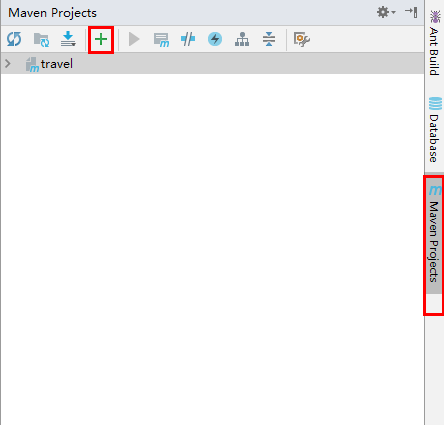
《黑马旅游网》综合案例

# 前言

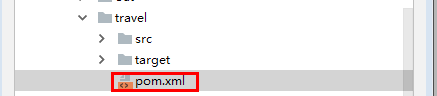
为了巩固web基础知识，提升综合运用能力，故而讲解此案例。要求，每位同学能够独立完成此案例。

# 项目导入

点击绿色＋按钮

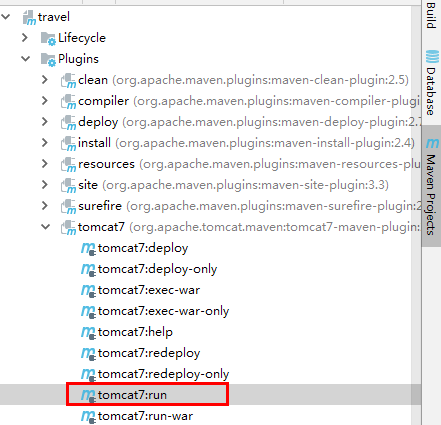


选择travel项目的pom.xml文件，点击ok，完成项目导入。需要等待一小会，项目初始化完成。

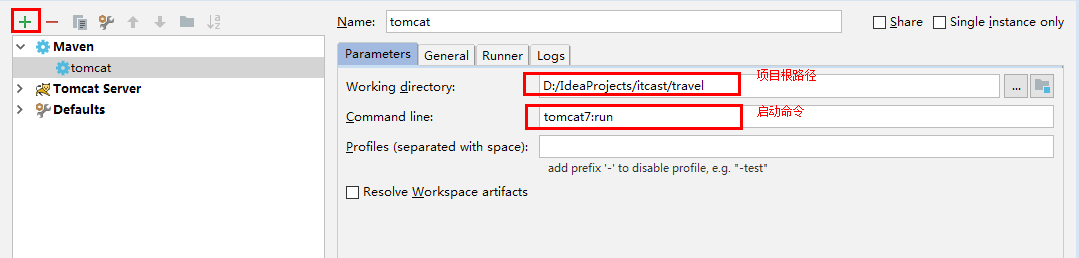


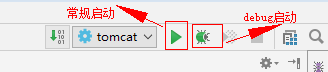
# 启动项目

## 方式一：



## 方式二：配置maven快捷启动





# 技术选型

## Web层

* 1. Servlet：前端控制器
  2. html：视图
  3. Filter：过滤器
  4. BeanUtils：数据封装
  5. Jackson：json序列化工具

## Service层

* 1. Javamail：java发送邮件工具
  2. Redis：nosql内存数据库
  3. Jedis：java的redis客户端

## Dao层

* 1. Mysql：数据库
  2. Druid：数据库连接池
  3. JdbcTemplate：jdbc的工具

# 创建数据库

-- 创建数据库

CREATE DATABASE travel;

-- 使用数据库

USE travel;

