# JavaWeb

## day25\_综合案例1

### 0.1学习目标

1.能够完成项目注册功能

2.能够完成邮箱登录功能

### 0.2 学习指南

* 注册功能的表单校验
* 注册功能的验证码校验
* 注册功能的邮箱激活
* 登录功能的实现

### 第一堂课

本节知识点

1.1项目导入

1.2项目启动

1.3技术选型

1.4创建数据库

1.5注册功能\_功能分析

本节目标

完成项目环境的搭建

#### 项目导入

##### 1.1.1知识概述

导入maven项目

##### 1.1.2视频详情



##### 1.1.3总结与补充

无

##### 1.1.4课堂提问与练习

无

##### 1.1.5习题答案

无

##### 1.1.6视频缺陷

无

##### 1.1.7视频扩展

无

#### 项目启动

##### 1.2.1知识概述

修改项目的端口号在pom.xml

<!--tomcat插件-->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<!-- tomcat7的插件， 不同tomcat版本这个也不一样 -->

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.1</version>

<configuration>

<!-- 通过maven tomcat7:run运行项目时，访问项目的端口号 -->

<port>8086</port>

<!-- 项目访问路径 本例：localhost:9090, 如果配置的aa， 则访问路径为localhost:9090/aa-->

<path>/travel</path>

</configuration>

</plugin>

</plugins>

##### 1.2.2视频详情



##### 1.2.3总结与补充

为了避免端口冲突尽量不要使用8080端口

##### 1.2.4课堂提问与练习

无

##### 1.2.5习题答案

无

##### 1.2.6视频缺陷

无

##### 1.2.7视频扩展

无

#### 技术选型

##### 1.3.1知识概述

  4.1 Web层

a) Servlet：前端控制器

b) html：视图

c) Filter：过滤器

d) BeanUtils：数据封装

e) Jackson：json序列化工具

4.2 Service层

f) Javamail：java发送邮件工具

g) Redis：nosql内存数据库

h) Jedis：java的redis客户端

4.3 Dao层

i) Mysql：数据库

j) Druid：数据库连接池

k) JdbcTemplate：jdbc的工具

##### 1.3.2视频详情



##### 1.3.3总结与补充

##### 1.3.4课堂提问与练习

无

##### 1.3.5习题答案

无

##### 1.3.6视频缺陷

无

##### 1.3.7视频扩展

无

#### 1.4创建数据库

##### 1.4.1知识概述

-- 创建数据库

CREATE DATABASE travel;

-- 使用数据库

USE travel;

--创建表

复制提供好的sql

##### 1.4.2视频详情



##### 1.4.3总结与补充

##### 1.4.4课堂提问与练习

##### 1.4.5习题与答案

##### 1.4.6视频缺陷

无

##### 1.4.7视频扩展

无

#### 1.5注册功能\_功能分析

##### 1.5.1知识概述

-- 创建数据库

CREATE DATABASE travel;

-- 使用数据库

USE travel;

--创建表

复制提供好的sql

##### 1.5.2视频详情



##### 1.5.3总结与补充

##### 1.5.4课堂提问与练习

##### 1.5.5习题与答案

##### 1.5.6视频缺陷

无

##### 1.5.7视频扩展

无

### 第二堂课

本节知识点：

2.1注册功能\_前台\_表单校验1

2.2注册功能\_前台\_表单校验2

2.3注册功能\_前台\_异步提交表单

目标：

1、能够完成表单校验

2、能够完成异步提交表单

#### 注册功能\_前台\_表单校验1

##### 2.1.1知识概述

用正则表达式校验账号密码是否符合规范

**function** *checkUsername*() {  
 //1.获取用户名值  
 **var** username = **$**(**"#username"**).val();  
 //2.定义正则  
 **var** reg\_username = /^\w{8,20}$/;  
   
 //3.判断，给出提示信息  
 **var** flag = reg\_username.test(username);  
 **if**(flag){  
 //用户名合法  
 **$**(**"#username"**).css(**"border"**,**""**);  
 }**else**{  
 //用户名非法,加一个红色边框  
 **$**(**"#username"**).css(**"border"**,**"1px solid red"**);  
 }  
   
