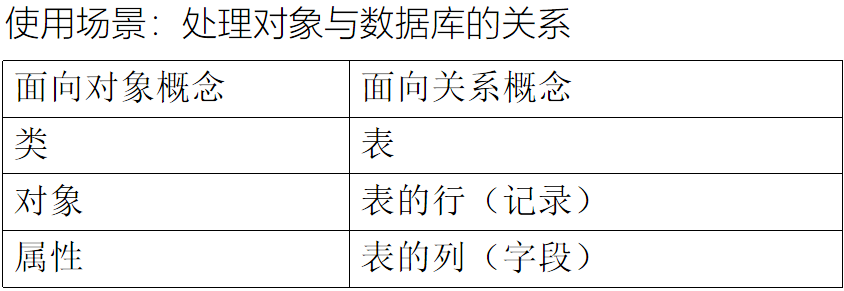
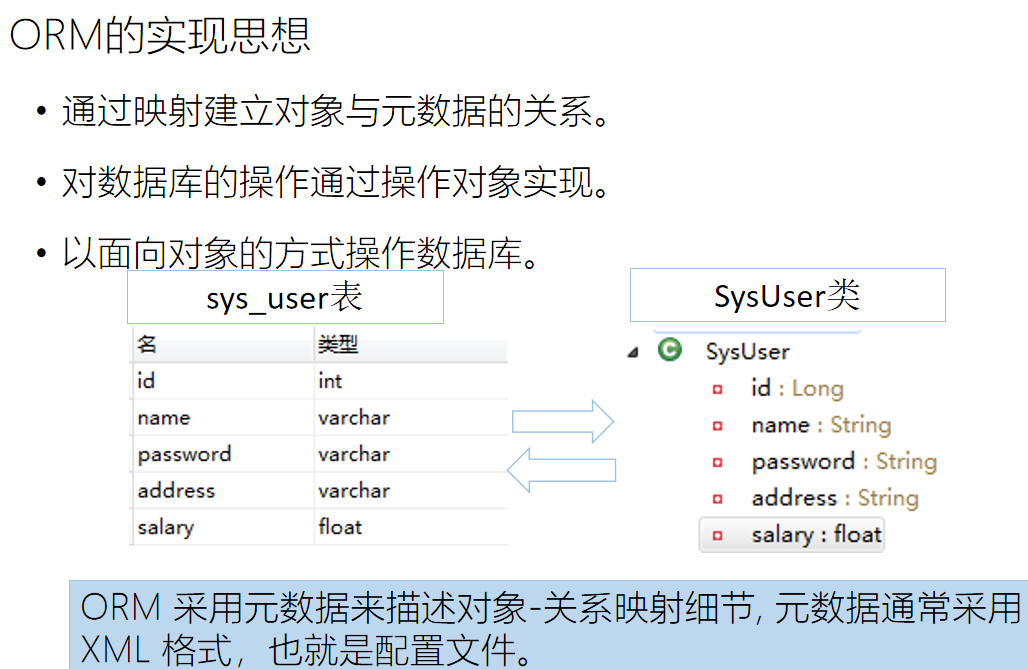
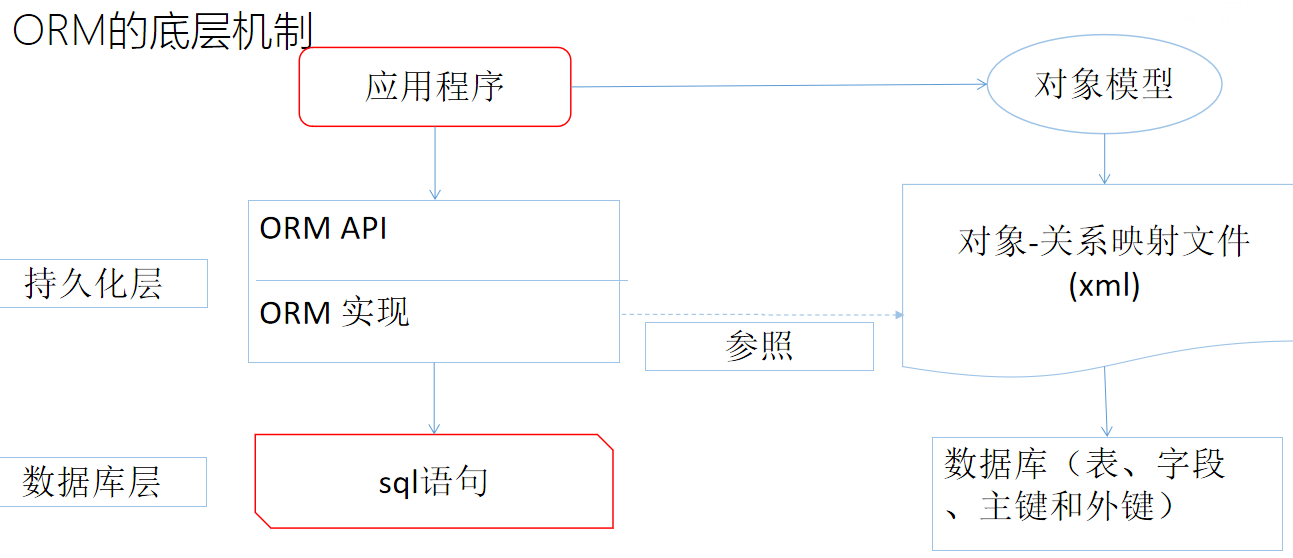
## ORM介绍

