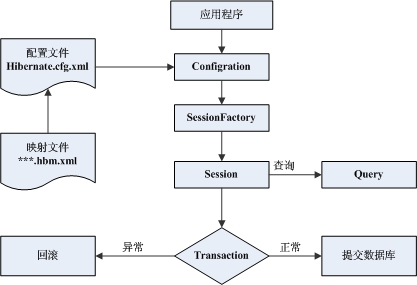
**Hibernate的核心组件**  
在基于MVC设计模式的JAVA WEB应用中，Hibernate可以作为模型层/数据访问层。它通过配置文件(hibernate.properties或hibernate.cfg.xml)和映射文件(\*\*\*.hbm.xml)把JAVA对象或PO(Persistent Object,持久化对象)映射到数据库中的数据库，然后通过操作PO，对数据表中的数据进行增，删，改，查等操作。  
除配置文件，映射文件和持久化类外，Hibernate的核心组件包括以下几部分：  
a)Configuration类：用来读取Hibernate配置文件，并生成SessionFactory对象。  
b)SessionFactory接口：产生Session实例工厂。  
c)Session接口：用来操作PO。它有get(),load(),save(),update()和delete()等方法用来对PO进行加载，保存，更新及删除等操作。它是Hibernate的核心接口。  
d)Query接口：用来对PO进行查询操。它可以从Session的createQuery()方法生成。  
e)Transaction接口：用来管理Hibernate事务，它主要方法有commit()和rollback()，可以从Session的beginTrancation()方法生成。

**Persistent Object**  
持久化对象可以是普通的Javabeans,惟一特殊的是它们与（仅一个）Session相关联。JavaBeans在Hibernate中存在三种状态：  
1.临时状态(transient):当一个JavaBean对象在内存中孤立存在，不与数据库中的数据有任何关联关系时，那么这个JavaBeans对象就称为临时对象(Transient Object)。  
2.持久化状态(persistent):当一个JavaBean对象与一个Session相关联时，就变成持久化对象(Persistent Object)  
3.脱管状态(detached):在这个Session被关闭的同时，这个对象也会脱离持久化状态，就变成脱管状态(Detached Object)，可以被应用程序的任何层自由使用，例如可以做与表示层打交道的数据舆对象(Data Transfer Object)。

**Hibernate的运行过程**Hibernate的运行过程如下：  
A:应用程序先调用Configration类，该类读取Hibernate的配置文件及映射文件中的信息，并用这些信息生成一个SessionFactpry对象。  
B:然后从SessionFactory对象生成一个Session对象，并用Session对象生成Transaction对象;可通过Session对象的get(),load(),save(),update(),delete()和saveOrUpdate()等方法对PO进行加载，保存，更新，删除等操作;在查询的情况下，可通过Session对象生成一个Query对象，然后利用Query对象执行查询操作;如果没有异常，Transaction对象将 提交这些操作结果到数据库中。  
  
Hibernate的运行过程如下图：  


Hibernate简单示例：  
数据：

IMG_257create table T\_register  
IMG_258(  
IMG_259    id    **int** primary key,  
IMG_260    userName **varchar**(**30**),  
IMG_261    userPwd **varchar**(**30**),  
IMG_262    sex **varchar**(**10**),  
IMG_263    age **int**  
IMG_264)

视图层：注册页面register.jsp

IMG_265<%@ page language="java" contentType="text/html; charset=GBK"  
IMG_266    pageEncoding="GBK"%>  
IMG_267<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
IMG_268<html>  
IMG_269<head>  
IMG_270<meta http-equiv="Content-Type" content="text/html; charset=GBK">  
IMG_271<title>Insert title here</title>  
IMG_272</head>  
IMG_273<body>  
IMG_274  
IMG_275<form action="servlet/RegisterServlet" method="POST">  
IMG_276    用户名：<input type="text" name="userName" /><BR>  
IMG_277    密  码：<input type="text" name="userPwd" /><BR>  
IMG_278    性  别：<input type="text" name="sex" /><BR>  
IMG_279    年  龄：<input type="text" name="age" /><BR>  
IMG_280    <input type="submit" value="save" />  
IMG_281</form>  
IMG_282</body>  
IMG_283</html>