 **return** flag;  
 }

##### 2.1.2视频详情



##### 2.1.3总结与补充

##### 2.1.4课堂提问与练习

无

##### 2.1.5习题答案

无

##### 2.1.6视频缺陷

无

##### 2.1.7视频扩展

无

#### 2.2注册功能\_前台\_表单校验2

##### 2.2.1知识概述

//校验密码  
 **function** *checkPassword*() {  
 //1.获取密码值  
 **var** password = **$**(**"#password"**).val();  
 //2.定义正则  
 **var** reg\_password = /^\w{8,20}$/;  
  
 //3.判断，给出提示信息  
 **var** flag = reg\_password.test(password);  
 **if**(flag){  
 //密码合法  
 **$**(**"#password"**).css(**"border"**,**""**);  
 }**else**{  
 //密码非法,加一个红色边框  
 **$**(**"#password"**).css(**"border"**,**"1px solid red"**);  
 }  
  
 **return** flag;  
 }  
  
 //校验邮箱  
**function** *checkEmail*(){  
 //1.获取邮箱  
 **var** email = **$**(**"#email"**).val();  
 //2.定义正则 itcast@163.com  
 **var** reg\_email = /^\w+@\w+\.\w+$/;  
  
 //3.判断  
 **var** flag = reg\_email.test(email);  
 **if**(flag){  
 **$**(**"#email"**).css(**"border"**,**""**);  
 }**else**{  
 **$**(**"#email"**).css(**"border"**,**"1px solid red"**);  
 }  
  
 **return** flag;  
}

##### 2.2.2视频详情



##### 2.2.3总结与补充

##### 2.2.4课堂提问与练习

无

##### 2.2.5习题答案

无

##### 2.2.6视频缺陷

无

##### 2.2.7视频扩展

无

#### 2.3注册功能\_前台\_异步提交表单

##### 2.3.1知识概述

**$**(**function** () {  
 //当表单提交时，调用所有的校验方法  
 **$**(**"#registerForm"**).submit(**function**(){  
  
 **return** *checkUsername*() && *checkPassword*() && *checkEmail*();  
 //如果这个方法没有返回值，或者返回为true，则表单提交，如果返回为false，则表单不提交  
 });  
 //当某一个组件失去焦点是，调用对应的校验方法  
 **$**(**"#username"**).blur(*checkUsername*);  
 **$**(**"#password"**).blur(*checkPassword*);  
 **$**(**"#email"**).blur(*checkEmail*);  
  
  
 });

##### 2.3.2视频详情



##### 2.3.3总结与补充

##### 2.3.4课堂提问与练习

无

##### 2.3.5习题答案

无

##### 2.3.6视频缺陷

无

##### 2.3.7视频扩展

无

### 第三堂课

本节知识点

3.1注册功能\_后台\_Servlet代码实现

3.2注册功能\_后台\_Dao代码实现

3.3注册功能\_后台\_service代码实现&测试

3.4注册功能\_后台\_验证码&前台数据处理

本节目标

1. 完成注册功能后台代码
2. 完成验证码的前后台处理

#### 3.1注册功能\_后台\_Servlet代码实现

##### 3.1.1知识概述

    @WebServlet("/registUserServlet")

public class RegistUserServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//验证校验

String check = request.getParameter("check");

//从sesion中获取验证码

HttpSession session = request.getSession();

String checkcode\_server = (String) session.getAttribute("CHECKCODE\_SERVER");

session.removeAttribute("CHECKCODE\_SERVER");//为了保证验证码只能使用一次

//比较

if(checkcode\_server == null || !checkcode\_server.equalsIgnoreCase(check)){

//验证码错误

ResultInfo info = new ResultInfo();

//注册失败

info.setFlag(false);

info.setErrorMsg("验证码错误");

//将info对象序列化为json

ObjectMapper mapper = new ObjectMapper();