ORM（ Object Relational Mapping ）：对象关系映射，用于实现面向对象编程语言里不同系统之间数据的转换。







## hibernate是什么

Hibernate： 面向java环境的对象/关系数据库映射工具，ORM的具体实现。

开源的持久层框架。

基于ORM(Object/Relational Mapping)思想的具体实现。

连接java应用和数据库的中间件。

对JDBC进行封装，负责java对象的持久化。

封装对数据库的访问细节，以面相对象的方式操作数据库。

## 使用Hibernate的优点

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使用java的反射机制，而不是字节码增强程序类实现透明性

Hibernate最终生成sql语句并执行，性能较好。

灵活性高，支持各种关联关系。

支持多种数据库类型，使用hibernate开发的程序可以在不同数据库中移植。

## Hibernate资源下载

<https://sourceforge.net/projects/hibernate/files/hibernate-orm/>

## 使用hibernate操作数据库的步骤

### 创建一个数据表，记住数据库名和表名

user表： 编号id、用户名username、密码password、生日birthday、手机号phone

### pom.xml

|  |
| --- |
| <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.30</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>4.1.1.Final</version>  </dependency> |

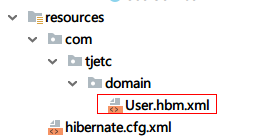
### hibernate.cfg.xml src/main/resources

|  |
| --- |
| <?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.username"*>root</property>  <property name=*"connection.url"*>  jdbc:mysql://localhost:3306/cc  </property>  <property name=*"hibernate.dialect"*>  org.hibernate.dialect.MySQLDialect  </property>  <property name=*"connection.password"*>123456</property>  <property name=*"connection.driver\_class"*>  com.mysql.jdbc.Driver  </property>  <property name=*"show\_sql"*>true</property>  <property name=*"format\_sql"*>true</property>  <property name=*"hbm2ddl.auto"*>update</property>  <mapping resource=*"com/tjetc/domain/User.hbm.xml"*/>  </session-factory>  </hibernate-configuration> |

### 在src/main/java中创建实体类User

|  |
| --- |
| public class User {  private Integer id;  private String username;  private String password;  private Date birthday;  private String phone;  @Override  public String toString() {  return "User{" +  "id=" + id +  ", username='" + username + '\'' +  ", password='" + password + '\'' +  ", birthday=" + birthday +  ", phone='" + phone + '\'' +  '}';  }  public User(Integer id, String username, String password, Date birthday, String phone) {  this.id = id;  this.username = username;  this.password = password;  this.birthday = birthday;  this.phone = phone;  }  public User() {  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  } |