设计持久化类TRegister.java

IMG_284  
IMG_285package hibernate.PO;  
IMG_286/\*\*  
IMG_287 \* 持久化类  
IMG_288 \*/  
IMG_289  
IMG_290public class TRegister  implements java.io.Serializable {  
IMG_291  
IMG_292  
IMG_293    // Fields      
IMG_294  
IMG_295     private Integer id;  
IMG_296     private String userName;  
IMG_297     private String userPwd;  
IMG_298     private String sex;  
IMG_299     private Integer age;  
IMG_300  
IMG_301  
IMG_302    // Constructors  
IMG_303  
IMG_304    /\*\* default constructor \*/  
IMG_305    public TRegister() {  
IMG_306    }  
IMG_307  
IMG_308    /\*\* minimal constructor \*/  
IMG_309    public TRegister(Integer id) {  
IMG_310        this.id = id;  
IMG_311    }  
IMG_312      
IMG_313    /\*\* full constructor \*/  
IMG_314    public TRegister(Integer id, String userName, String userPwd, String sex, Integer age) {  
IMG_315        this.id = id;  
IMG_316        this.userName = userName;  
IMG_317        this.userPwd = userPwd;  
IMG_318        this.sex = sex;  
IMG_319        this.age = age;  
IMG_320    }  
IMG_321  
IMG_322     
IMG_323    // Property accessors  
IMG_324  
IMG_325    public Integer getId() {  
IMG_326        return this.id;  
IMG_327    }  
IMG_328      
IMG_329    public void setId(Integer id) {  
IMG_330        this.id = id;  
IMG_331    }  
IMG_332  
IMG_333    public String getUserName() {  
IMG_334        return this.userName;  
IMG_335    }  
IMG_336      
IMG_337    public void setUserName(String userName) {  
IMG_338        this.userName = userName;  
IMG_339    }  
IMG_340  
IMG_341    public String getUserPwd() {  
IMG_342        return this.userPwd;  
IMG_343    }  
IMG_344      
IMG_345    public void setUserPwd(String userPwd) {  
IMG_346        this.userPwd = userPwd;  
IMG_347    }  
IMG_348  
IMG_349    public String getSex() {  
IMG_350        return this.sex;  
IMG_351    }  
IMG_352      
IMG_353    public void setSex(String sex) {  
IMG_354        this.sex = sex;  
IMG_355    }  
IMG_356  
IMG_357    public Integer getAge() {  
IMG_358        return this.age;  
IMG_359    }  
IMG_360      
IMG_361    public void setAge(Integer age) {  
IMG_362        this.age = age;  
IMG_363    }  
IMG_364}

设计Hibernate配置文件hibernate.cfg.xml

IMG_365  
IMG_366<?xml version='1.0' encoding='UTF-8'?>  
IMG_367<!DOCTYPE hibernate-configuration PUBLIC  
IMG_368          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
IMG_369          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
IMG_370  
IMG_371<!-- Generated by MyEclipse Hibernate Tools.                   -->  
IMG_372<hibernate-configuration>  
IMG_373  
IMG_374<session-factory>  
IMG_375    <property name="connection.username">root</property>  
IMG_376    <property name="connection.url">  
IMG_377        jdbc:mysql://localhost:3306/study  
IMG_378    </property>  
IMG_379    <property name="dialect">  
IMG_380        org.hibernate.dialect.MySQLDialect  
IMG_381    </property>  
IMG_382    <property name="myeclipse.connection.profile">MySQL5.0</property>  
IMG_383    <property name="connection.password">root</property>  
IMG_384    <property name="connection.driver\_class">  
IMG_385        org.gjt.mm.mysql.Driver  
IMG_386    </property>  
IMG_387    <property name="show\_sql">true</property>  
IMG_388    <mapping resource="hibernate/PO/TRegister.hbm.xml" />  
IMG_389  
IMG_390</session-factory>  
IMG_391  
IMG_392</hibernate-configuration>

设计映射文件TRegister.hbm.xml

IMG_393  
IMG_394<?xml version="1.0" encoding="utf-8"?>  
IMG_395<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
IMG_396"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
IMG_397<!--   
IMG_398    Mapping file autogenerated by MyEclipse - Hibernate Tools  
IMG_399-->  
IMG_400<hibernate-mapping>  
IMG_401    <class name="hibernate.PO.TRegister" table="t\_register">  
IMG_402        <id name="id" type="java.lang.Integer">  
IMG_403            <column name="id" />  
IMG_404            <!-- 我在MYSQL中并没有设置该字段递增，但在Hibernate中依然可以设置为递增 -->  
IMG_405            <generator class="increment" />  
IMG_406        </id>  
IMG_407        <property name="userName" type="java.lang.String">  
IMG_408            <column name="userName" length="30" />  
IMG_409        </property>  
IMG_410        <property name="userPwd" type="java.lang.String">  
IMG_411            <column name="userPwd" length="30" />  
IMG_412        </property>  
IMG_413        <property name="sex" type="java.lang.String">  
IMG_414            <column name="sex" length="10" />  
IMG_415        </property>  
IMG_416        <property name="age" type="java.lang.Integer">  
IMG_417            <column name="age" />  
IMG_418        </property>  
IMG_419    </class>  
IMG_420</hibernate-mapping>  
IMG_421

