第六次实验

实验要求

**1、实验目的**

* 1. 通过实例开发，熟练掌握MyBatis的集成和应用方法。

**2、实验内容**

2.1按照课上讲的方法实现SpringMVC，MyBatis集成过程：

要求：

(1)配置成功以后，注册页面，使用form设计如下界面：

(2)用post方法，提交给后台实现信息上报数据库，并查询用户名和密码是否匹配，如果匹配，Redirect到welcome.jsp，显示欢迎你[username]。如果不匹配，返回登录页面，并提示密码不匹配。

目录结构

└─src

├─main

│ ├─java

│ │ └─tech

│ │ └─haonan

│ │ ├─controller

│ │ │ LoginHandler.java // controller层

│ │ │

│ │ ├─dao

│ │ │ UserDao.java // 与数据库交流

│ │ │

│ │ ├─entity

│ │ │ User.java // 实体类

│ │ │

│ │ └─service

│ │ UserService.java // service 层

│ │ UserServiceImpl.java // 实现类

│ │

│ ├─resources

│ │ │ dbconfig.properties // 数据库配置

│ │ │ mybatis-config.xml // mybatis的主配置文件

│ │ │ springmvc.xml // springMVC 的配置文件 测试时候用

│ │ │ springMybatis-config.xml // spring 和mybatis的混合配置文件

│ │ │

│ │ └─mapper

│ │ UserMapper.xml // xml 写sql语句

│ │

│ └─webapp

│ │ hhh.jsp

│ │

│ └─WEB-INF

│ │ web.xml // web的主配置文件

│ │

│ └─pages

│ findAll.jsp // 展示所有数据库信息

│ index.jsp //主页提供三个选项登录,注册和查看所有用户

│ login.jsp // 登录jsp

│ loginSuccess.jsp //登录成功后跳转 失败之后跳转回原界面

│ register.jsp //注册jsp

│ registerError.jsp //如果用户名重复 就注册失败

│ registerSuccess.jsp //注册成功跳转

│

└─test

└─java

Test.java // junit 测试

TestMyBatis.java //测试mybatis

实验说明

这次实验是将springMVC和Mybatis结合起来,不过这次实验我也添加了spring的代码,ssm整合起来完整的框架,非常的好用.

核心代码

1. 以下是login.jsp文件内容

<body>

    <form action="<%=request.getContextPath()%>/login" method="post">

        <h4 style="color: red;">${error}</h4>

        登录界面 <br>

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

        密码  : <input type="text" name="password"> <br>

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

    </form>

</body>

1. 以下是success.jsp文件核心代码

<body>

    <h1>你好 , ${username}</h1>

</body>

1. 以下是web.xml文件的配置

 <!DOCTYPE web-app PUBLIC

 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

 "http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app version="4.0"

         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"

         metadata-complete="true">

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

  <welcome-file-list>

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

  </welcome-file-list>

  <!--配置监听器  监听ServletContext的创建 ContextLoaderListener

    默认值加载WEB-INT下的applicationContext.xml配置文件 但是可以传参进去 指名配置文件的地址

    配置spring的配置文件地址

  -->

  <context-param>

    <param-name>contextConfigLocation</param-name>

    <param-value>classpath:springMybatis-config.xml</param-value>

  </context-param>

  <listener>

    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

  </listener>

  <!--Spring MVC   配置前端控制器   拦截所有请求-->

  <servlet>

    <servlet-name>dispatcherServlet</servlet-name>

    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

    <init-param>

      <param-name>contextConfigLocation</param-name>

      <param-value>classpath:springmvc.xml</param-value>

    </init-param>

    <!--启动服务器,创建该servlet-->

    <load-on-startup>1</load-on-startup>

  </servlet>

  <servlet-mapping>

    <servlet-name>dispatcherServlet</servlet-name>

    <url-pattern>/</url-pattern><!--拦截所有请求-->

  </servlet-mapping>

  <!--字符编码过滤器 防止中文乱码 -->

  <filter>

    <filter-name>characterEncodingFilter</filter-name>

    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

    <init-param>

      <param-name>encoding</param-name>

      <param-value>UTF-8</param-value>

    </init-param>

  </filter>

  <filter-mapping>

    <filter-name>characterEncodingFilter</filter-name>

    <url-pattern>/\*</url-pattern>

  </filter-mapping>

  <!--加载静态文件-->

  <servlet-mapping>

    <servlet-name>default</servlet-name>

    <url-pattern>\*.js</url-pattern>  <!--js文件-->

  </servlet-mapping>

  <servlet-mapping>

    <servlet-name>default</servlet-name>

    <url-pattern>\*.css</url-pattern>  <!--css文件-->

  </servlet-mapping>

  <servlet-mapping>

    <servlet-name>default</servlet-name>

    <url-pattern>\*.jpg</url-pattern>  <!--图片文件-->

  </servlet-mapping>

</web-app>

1. 以下是springMybatis-config.xml

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:context="http://www.springframework.org/schema/context"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"