### 在src/main/resources中创建对应的xml文件 User.hbm.xml 注意和实体类的包名和类名保持一致



|  |
| --- |
| <?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="com.tjetc.domain">  <class name="User">  <id name="id">  <generator class="native" />  </id>  <property name="username" />  <property name="password" />  <property name="birthday" type="date" />  <property name="phone" />  </class>  </hibernate-mapping> |

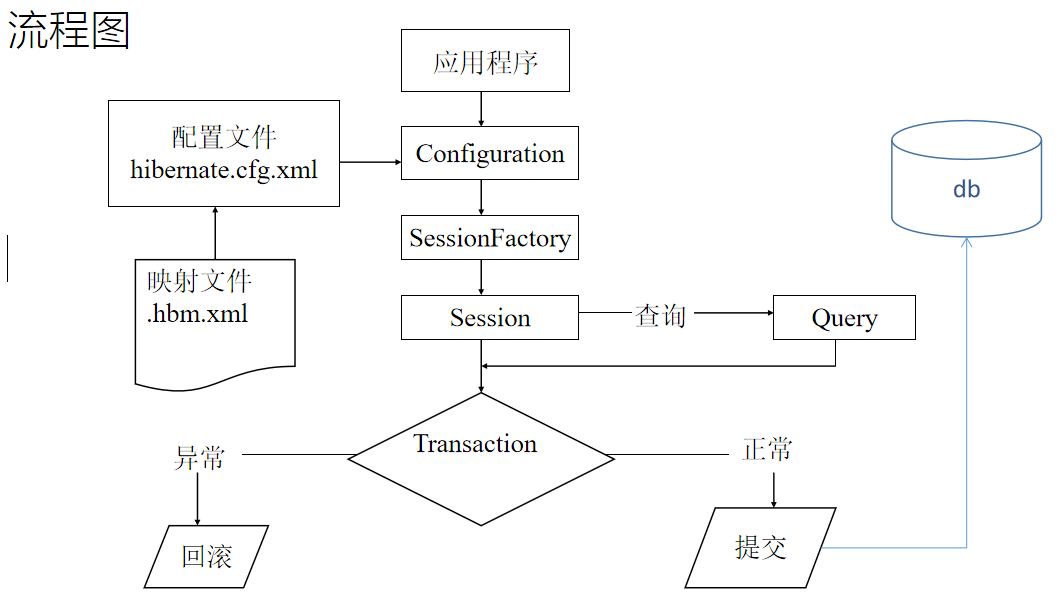
### 修改hibernage.cfg.xml 中对文件的映射文件

|  |
| --- |
| <?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.username">root</property>  <property name="connection.url">  jdbc:mysql://localhost:3306/b?serverTimezone=GMT%2B8  </property>  <property name="hibernate.dialect">  org.hibernate.dialect.MySQLDialect  </property>  <property name="connection.password">123456</property>  <property name="conne.Driverction.driver\_class">  com.mysql.cj.jdbc.Driver  </property>  <property name="show\_sql">true</property>  <property name="format\_sql">true</property>  <property name="hbm2ddl.auto">update</property>  <mapping resource="com/tjetc/domain/User.hbm.xml"/>  </session-factory>  </hibernate-configuration> |