--创建表

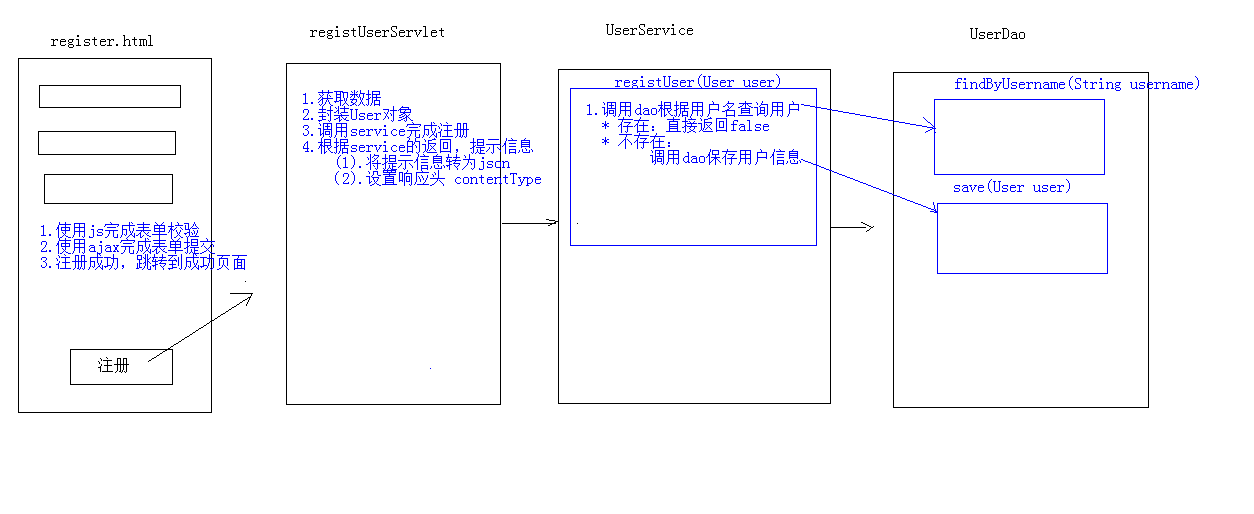
复制提供好的sql

# 注册功能

## 页面效果



## 功能分析



## 代码实现

### 前台代码实现

### 表单校验

提升用户体验，并减轻服务器压力。

//校验用户名  
//单词字符，长度8到20位  
**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;  
 }  
  
 //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;  
}  
  
**$**(**function** () {  
 //当表单提交时，调用所有的校验方法  
 **$**(**"#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*);  
  
  
 });

### 异步(ajax)提交表单

在此使用异步提交表单是为了获取服务器响应的数据。因为我们前台使用的是html作为视图层，不能够直接从servlet相关的域对象获取值，只能通过ajax获取响应数据



### 后台代码实现

### 编写RegistUserServlet

@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);  
 }  
}

### 编写UserService以及UserServiceImpl

**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**;  
 }  
}

### 编写UserDao以及UserDaoImpl

**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());  
 }  
}

### 邮件激活

为什么要进行邮件激活？为了保证用户填写的邮箱是正确的。将来可以推广一些宣传信息，到用户邮箱中。

### 发送邮件

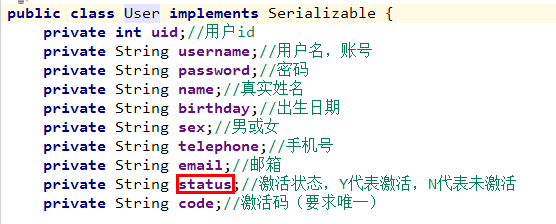
1. 申请邮箱
2. 开启授权码
3. 在MailUtils中设置自己的邮箱账号和密码(授权码)