String json = mapper.writeValueAsString(info);

response.setContentType("application/json;charset=utf-8");

response.getWriter().write(json);

return;

}

//1.获取数据

Map<String, String[]> map = request.getParameterMap();

//2.封装对象

User user = new User();

try {

BeanUtils.populate(user,map);

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (InvocationTargetException e) {

e.printStackTrace();

}

//3.调用service完成注册

UserService service = new UserServiceImpl();

boolean flag = service.regist(user);

ResultInfo info = new ResultInfo();

//4.响应结果

if(flag){

//注册成功

info.setFlag(true);

}else{

//注册失败

info.setFlag(false);

info.setErrorMsg("注册失败!");

}

//将info对象序列化为json

ObjectMapper mapper = new ObjectMapper();

String json = mapper.writeValueAsString(info);

//将json数据写回客户端

//设置content-type

response.setContentType("application/json;charset=utf-8");

response.getWriter().write(json);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

this.doPost(request, response);

}

}

##### 3.1.2视频详情



##### 3.1.3总结与补充

##### 3.1.4课堂提问与练习

##### 3.1.5习题答案

##### 3.1.6视频缺陷

无

##### 3.1.7视频扩展

无

#### 3.2注册功能\_后台\_Dao代码实现

##### 3.2.1知识概述

public class UserDaoImpl implements UserDao {

private JdbcTemplate template = new JdbcTemplate(JDBCUtils.getDataSource());

@Override

public User findByUsername(String username) {

User user = null;

try {

//1.定义sql

String sql = "select \* from tab\_user where username = ?";

//2.执行sql

user = template.queryForObject(sql, new BeanPropertyRowMapper<User>(User.class), username);

} catch (Exception e) {

}

return user;

}

@Override

public void save(User user) {

//1.定义sql

String sql = "insert into tab\_user(username,password,name,birthday,sex,telephone,email) values(?,?,?,?,?,?,?)";

//2.执行sql

template.update(sql,user.getUsername(),

user.getPassword(),

user.getName(),

user.getBirthday(),

user.getSex(),

user.getTelephone(),

user.getEmail());

}

}

##### 3.2.2视频详情



##### 3.2.3总结与补充

##### 3.2.4课堂提问与练习

无

##### 3.2.5习题答案

无

##### 3.2.6视频缺陷

无

##### 3.2.7视频扩展

无

#### 3.3注册功能\_后台\_service代码实现&测试

##### 3.3.1知识概述

public class UserServiceImpl implements UserService {

private UserDao userDao = new UserDaoImpl();

/\*\*

\* 注册用户

\* @param user

\* @return

\*/

@Override

public boolean regist(User user) {

//1.根据用户名查询用户对象

User u = userDao.findByUsername(user.getUsername());

//判断u是否为null

if(u != null){

//用户名存在，注册失败

return false;

}

//2.保存用户信息

userDao.save(user);

return true;