### 增删改查测试:

|  |
| --- |
| <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.13.2</version>  </dependency> |
| public final class HibernateUtil {  private static SessionFactory sessionFactory;  private HibernateUtil() {  }  static {  try {  sessionFactory= new Configuration().configure().buildSessionFactory();  } catch (Throwable e) {  throw new ExceptionInInitializerError(e);  }  }  public static SessionFactory getSessionFactory() {  return sessionFactory;  }  } |
| public class Test1 {  private SessionFactory factory= HibernateUtil.getSessionFactory();  @Test  public void testAdd() {  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  User user = new User();  user.setId(3);  user.setUsername("lizi3");  user.setPassword("1233");  user.setBirthday(new Date());  user.setPhone("133123456789");  session.save(user);  transaction.commit();  session.close();  }  @Test  public void testUpdate() {  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  User user = new User();  user.setId(1);  user.setUsername("lizi3");  user.setPassword("1233");  user.setBirthday(new Date(System.currentTimeMillis()-10000000));  user.setPhone("18912345678");  session.update(user);  transaction.commit();  session.close();  }  @Test  public void testDel() {  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  User user = new User();  user.setId(1);  session.delete(user);  transaction.commit();  session.close();  }  @Test  public void testSelect() {  Session session = factory.openSession();  List list = session.createQuery("from User where username like ?").setString(0, "%z%").setFirstResult(0).setMaxResults(2).list();  System.out.println("list = " + list);  session.close();  }  @Test  public void testCount() {  Session session = factory.openSession();  int count=((Long)session.createQuery("select count(\*) from User where username like ?").setString(0,"%%").uniqueResult()).intValue();  System.out.println("count = " + count);  session.close();  }  @Test  public void testFindById() {  Session session = factory.openSession();  User user = (User) session.createQuery("from User where id=?").setInteger(0, 1).uniqueResult();  System.out.println("user = " + user);  session.close();  }  } |

## hibernate运行流程



应用程序先调用Configuration类,该类读取Hibernate配置文件及映射文件中的信息;

读取后创建SessionFactory对象，该对象封装了配置配置信息;

通过工厂类SessionFactory创建Session对象（数据库操作对象）；

通过Session对象生成Transaction对象（事务控制对象）；

调用Session对象的方法进行CURD等操作；

提交事物。

关闭资源

## 注解

hibernate.cfg.xml

|  |
| --- |
| <mapping class=*"com.tjetc.domain.User"*/> |

User.java

|  |
| --- |
| @Entity  public class User {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private Integer id;  private String username;  private String password;  @Type(type = "date")  private Date birthday;  private String phone;  @Override  public String toString() {  return "User{" +  "id=" + id +  ", username='" + username + '\'' +  ", password='" + password + '\'' +  ", birthday=" + birthday +  ", phone='" + phone + '\'' +  '}';  }  public User(Integer id, String username, String password, Date birthday, String phone) {  this.id = id;  this.username = username;  this.password = password;  this.birthday = birthday;  this.phone = phone;  }  public User() {  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  } |

## Struts2 Hibernate实现用户登录功能

任务名称 基于struts2整合hibernate实现数据库用户信息表CURD操作设计

开发环境 Intellij IDEA Chrome

完成功能 Struts2整合Hibernate实现用户登录，登录后进入主页面，提供用户的列表，同时提供新增链接，实现CRUD操作。

### pom.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.tjetc</groupId>  <artifactId>day0620struts3</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>org.apache.struts</groupId>  <artifactId>struts2-core</artifactId>  <version>2.5.2</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>4.0.1</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>4.1.1.Final</version>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.30</version>  </dependency>  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <version>2.2</version>  <configuration>  <port>8080</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

### web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"  version="4.0">  <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  </web-app> |

### struts.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"  "http://struts.apache.org/dtds/struts-2.5.dtd">  <struts>  <package name="hello" namespace="/" extends="struts-default">  <action name="hello" class="com.tjetc.action.HelloAction">  <result name="hello">/hello.jsp</result>  </action>  </package>  <package name="login" namespace="/" extends="struts-default">  <global-allowed-methods>regex:.\*</global-allowed-methods>  <action name="user\_\*" class="com.tjetc.action.UserAction" method="{1}">  <result name="msg">msg.jsp</result>  <result name="toList" type="redirect">/user\_list</result>  <result name="list">/list.jsp</result>  <result name="update">/update.jsp</result>  </action>  </package>  </struts> |