       xmlns:tx="http://www.springframework.org/schema/tx" xmlns:p="http://www.springframework.org/schema/p"

       xmlns:util="http://www.springframework.org/schema/util" xmlns:jdbc="http://www.springframework.org/schema/jdbc"

       xmlns:cache="http://www.springframework.org/schema/cache"

       xsi:schemaLocation="

        http://www.springframework.org/schema/context

        http://www.springframework.org/schema/context/spring-context.xsd

        http://www.springframework.org/schema/beans

        http://www.springframework.org/schema/beans/spring-beans.xsd

        http://www.springframework.org/schema/tx

        http://www.springframework.org/schema/tx/spring-tx.xsd

        http://www.springframework.org/schema/jdbc

        http://www.springframework.org/schema/jdbc/spring-jdbc-3.1.xsd

        http://www.springframework.org/schema/cache

        http://www.springframework.org/schema/cache/spring-cache-3.1.xsd

        http://www.springframework.org/schema/aop

        http://www.springframework.org/schema/aop/spring-aop.xsd

        http://www.springframework.org/schema/util

        http://www.springframework.org/schema/util/spring-util.xsd">

    <!-- 自动扫描 -->

    <context:component-scan base-package="tech.haonan">

        <!--配置spring 注解不扫描-->

        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>

    </context:component-scan>

    <!--加载dbconfig.properties文件  可以快速修改数据库的配置-->

    <context:property-placeholder location="classpath:dbconfig.properties"/>

<!--    <bean id="comboPooledDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">-->

<!--        <property name="driverClass"  value="com.mysql.cj.jdbc.Driver"/>-->

<!--        <property name="jdbcUrl"  value="jdbc:mysql://localhost:3306/mybatis?serverTimezone=UTC&amp;useUnicode=true&amp;characterEncoding=UTF-8"/>-->

<!--        <property name="user" value="root"/>-->

<!--        <property name="password"  value="123456"/>-->

<!--    </bean>-->

    <bean id="comboPooledDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

        <property name="driverClass"  value="${jdbc.driverClass}"/>

        <property name="jdbcUrl"  value="${jdbc.jdbcUrl}"/>

        <property name="user" value="${jdbc.user}"/>

        <property name="password"  value="${jdbc.password}"/>

    </bean>

    <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

        <!--mybatis 全局配置文件-->

        <property name="configLocation" value="classpath:mybatis-config.xml"/>

        <property name="dataSource" ref="comboPooledDataSource" />

        <!-- 自动扫描mapping.xml文件 -->

        <property name="mapperLocations" value="classpath:mapper/\*.xml"/>

    </bean>

    <!-- DAO接口所在包名，Spring会自动查找其下的类 -->

    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

        <property name="basePackage" value="tech.haonan.dao" />

    </bean>

    <!-- (事务管理)transaction manager, use JtaTransactionManager for global tx -->

    <bean id="transactionManager"

          class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

        <property name="dataSource" ref="comboPooledDataSource" />

    </bean>

<!--    <aop:config>-->

<!--        &lt;!&ndash;切入点表达式&ndash;&gt;-->

<!--        -->

<!--    </aop:config>-->

</beans>

1. 以下是MainHandler.java的核心代码

@Controller

public class LoginHandler {

    // set注入 这个推荐

    private UserService userService;

    @Autowired

    public void setUserService (UserService userService) {

        this.userService = userService;

    }

    // 跳转到index.jsp

    @RequestMapping("/index")

    public String index(){

        return "index";

    }

    //跳转到login.jsp

    @RequestMapping("/loginJsp")

    public String login(){

        return "login";

    }

    //跳转到register.jsp

    @RequestMapping("/registerJsp")

    public String register(){

        return "register";

    }

    // 注册接口

    @RequestMapping("/register")

    public ModelAndView register(String username,String password){

        Random r = new Random();

        Integer randomNumber;

        ModelAndView modelAndView = new ModelAndView();

        //查询是否用户名重复

        if(userService.queryDuplicateOfUsername(username) == 1){ // 说明用户名重复

            modelAndView.setViewName("registerError");

            modelAndView.addObject("username",username);

