1706班第10周第1天课堂笔记

HRS - Human Resource System

需求分析

- 部门管理 - 新增 / (删除) / 更新 / 列出
- 员工管理 - 新增 / 删除 / 更新 / 分页查看 / 查看详情
- 系统功能 - 用户登录 / 注册 / 注销

设计

1. 业务实体分析

- Dept (id, name, location)
- Emp (id, name, job, gender, sal, mgr, hireDate, status, photo, tel, dept)
- User (username, password, email)

2. 数据库建模

```sql
DROP DATABASE IF EXISTS `hrs`;
CREATE DATABASE `hrs` default charset utf8;
USE `hrs`;

DROP TABLE IF EXISTS `tb_dept`;
CREATE TABLE `tb_dept` (
  `dno` int(11) NOT NULL,
  `dname` varchar(20) NOT NULL,
  `dloc` varchar(50) NOT NULL,
  PRIMARY KEY (`dno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `tb_emp`;
CREATE TABLE `tb_emp` (
  `eno` int(11) NOT NULL,
  `ename` varchar(20) NOT NULL,
  `esex` bit(1) DEFAULT b'1',
  `ejob` varchar(20) NOT NULL,
  `emgr` int(11) DEFAULT NULL,
  `esal` float DEFAULT NULL,
  `ehiredate` date DEFAULT NULL,
  `estatus` bit(1) DEFAULT b'1',
  `ephoto` varchar(255) DEFAULT NULL,
  `etel` char(11) DEFAULT NULL,
  `dno` int(11) NOT NULL,
  PRIMARY KEY (`eno`),
  KEY `fk_emp_dno` (`dno`),
  KEY `fk_emp_emgr` (`emgr`),
  CONSTRAINT `fk_emp_dno` FOREIGN KEY (`dno`) REFERENCES `tb_dept` (`dno`) ON
UPDATE CASCADE,
  CONSTRAINT `fk_emp_emgr` FOREIGN KEY (`emgr`) REFERENCES `tb_emp` (`eno`)
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `tb_user`;
CREATE TABLE `tb_user` (
  `username` varchar(20) NOT NULL,
  `password` char(32) NOT NULL,
  `email` varchar(255) NOT NULL,
  PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
INSERT INTO `tb_dept` VALUES ('10','财务部','北京'), ('20','研发部','成都'),
('30','销售部','深圳');
INSERT INTO `tb_user` VALUES
('jackfrued','4297f44b13955235245b2497399d7a93','jackfrued@126.com');
```

架构

- 表示层：处理用户请求，显示视图给用户
  - MVC架构模式 - 实现模型（数据）和视图（显示）的解耦合
    - Controller - Servlet
    - Model - JavaBean
    - View - JSP / EL / JSTL

```java
package com.qfedu.controller;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.qfedu.service.UserService;
import com.qfedu.service.impl.UserServiceImpl;

@WebServlet(urlPatterns = "/login", loadOnStartup = 1)
public class LoginServlet extends HttpServlet {
    private UserService userService = new UserServiceImpl();

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        if (username != null && password != null) {
            if (userService.login(username, password)) {
                req.getSession().setAttribute("username", username);
                resp.sendRedirect("dept");
            } else {
                req.setAttribute("hint", "用户名或密码错误!");
                req.getRequestDispatcher("login.jsp").forward(req,
resp);
            }
        } else {
            req.setAttribute("hint", "请输入有效的登录信息!");
            req.getRequestDispatcher("login.jsp").forward(req, resp);
        }
    }
}
```

- 业务层：处理核心的业务逻辑
  - 事务脚本模式 (贫血模型)
    - 用户的每个请求称为一个事务
    - 每个事务对应业务层的一个方法
    - 业务方法的代码就是一段事务脚本

```
package com.qfedu.service;

import com.qfedu.domain.User;

/**
 * 用户相关业务接口
 * @author 骆昊
 *
 */
public interface UserService {

    /**
     * 登录
     * @param username 用户名
     * @param password 密码
     * @return 登录成功返回true否则返回false
     */
    boolean login(String username, String password);

    /**
     * 注册
     * @param user 用户对象
     * @return 注册成功返回true否则返回false
     */
    boolean register(User user);
}
```

```java
package com.qfedu.service.impl;

import org.apache.commons.codec.digest.DigestUtils;

import com.qfedu.domain.User;
import com.qfedu.persistence.UserDao;
import com.qfedu.persistence.impl.UserDaoImpl;
import com.qfedu.service.UserService;

/**
 * 用户相关业务实现类
 * @author 骆昊
 *
 */
public class UserServiceImpl implements UserService {
    private UserDao userDao = new UserDaoImpl();

    @Override
    public boolean login(String username, String password) {
        User temp = userDao.findByUsername(username);
        if (temp != null) {
            String md5 = DigestUtils.md5Hex(password);
            return temp.getPassword().equals(md5);
        }
        return false;
    }

    @Override
    public boolean register(User user) {
        User temp = userDao.findByUsername(user.getUsername());
        if (temp == null) {
            String md5 = DigestUtils.md5Hex(user.getPassword());
            user.setPassword(md5);
            return userDao.save(user);
        }
        return false;
    }

}
```

- 持久层：完成持久化操作(CRUD)
  - DAO模式
    - Data Accessor - 封装CRUD操作的方法
    - Active Domain Object - 以对象为单位组织数据
  - 实体类

```java
package com.qfedu.domain;

import java.io.Serializable;

/**
 * 用户类
 * @author 骆昊
 *
 */
public class User implements Serializable {
    private String username;        // 用户名
    private String password;            // 密码
    private String email;               // 邮箱

    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 String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

}
```

- DAO接口 - 层与层之间通过接口关联起来

```java
package com.qfedu.persistence;

import com.qfedu.domain.User;

/**
 * 用户数据访问对象
 * @author 骆昊
 *
 */
public interface UserDao {

    /**
     * 根据用户名查找用户
     * @param username 用户名
     * @return 用户对象或null
     */
    User findByUsername(String username);

    /**
     * 保存用户
     * @param user 用户对象
     * @return 保存成功返回true否则返回false
     */
    boolean save(User user);
}
```

- DAO实现类

```java
package com.qfedu.persistence.impl;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import com.qfedu.domain.User;
import com.qfedu.persistence.UserDao;

/**
 * 用户数据访问对象的实现类
 * @author 骆昊
 *
 */
public class UserDaoImpl implements UserDao {

    static {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    @Override
    public User findByUsername(String username) {
        User user = null;
        try (Connection connection = DriverManager.getConnection(
                "jdbc:mysql:///hrs", "root", "123456")) {
            PreparedStatement stmt = connection.prepareStatement(
                    "select password, email from tb_user where
username=?");
            stmt.setString(1, username);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                user = new User();
                user.setUsername(username);
                user.setPassword(rs.getString("password"));

                user.setEmail(rs.getString("email"));
```

```
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return user;
    }

    @Override
    public boolean save(User user) {
        try (Connection connection = DriverManager.getConnection(
                "jdbc:mysql:///hrs", "root", "123456")) {
            PreparedStatement stmt = connection.prepareStatement(
                    "insert into tb_user values (?,?,?)");
            stmt.setString(1, user.getUsername());
            stmt.setString(2, user.getPassword());
            stmt.setString(3, user.getEmail());
            return stmt.executeUpdate() == 1;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

}
```

> 说明：写代码的终极原则是："高内聚低耦合" (high cohesion low coupling)