### hibernate.cfg.xml

|  |
| --- |
| <?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.username">root</property>  <property name="connection.url">  jdbc:mysql://localhost:3306/b?serverTimezone=GMT%2B8  </property>  <property name="hibernate.dialect">  org.hibernate.dialect.MySQLDialect  </property>  <property name="connection.password">123456</property>  <property name="connection.driver\_class">  com.mysql.jdbc.Driver  </property>  <property name="show\_sql">true</property>  <property name="format\_sql">true</property>  <property name="hbm2ddl.auto">update</property>  <mapping resource="com/tjetc/domain/User.hbm.xml"/>  </session-factory>  </hibernate-configuration> |

### User.hbm.xml

|  |
| --- |
| <?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="com.tjetc.domain">  <class name="com.tjetc.domain.User" table="user">  <id name="id">  <generator class="native" />  </id>  <property name="username" />  <property name="password" />  <property name="birthday" type="date" />  <property name="phone" />  </class>  </hibernate-mapping> |

### Login.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <base href="${pageContext.request.contextPath}/"/>  </head>  <body>  <form action="user\_login" method="post">  用户名: <input type="text" name="username"> <br>  密码: <input type="password" name="password"> <br>  <input type="submit" value="提交">  </form>  </body>  </html> |

### UserAction.java

|  |
| --- |
| package com.tjetc.action;  import com.opensymphony.xwork2.ModelDriven;  import com.sun.org.apache.regexp.internal.RE;  import com.tjetc.domain.User;  import com.tjetc.service.UserService;  import org.apache.struts2.ServletActionContext;  import java.util.List;  public class UserAction implements ModelDriven<User> {  private UserService userService=new UserServiceImpl();  private User user=new User();  private String msg;  public String getMsg() {  return msg;  }  public void setMsg(String msg) {  this.msg = msg;  }  public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }  @Override  public User getModel() {  return user;  }  public String login(){  User user1=userService.findByUsername(user.getUsername());  if (user1==null){  msg="用户名不存在";  return "msg";  }  else if (!user1.getPassword().equals(user.getPassword())){  msg="密码不正确";  return "msg";  }  else {  ServletActionContext.getRequest().getSession().setAttribute("user", user1);  return "toList";  }  }  public String list(){  String username = user.getUsername();  username=username==null?"":username;  List<User> list=userService.list(username);  ServletActionContext.getRequest().setAttribute("list", list);  return "list";  }  public String findById(){  user=userService.findById(user.getId());  return "update";  }  public String update(){  System.out.println("UserAction.update([])方法:"+user);  boolean b=userService.update(user);  return "toList";  }  public String add(){  System.out.println("UserAction.add([])方法:"+user);  boolean b=userService.add(user);  return "toList";  }  public String del(){  System.out.println("UserAction.del([])方法:"+user);  boolean b=userService.del(user);  return "toList";  }  } |
|  |
|  |

### UserDao

|  |
| --- |
| public interface UserDao {  User findByUsername(String username);  List<User> list(String username);  User findById(Integer id);  boolean add(User user);  boolean update(User user);  boolean del(User user);  } |

### HibernateUtil

|  |
| --- |
| public final class HibernateUtil {  private static SessionFactory sessionFactory;  private HibernateUtil() {  }  static {  try {  sessionFactory= new Configuration().configure().buildSessionFactory();  } catch (Throwable e) {  throw new ExceptionInInitializerError(e);  }  }  public static SessionFactory getSessionFactory() {  return sessionFactory;  }  } |