}

}

##### 3.3.2视频详情



##### 3.3.3总结与补充

##### 3.3.4课堂提问与练习

##### 3.3.5习题答案

无

##### 3.3.6视频缺陷

无

##### 3.3.7视频扩展

无

#### 3.4注册功能\_后台\_验证码&前台数据处理

##### 3.4.1知识概述

修改代码，增强代码的健壮性

##### 3.4.2视频详情



##### 3.4.3总结与补充

##### 3.4.4课堂提问与练习

##### 3.4.5习题答案

无

##### 3.4.6视频缺陷

无

##### 3.4.7视频扩展

无

### 第四堂课

本节知识点：

4.1注册功能\_后台\_邮件激活分析

4.2注册功能\_邮件激活\_发送邮件

4.3注册功能\_邮件激活\_点击激活分析

4.4注册功能\_邮件激活\_代码实现\_Servlet

本节目标 ：

能够完成邮件激活发送邮件

能够完成邮件发送的servlet

#### 4.1注册功能\_后台\_邮件激活分析

##### 4.1.1知识概述

发送邮件为了用户的身份验证。

##### 4.1.2视频详情



##### 4.1.3总结与补充

##### 4.1.4课堂提问与练习

无

##### 4.1.5习题答案

无

##### 4.1.6视频缺陷

无

##### 4.1.7视频扩展

无

#### 4.2注册功能\_邮件激活\_发送邮件

##### 4.2.1知识概述

public final class MailUtils {

private static final String USER = ""; // 发件人称号，同邮箱地址

private static final String PASSWORD = ""; // 如果是qq邮箱可以使户端授权码，或者登录密码

/\*\*

\*

\* @param to 收件人邮箱

\* @param text 邮件正文

\* @param title 标题

\*/

/\* 发送验证信息的邮件 \*/

public static boolean sendMail(String to, String text, String title){

try {

final Properties props = new Properties();

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.host", "smtp.qq.com");

// 发件人的账号

props.put("mail.user", USER);

//发件人的密码

props.put("mail.password", PASSWORD);

// 构建授权信息，用于进行SMTP进行身份验证

Authenticator authenticator = new Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

// 用户名、密码

String userName = props.getProperty("mail.user");

String password = props.getProperty("mail.password");

return new PasswordAuthentication(userName, password);

}

};

// 使用环境属性和授权信息，创建邮件会话

Session mailSession = Session.getInstance(props, authenticator);

// 创建邮件消息

MimeMessage message = new MimeMessage(mailSession);

// 设置发件人

String username = props.getProperty("mail.user");

InternetAddress form = new InternetAddress(username);

message.setFrom(form);

// 设置收件人

InternetAddress toAddress = new InternetAddress(to);

message.setRecipient(Message.RecipientType.TO, toAddress);

// 设置邮件标题

message.setSubject(title);

// 设置邮件的内容体

message.setContent(text, "text/html;charset=UTF-8");

// 发送邮件

Transport.send(message);

return true;

}catch (Exception e){

e.printStackTrace();

}

return false;

}

public static void main(String[] args) throws Exception { // 做测试用

MailUtils.sendMail("itcast\_xian@163.com","你好，这是一封测试邮件，无需回复。","测试邮件");

System.out.println("发送成功");