设计hibernate基础类HibernateUtil.java

IMG_422package hibernate;  
IMG_423/\*\*  
IMG_424 \* hibernate 基础类  
IMG_425 \* @author fengyan  
IMG_426 \* date 2007-01-09 02:32  
IMG_427 \*/  
IMG_428import org.hibernate.HibernateException;  
IMG_429import org.hibernate.Session;  
IMG_430import org.hibernate.SessionFactory;  
IMG_431import org.hibernate.cfg.Configuration;  
IMG_432  
IMG_433public class HibernateUtil {  
IMG_434      
IMG_435    private static final SessionFactory sessionFactory;  
IMG_436      
IMG_437    static  
IMG_438    {  
IMG_439        try  
IMG_440        {  
IMG_441            Configuration config = new Configuration().configure("/hibernate/hibernate.cfg.xml");  
IMG_442            sessionFactory = config.buildSessionFactory();  
IMG_443        }  
IMG_444        catch(Throwable e)  
IMG_445        {  
IMG_446            throw new ExceptionInInitializerError(e);  
IMG_447        }  
IMG_448    }  
IMG_449      
IMG_450    public static final ThreadLocal session = new ThreadLocal();  
IMG_451      
IMG_452    public static Session currentSession() throws HibernateException  
IMG_453    {  
IMG_454        Session s = (Session)session.get();  
IMG_455        //Open a new Session,if this Thread has none yet  
IMG_456        if(s == null || !s.isOpen())  
IMG_457        {  
IMG_458            s = sessionFactory.openSession();  
IMG_459            session.set(s);  
IMG_460        }  
IMG_461        return s;  
IMG_462    }  
IMG_463      
IMG_464    public static void closeSession() throws HibernateException  
IMG_465    {  
IMG_466        Session s = (Session)session.get();  
IMG_467        session.set(null);  
IMG_468        if(s != null)  
IMG_469            s.close();  
IMG_470    }  
IMG_471  
IMG_472}  
IMG_473

设计控制类

IMG_474package hibernate.servlet;  
IMG_475/\*\*  
IMG_476 \* @author fengyan  
IMG_477 \* date  2007-01-09 02:44  
IMG_478 \* 设计Hibernate控制类  
IMG_479 \*/  
IMG_480import hibernate.HibernateUtil;  
IMG_481import hibernate.PO.TRegister;  
IMG_482  
IMG_483import java.io.IOException;  
IMG_484import java.io.PrintWriter;  
IMG_485  
IMG_486import javax.servlet.ServletException;  
IMG_487import javax.servlet.http.HttpServlet;  
IMG_488import javax.servlet.http.HttpServletRequest;  
IMG_489import javax.servlet.http.HttpServletResponse;  
IMG_490  
IMG_491import org.hibernate.HibernateException;  
IMG_492import org.hibernate.Session;  
IMG_493import org.hibernate.Transaction;  
IMG_494  
IMG_495public class RegisterServlet extends HttpServlet {  
IMG_496  
IMG_497    private static final String CONTENT\_TYPE = "text/html;charset=GBK";  
IMG_498    public void init() throws ServletException {  
IMG_499        // Put your code here  
IMG_500    }  
IMG_501    public void destroy() {  
IMG_502        super.destroy(); // Just puts "destroy" string in log  
IMG_503        // Put your code here  
IMG_504    }  
IMG_505  
IMG_506      
IMG_507    public void doGet(HttpServletRequest request, HttpServletResponse response)  
IMG_508            throws ServletException, IOException {  
IMG_509  
IMG_510        response.setContentType(CONTENT\_TYPE);  
IMG_511        request.setCharacterEncoding("GBK");  
IMG_512          
IMG_513        PrintWriter out = response.getWriter();  
IMG_514          
IMG_515        String userName = request.getParameter("userName");  
IMG_516        String userPwd = request.getParameter("userPwd");  
IMG_517        String sex = request.getParameter("sex");  
IMG_518        int age = Integer.parseInt(request.getParameter("age"));  
IMG_519          
IMG_520        TRegister rg = new TRegister();  
IMG_521        rg.setAge(age);  
IMG_522        rg.setSex(sex);  
IMG_523        rg.setUserName(userName);  
IMG_524        rg.setUserPwd(userPwd);  
IMG_525          
IMG_526        Session session = HibernateUtil.currentSession();//生成Session实例  
IMG_527        Transaction tx = session.beginTransaction();  
IMG_528          
IMG_529        try  
IMG_530        {  
IMG_531            session.save(rg);    //保存持久类对象  
IMG_532            tx.commit();        //提交到数据库  
IMG_533            session.close();  
IMG_534            response.sendRedirect("registerOK.jsp");  
IMG_535        }  
IMG_536        catch(HibernateException e)  
IMG_537        {  
IMG_538            e.printStackTrace();  
IMG_539            tx.rollback();  
IMG_540        }  
IMG_541          
IMG_542    }  
IMG_543  
IMG_544    public void doPost(HttpServletRequest request, HttpServletResponse response)  
IMG_545            throws ServletException, IOException {  
IMG_546        doGet(request,response);  
IMG_547          
IMG_548    }  
IMG_549  
IMG_550  
IMG_551  
IMG_552}