第四次实验

实验要求

1. 通过实例开发，熟练掌握JSP语法及相关标签的应用。
2. 熟练掌握Servlet作为控制器，实现用户登录的处理和界面跳转。

采用JSP+Servlet+JavaBean+JDBC方式开发一个Web登录程序。

登录

要求：数据库中建一个“用户名-密码”表，用户由页面上输入用户名和密码，单击【登录】按钮提交，程序通过JDBC访问数据库中的表来验证用户，验证通过转到主页并回显欢迎信息，否则跳转至出错页。

实验步骤

（1）建立数据库与表

在MySQL test数据库中，建立一个用户表userTable，表结构如下表所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 自增 | 描述 |
| id | int | 是 | 增1 | ID号 |
| username | varchar(20) |  |  | 用户名 |
| password | varchar(20) |  |  | 密码 |

字段包括：id、username和password。其中id设为自动增长的int型，并设为主键。username和password都设为varchar型。表建好后，向其中录入两条数据记录。

（2）编写JSP

本例只须写3个JSP文件：login.jsp（登录页）、main.jsp（主页）和error.jsp（出错页)

1. login.jsp

【2】main.jsp

【3】error.jsp

（3）构造JavaBean、创建JDBC

目录结构

├─src

│ └─main

│ ├─java

│ │ └─tech

│ │ └─haonan

│ │ ├─dao // 持久层 用来与数据库进行交互

│ │ │ UserDao.java

│ │ │ UserDaoImpl.java

│ │ ├─entity //实体类

│ │ │ User.java

│ │ ├─filter // filter 过滤非法请求和设置字符集编码

│ │ │ EncodingFilter.java

│ │ │ UserNameFilter.java

│ │ ├─Service //中间层 负责提供接口给servlet层 抽象Dao层

│ │ │ LoginService.java //接口实现类

│ │ │ LoginServiceImpl.java //接口

│ │ └─servlet // servlet层

│ │ LoginServlet.java

│ └─webapp

│ │ error.jsp // 登录密码和用户名不匹配跳转

│ │ login.jsp // 登录界面

│ │ main.jsp // 登录成功返回信息

│ └─WEB-INF

│ web.xml //web.xml 配置文件

实验说明

这次实验主要是糅合数据库,servlet,前端jsp的知识进行一次全流程的实现,需要一定的学习能力和知识储备,从前端发送数据,servlet接受数据调用service层接口,service层调用Dao层实现类与数据库进行交互,数据sql语句查询结果进行层层返回到servlet层进行验证,根据情况进行合适的跳转.整个流程需要搞懂各个层之间的衔接.

核心代码

1. 以下是login.jsp文件内容

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

    <title>Title</title>

</head>

<script>

    window.onload = function (){

        let oButton = document.getElementById("submit");

        let oUsernameText = document.getElementById("username");

        let oPasswordText = document.getElementById("password");

        oButton.onclick = function (){

            if(oUsernameText.value ===null ||oUsernameText.value ==="" ){

                alert("用户名不能为空");

            }

            if(oPasswordText.value === null ||oPasswordText.value === "" ){

                alert("密码不能为空");

            }

        }

    }

</script>

<body>

    <form action="/login " method="post">

        用户名 <input type="text"  id="username" name="username"> <br>

        密码 <input type="password" id="password" name="password" >  <br>

        <input type="submit" id="submit" value="提交" >

    </form>

</body>

</html>

2.以下是main.jsp文件核心代码

<!--获取从后端得到的name值-->

<h1>登录成功，欢迎您  <%=username%></h1>

<h2>

    <a href="/login.jsp">点击重新跳转到登录界面</a>

</h2>

3.以下是error.jsp文件核心代码

<h1><%=str%></h1>

<h2>

    <a href="/login.jsp">点击重新跳转到登录界面</a>

</h2>

1. 以下是web.xml文件的配置

因为可以直接在filter中使用注解,所以这里就可以省略对filter的bean配置,所以就很省事.