### UserDaoImpl

|  |
| --- |
| public class UserDaoImpl implements com.tjetc.dao.UserDao {  private static SessionFactory sessionFactory = HibernateUtil.getSessionFactory();  @Override  public User findByUsername(String username) {  Session session = sessionFactory.openSession();  User user = (User) session.createQuery("from User where username=?").setString(0, username).uniqueResult();  session.close();  return user;  }  @Override  public List<User> list(String username) {  Session session = sessionFactory.openSession();  List list = session.createQuery("from User u where u.username like ?").setString(0,"%"+username+"%").list();  session.close();  return list;  }  @Override  public User findById(Integer id) {  Session session = sessionFactory.openSession();  User user = (User) session.createQuery("from User where id=?").setInteger(0, id).uniqueResult();  session.close();  return user;  }  @Override  public boolean add(User user) {  Session session = sessionFactory.openSession();  Transaction transaction = session.beginTransaction();  Serializable save = session.save(user);  transaction.commit();  session.close();  return save!=null;  }  @Override  public boolean update(User user) {  try {  Session session = sessionFactory.openSession();  Transaction transaction = session.beginTransaction();  session.update(user);  transaction.commit();  session.close();  return true;  } catch (HibernateException e) {  e.printStackTrace();  }  return false;  }  @Override  public boolean del(User user) {  try {  Session session = sessionFactory.openSession();  Transaction transaction = session.beginTransaction();  session.delete(user);  transaction.commit();  session.close();  return true;  } catch (HibernateException e) {  e.printStackTrace();  }  return false;  }  } |

### UserService

|  |
| --- |
| public interface UserService {  User findByUsername(String username);  List<User> list(String username);  User findById(Integer id);  boolean update(User user);  boolean del(User user);  boolean add(User user);  } |

### 10．UserServiceImpl

|  |
| --- |
| public class UserServiceImpl implements UserService {  private UserDao userDao = new UserDaoImpl();  @Override  public User findByUsername(String username) {  return userDao.findByUsername(username);  }  @Override  public List<User> list(String username) {  return userDao.list(username);  }  @Override  public User findById(Integer id) {  return userDao.findById(id);  }  @Override  public boolean update(User user) {  return userDao.update(user);  }  @Override  public boolean del(User user) {  return userDao.del(user);  }  @Override  public boolean add(User user) {  return userDao.add(user);  }  } |

### User

|  |
| --- |
| public class User {  private Integer id;  private String username;  private String password;  private Date birthday;  private String phone;  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  } |

### list.jsp

|  |
| --- |
| <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <base href="${pageContext.request.contextPath}/"/>  <script>  function query() {  location.href='user\_list?username='+document.getElementById("username").value  }  function add() {  location.href='add.jsp'  }  </script>  </head>  <body>  <div>  <input type="text" id="username" value="${username}">  <button onclick="query()">查询</button>  <button onclick="add()">提交</button>  </div>  <table border="1" cellspacing="0" width="70%">  <tr align="center">  <th>序号</th>  <th>用户名</th>  <th>密码</th>  <th>生日</th>  <th>手机</th>  <th>操作</th>  </tr>  <c:forEach items="${list}" var="user">  <tr align="center">  <td>${user.id}</td>  <td>${user.username}</td>  <td>${user.password}</td>  <td>${user.birthday}</td>  <td>${user.phone}</td>  <td>  <a href="user\_findById?id=${user.id}">修改</a>  <a href="user\_del?id=${user.id}">删除</a>  </td>  </tr>  </c:forEach>  </table>  </body>  </html> |

### add.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <base href="${pageContext.request.contextPath}/"/>  </head>  <body>  <form action="user\_add" method="post">  用户名:<input type="text" name="username" /><br/>  密码:<input type="password" name="password" /><br/>  生日:<input type="date" name="birthday" /><br/>  手机:<input type="text" name="phone" /><br/>  <input type="submit" value="提交"/><br/>  </form>  </body>  </html> |

### update.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <base href="${pageContext.request.contextPath}/"/>  </head>  <body>  <form action="user\_update" method="post">  <input type="hidden" name="id" value="${user.id}">  用户名:<input type="text" name="username" value="${user.username}"/><br/>  密码:<input type="password" name="password" value="${user.password}"/><br/>  生日:<input type="date" name="birthday" value="${user.birthday}" /><br/>  手机:<input type="text" name="phone" value="${user.phone}" /><br/>  <input type="submit" value="提交"/><br/>  </form>  </body>  </html> |

### msg.jsp

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Insert title here</title>  </head>  <body>  ${msg} <br>  <a href="login.jsp">去登录</a>  </body>  </html> |