}

##### 4.2.2视频详情



##### 4.2.3总结与补充

##### 4.2.4课堂提问与练习

##### 4.2.5习题答案

##### 4.2.6视频缺陷

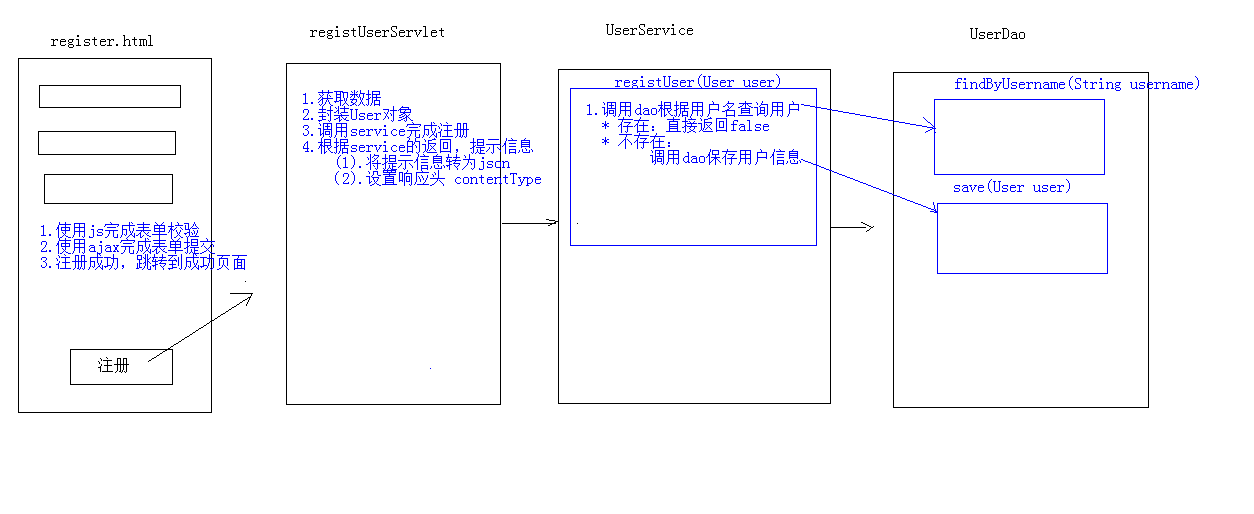
无

##### 4.2.7视频扩展

无

#### 4.3注册功能\_邮件激活\_点击激活分析

##### 4.3.1知识概述



##### 4.3.2视频详情



##### 4.3.3总结与补充

##### 4.3.4课堂提问与练习

无

。

##### 4.3.5习题答案

无

##### 4.3.6视频缺陷

无

##### 4.3.7视频扩展

无

#### 4.4注册功能\_邮件激活\_代码实现\_Servlet

##### 4.4.1知识概述

@WebServlet("/registUserServlet")

public class RegistUserServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//验证校验

String check = request.getParameter("check");

//从sesion中获取验证码

HttpSession session = request.getSession();

String checkcode\_server = (String) session.getAttribute("CHECKCODE\_SERVER");

session.removeAttribute("CHECKCODE\_SERVER");//为了保证验证码只能使用一次

//比较

if(checkcode\_server == null || !checkcode\_server.equalsIgnoreCase(check)){

//验证码错误

ResultInfo info = new ResultInfo();

//注册失败

info.setFlag(false);

info.setErrorMsg("验证码错误");

//将info对象序列化为json

ObjectMapper mapper = new ObjectMapper();

String json = mapper.writeValueAsString(info);

response.setContentType("application/json;charset=utf-8");

response.getWriter().write(json);

return;

}

//1.获取数据

Map<String, String[]> map = request.getParameterMap();

//2.封装对象

User user = new User();

try {

BeanUtils.populate(user,map);

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (InvocationTargetException e) {

e.printStackTrace();

}

//3.调用service完成注册

UserService service = new UserServiceImpl();

boolean flag = service.regist(user);

ResultInfo info = new ResultInfo();

//4.响应结果

if(flag){

//注册成功

info.setFlag(true);

}else{

//注册失败

info.setFlag(false);

info.setErrorMsg("注册失败!");

}

##### 4.4.2视频详情



##### 4.4.3总结与补充

##### 4.4.4课堂提问与练习

无

。

##### 4.4.5习题答案

无

##### 4.4.6视频缺陷

无

##### 4.4.7视频扩展

无

### 第五堂课

本节知识点：

5.1注册功能\_邮件激活\_代码实现\_Service&Dao

5.2登录功能\_分析

5.3登录功能\_实现\_Servlet

5.4登录功能\_实现\_Service&Dao

本节目标 ：

实现邮件激活代码service与dao

显示登录功能

#### 注册功能\_邮件激活\_代码实现\_Service&Dao

##### 5.1.1知识概述

public boolean regist(User user) {

//1.根据用户名查询用户对象

User u = userDao.findByUsername(user.getUsername());

//判断u是否为null

if(u != null){

//用户名存在，注册失败

return false;

}

//2.保存用户信息

//2.1设置激活码，唯一字符串

user.setCode(UuidUtil.getUuid());

//2.2设置激活状态

user.setStatus("N");

userDao.save(user);

//3.激活邮件发送，邮件正文？

String content="<a href='http://localhost/travel/user/active?code="+user.getCode()+"'>点击激活【黑马旅游网】</a>";

MailUtils.sendMail(user.getEmail(),content,"激活邮件");

return true;

}

/\*\*

\* 激活用户

\* @param code

\* @return

\*/

@Override

public boolean active(String code) {

//1.根据激活码查询用户对象

User user = userDao.findByCode(code);

if(user != null){

//2.调用dao的修改激活状态的方法

userDao.updateStatus(user);

return true;

}else{

return false;