  <display-name>Archetype Created Web Application</display-name>

  <welcome-file-list>

    <welcome-file>login.jsp</welcome-file>

  </welcome-file-list>

1. 以下是EncodingFilter.java文件核心代码

@WebFilter(filterName = "EncodingFilter", urlPatterns = "/\*")

public class CharacterEncodingFilter implements Filter {

    @Override

    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {

        // 设置信息编码格式为utf-8

        servletRequest.setCharacterEncoding("utf-8");

        servletResponse.setCharacterEncoding("utf-8");

        servletResponse.setContentType("text/html;charset=UTF-8");

        // 将所有的请求放行

        filterChain.doFilter(servletRequest, servletResponse);

    }

}

1. 以下是UserNameFilter.java的核心代码

@WebFilter(filterName = "UserNameFilter" , urlPatterns = "/login")

public class UserNameFilter implements Filter {

    @Override

    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {

        // 获取username和password

        String name =  servletRequest.getParameter("username");

        String password = servletRequest.getParameter("password");

        boolean flag = false;

        if(name == null || name.equals("") ){

            // 将信息存储到Request中 后期jsp中可以使用

            servletRequest.setAttribute("error1","用户名不能为空");

            flag = true;

        }

        if(password == null || password.equals("") ){

            // 将信息存储到Request中 后期jsp中可以使用

            servletRequest.setAttribute("error2","密码不能为空");

            flag = true;

        }

        if(flag)

            servletRequest.getRequestDispatcher("error.jsp").forward(servletRequest,servletResponse);

        filterChain.doFilter(servletRequest,servletResponse);

    }

}

1. 以下是LoginServlet.java的核心代码

@WebServlet("/login")

public class LoginServlet extends HttpServlet{

    @Override

    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        String username =  req.getParameter("username");

        String password = req.getParameter("password");

        System.out.println(username + " " + password);

        // new一个Secvice层的对象 用于调用它的findByUserName方法 查询用户是否存在

        LoginService loginService = new LoginServiceImpl();

        //去查询用户输入的登录名和密码是否正确  层层调用 查询

        User user = loginService.findByUserName(username,password);

        if(user !=null){   // 调用函数返回了对象不为空 那么说明数据库存在信息

            req.setAttribute("username",username);

            req.getRequestDispatcher("main.jsp").forward(req,resp);

        }else{  //如果返回值为null 说明用户名密码错误

            resp.getWriter().println("<h1>用户名或者密码输入错误</h1>");

        }

    }

    @Override

    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        doGet(req,resp);

    }

}

1. 以下是UserDao.java (接口interface)的文件内容

public interface UserDao {

    User findByUserName(String username,String password);

}

1. 以下是UserDao.java 的实现类 UserDaoImpl.java文件

public class UserDaoImpl implements UserDao {

    @Override

    public User findByUserName(String username, String password) {

        User user = new User();

        Connection con = null;

        Statement sql = null;

        ResultSet rs = null;

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");  //com.mysql.jdbc.Driver已经弃用了，要加上cj

        }

        catch(Exception e) {

            System.out.println("forNameError:" + e);

        }

        // 访问url

        String url = "jdbc:mysql://192.168.43.50:4000/web?useSSL=false&characterEncoding=utf-8&serverTimezone=GMT"; //要加上serverTimezone=GMT，否则会报错

        // mysql数据库 用户名和密码

        String userString = "root";

        String passString = "123456";

        try{

            // 获取链接

            con = DriverManager.getConnection(url, userString, passString);}

        catch(Exception e) {

            System.out.println(e);

        }

        try {

            sql = con.createStatement();

            // sql 查询语句

            String querySentence = "SELECT \* FROM userTable where username = ? and password = ?";

            PreparedStatement ps = con.prepareStatement(querySentence);

            // 顶替 sql语句的  ?

            ps.setString(1,username);

            ps.setString(2,password);

            // 执行sql语句

            rs = ps.executeQuery() ;

            while(rs.next()) {

                user.setId(rs.getInt(1));

                user.setUsername(rs.getString(2));

                user.setPassword(rs.getString(3));

                System.out.printf("%d\t", user.getId());

                System.out.printf("%s\t", user.getUsername());

                System.out.printf("%s\t",user.getPassword());

            }

            // 关闭连接

            con.close();

        }

        catch(SQLException e) {

            System.out.println(e);

        }

        if(user.getUsername().length() > 0){

           // 如果查询到了信息 那么 返回一个user对象 否则返回null

           return user;

        }else

            return null;

    }

}

1. LoginService.java 文件

public interface LoginService {

    User findByUserName(String username,String password);

}

1. LoginServiceImpl.java 文件

public class LoginServiceImpl implements LoginService {

    UserDao userDao = new UserDaoImpl();

    @Override

    public User findByUserName(String username,String password) {

//        return null;

        return userDao.findByUserName(username,password);

    }

}