        }else{

            while (true){// 循环寻找一个不重复的数字  因为我的user对象里边不止id username password这3个属性

                randomNumber = r.nextInt(900);

                randomNumber += 100;

                if(userService.queryDuplicateOfTel("1312891" + randomNumber.toString()) != 1)  //如果发现不重复条吹u循环

                    break;

            }

            User user = new User(

                    username,

                password,

                    username+"@qq.com",

                null,

                null,

                "1312891" + randomNumber.toString(),

                null,

                "370819199303211" + randomNumber.toString(),

                true,

                (float)(randomNumber + 3000)

            );

            userService.saveUser(user);

            User sqlUser = userService.getUserByUserNameAndPasswd(username,password);

            modelAndView.addObject("user",sqlUser);

            modelAndView.setViewName("registerSuccess");

        }

        return modelAndView;

    }

    // 登录接口

    @RequestMapping("/login")

    public ModelAndView login(String username,String password){

        ModelAndView modelAndView = new ModelAndView();

        if(userService.getUserByUserNameAndPasswd(username,password) == null){  // 如果不匹配说明用户名或者密码有问题

            modelAndView.setViewName("login");

            modelAndView.addObject("error","用户名或者密码错误");

        }else{  // 如果用户名和密码正确 那么跳转到登录成功

            modelAndView.setViewName("loginSuccess");

            modelAndView.addObject("username",username);

        }

        return modelAndView;

    }

    // 返回数据库所有数据 用表格的方法列出来

    @RequestMapping("/findAll")

    public ModelAndView findAll( ){

        ModelAndView modelAndView = new ModelAndView("findAll");

        System.out.println("controller层测试成功");

        // 调用service方法

        List<User> userList = userService.findAll();

        // 把list 带到前端 然后遍历列出

        modelAndView.addObject("list",userList);

        return modelAndView;

    }

}

1. 下是UserDao.java的核心代码

@Repository

public interface UserDao {

    //查询所有账户信息

//    @Select("select \* from userTable")

    //查询所有用户

    public List<User> findAll();

    //Integer id, String username, String password, String email, Integer age, String tel, String addr, String idNumber, Integer married, Double salary

//    @Insert("insert into userTable (username,password,email,tel,idNumber,salary) values (#{username},#{password},#{email},#{tel},#{idNumber},#{salary})")

    //保存用户

    public void saveUser(User user);

//    @Select("select \* from usertable where username = #{username} and password = #{password}")

    // 通过用户名和密码寻找用户

    public User getUserByUserNameAndPasswd(@Param("username") String username, @Param("password")String password);

//    @Select("select count(\*) from userTable where username = #{username}")

    // username判断是否是重复的

    public Integer queryDuplicateOfUsername(@Param("username")String username);

//    @Select("select count(\*) from userTable where email = #{email}")

    // email 判断是否是重复的

    public Integer queryDuplicateOfEmail(@Param("email")String email);

//    @Select("select count(\*) from userTable where tel = #{tel}")

    // tel 判断是否是重复的

    public Integer queryDuplicateOfTel(@Param("tel")String tel);

//    @Select("select count(\*) from userTable where idNumber = #{idNumber}")

    // 身份证判断是否是重复的

    public Integer queryDuplicateOfIdNumber(@Param("idNumber")String idNumber);

}

1. 以下是UserMapper.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper

        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="tech.haonan.dao.UserDao"> <!--namespace是接口的全类名-->

    <select id="findAll" resultType="tech.haonan.entity.User">

        select \* from userTable

    </select>

    <insert id="saveUser" parameterType="tech.haonan.entity.User">

        insert into userTable (username,password,email,tel,idNumber,salary) values (#{username},#{password},#{email},#{tel},#{idNumber},#{salary})

    </insert>

    <select id="getUserByUserNameAndPasswd" resultType="tech.haonan.entity.User">

        select \* from usertable where username = #{username} and password = #{password}

    </select>

    <select id="queryDuplicateOfUsername" resultType="java.lang.Integer">

        select count(\*) from userTable where username = #{username}

    </select>

    <select id="queryDuplicateOfEmail" resultType="java.lang.Integer">

        select count(\*) from userTable where email = #{email}

    </select>

    <select id="queryDuplicateOfTel" resultType="java.lang.Integer">

        select count(\*) from userTable where tel = #{tel}

    </select>

    <select id="queryDuplicateOfIdNumber" resultType="java.lang.Integer">

        select count(\*) from userTable where idNumber = #{idNumber}

    </select>

</mapper>