}

}

##### 5.1.2视频详情



##### 5.1.3总结与补充

##### 5.1.4课堂提问与练习

无

##### 5.1.5习题答

无

##### 5.1.6视频缺陷

无

##### 5.1.7视频扩展

无

#### 登录功能\_分析

##### 5.2.1知识概述

验证账号密码是否存在

##### 5.2.2视频详情



##### 5.2.3总结与补充

##### 5.2.4课堂提问与练习

##### 5.2.5习题答案

##### 5.2.6视频缺陷

无

##### 5.2.7视频扩展

无

#### 5.3登录功能\_实现\_Servlet

##### 5.3.1知识概述

//1.获取用户名和密码数据

Map<String, String[]> map = request.getParameterMap();

//2.封装User对象

User user = new User();

try {

BeanUtils.populate(user,map);

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (InvocationTargetException e) {

e.printStackTrace();

}

//3.调用Service查询

UserService service = new UserServiceImpl();

User u = service.login(user);

ResultInfo info = new ResultInfo();

//4.判断用户对象是否为null

if(u == null){

//用户名密码或错误

info.setFlag(false);

info.setErrorMsg("用户名密码或错误");

}

//5.判断用户是否激活

if(u != null && !"Y".equals(u.getStatus())){

//用户尚未激活

info.setFlag(false);

info.setErrorMsg("您尚未激活，请激活");

}

//6.判断登录成功

if(u != null && "Y".equals(u.getStatus())){

//登录成功

info.setFlag(true);

}

//响应数据

ObjectMapper mapper = new ObjectMapper();

response.setContentType("application/json;charset=utf-8");

mapper.writeValue(response.getOutputStream(),info);

##### 5.3.2视频详情



##### 5.3.3总结与补充

##### 5.3.4课堂提问与练习

无

##### 5.3.5习题答案

##### 5.3.6视频缺陷

无

##### 5.3.7视频扩展

无

#### 5.4登录功能\_实现\_Service&Dao

##### 5.4.1知识概述

public User login(User user) {

return userDao.findByUsernameAndPassword(user.getUsername(),user.getPassword());

}

UserDao

public User findByUsernameAndPassword(String username, String password) {

User user = null;

try {

//1.定义sql

String sql = "select \* from tab\_user where username = ? and password = ?";

//2.执行sql

user = template.queryForObject(sql, new BeanPropertyRowMapper<User>(User.class), username,password);

} catch (Exception e) {

}

return user;

}

##### 5.4.2视频详情



##### 5.4.3总结与补充

##### 5.4.4课堂提问与练习

无

##### 5.4.5习题答案

##### 5.4.6视频缺陷

无

##### 5.3.7视频扩展

无

### 第六堂课

本节知识点：

6.1登录功能\_实现\_前台页面

6.2登录功能\_姓名提示

6.3退出功能

本节目标 ：

1. 完成登录功能前台代码
2. 完成退出功能

##### 6.1登录功能\_实现\_前台页面

##### 6.1.1知识概述

参考header.html页面

##### 6.1.2视频详情



##### 6.1.3总结与补充

##### 6.1.4课堂提问与练习

##### 6.1.5习题答案

##### 6.1.6视频缺陷

无

##### 6.1.7视频扩展

无

#### 6.2登录功能\_姓名提示

##### 6.2.1知识概述



##### 6.2.2视频详情



##### 6.2.3总结与补充

##### 6.2.4课堂提问与练习

##### 6.2.5习题答案

##### 6.2.6视频缺陷

无

##### 6.2.7视频扩展

无

#### 6.3退出功能

##### 6.3.1知识概述

//1.销毁session  
request.getSession().invalidate();  
  
//2.跳转登录页面  
response.sendRedirect(request.getContextPath()+**"/login.html"**);

##### 6.3.2视频详情



##### 6.3.3总结与补充

##### 6.3.4课堂提问与练习

##### 6.3.5习题答案

##### 6.3.6视频缺陷

无

##### 6.3.7视频扩展

无