代替，新的写法大致如下：

ServiceRegistryBuilder builder = new ServiceRegistryBuilder().configure();

builder.applySetting("connection.driver\_class", "oracle.jdbc.driver.OracleDriver");

builder.applySetting("connection.url", "jdbc:oracle:thin:@localhost:1521:orcl");

builder.applySetting("connection.username", "ztb");

builder.applySetting("connection.password", "ztb");

builder.applySetting("connection.pool\_size", "2");

builder.applySetting("hibernate.dialect", "org.hibernate.dialect.OracleDialect");

builder.applySetting("show\_sql", "true");

MetadataSources sources = new MetadataSources( builder.buildServiceRegistry() );

sources.addResource("cn/javass/h4/hello/UserModel.hbm.xml");

MetadataImpl metadata = (MetadataImpl) sources.buildMetadata();

SessionFactory sf = metadata.getSessionFactoryBuilder().buildSessionFactory();

这种写法，现在还没有实现完全，不太好用，所以官方给出的示例里面还是采用以前的方式，大家先了解一下。

4：这里使用的事务Transaction是Hibernate的Transaction，需要有，不能去掉。

为什么必须有这个Hibernate的事务呢？以HelloWorld为例来看看：