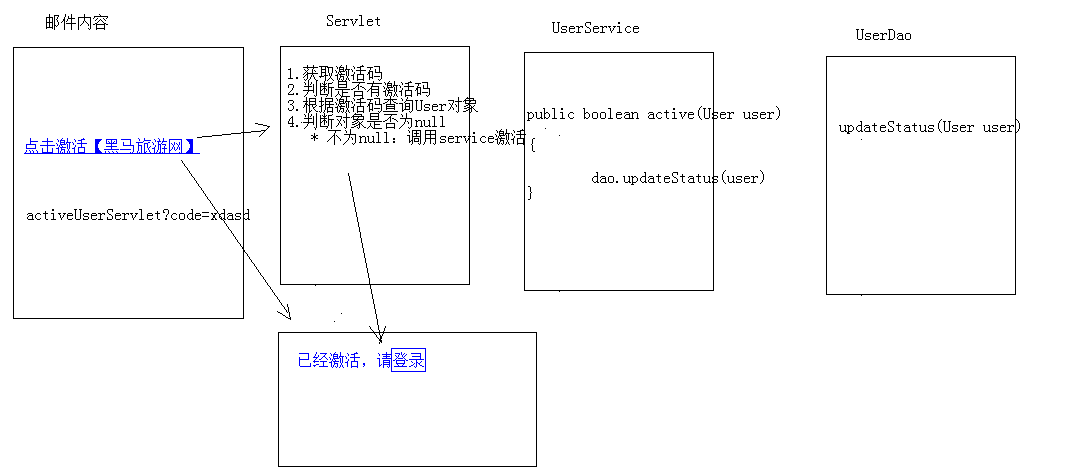
邮件工具类：MailUtils，调用其中sendMail方法可以完成邮件发送

### 用户点击邮件激活

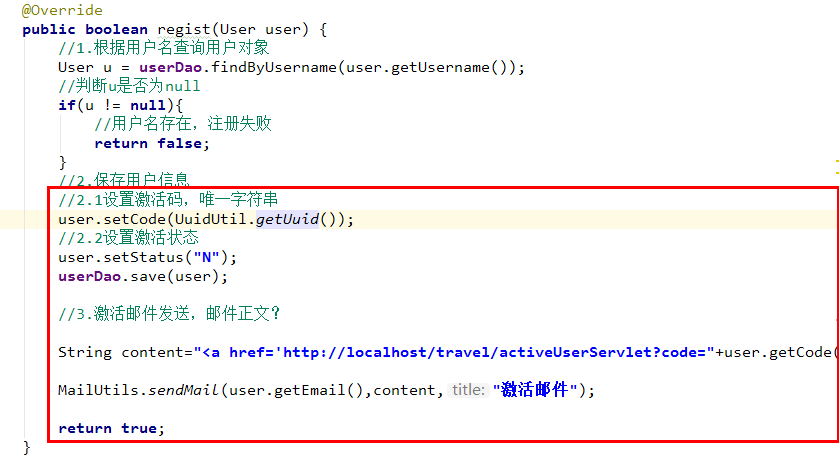
经过分析，发现，用户激活其实就是修改用户表中的status为‘Y’



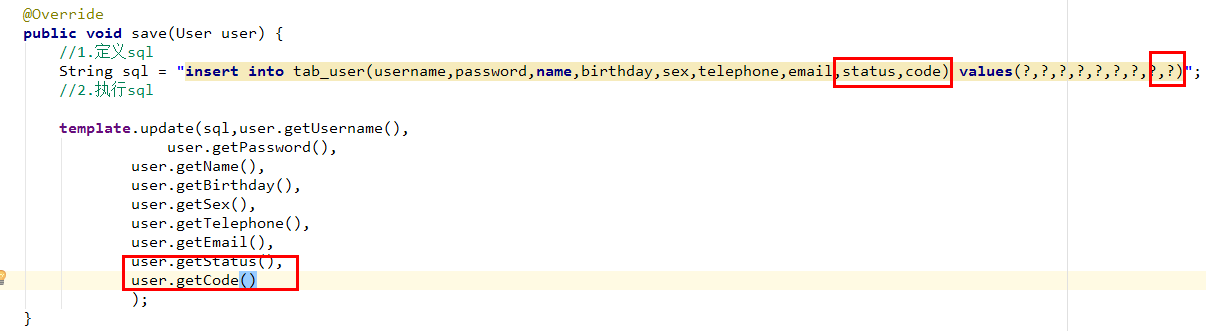
分析：



发送邮件代码：



修改保存Dao代码，加上存储status和code 的代码逻辑



激活代码实现：

ActiveUserServlet

//1.获取激活码  
String code = request.getParameter(**"code"**);  
**if**(code != **null**){  
 //2.调用service完成激活  
 UserService service = **new** UserServiceImpl();  
 **boolean** flag = service.active(code);  
  
 //3.判断标记  
 String msg = **null**;  
 **if**(flag){  
 //激活成功  
 msg = **"激活成功，请<a href='login.html'>登录</a>"**;  
 }**else**{  
 //激活失败  
 msg = **"激活失败，请联系管理员!"**;  
 }  
 response.setContentType(**"text/html;charset=utf-8"**);  
 response.getWriter().write(msg);

UserService：active

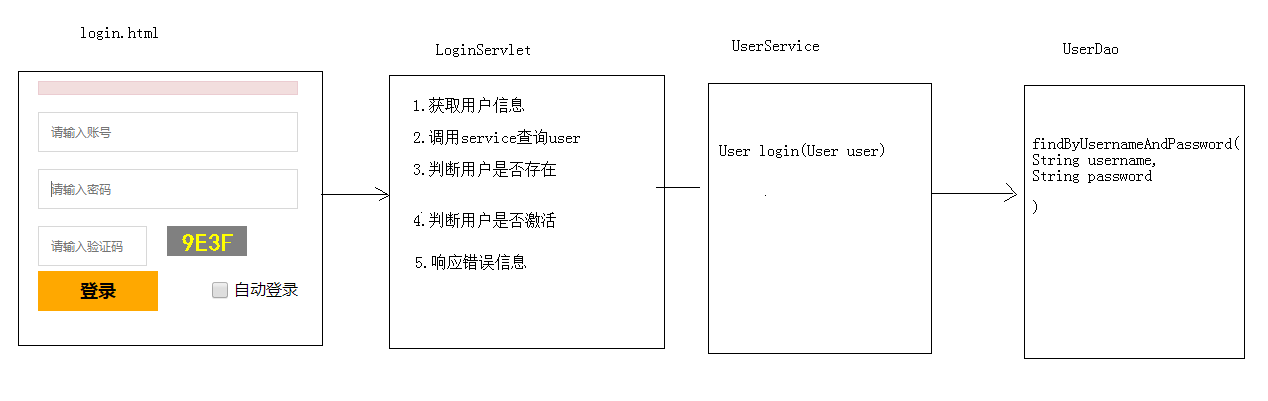
@Override  
**public boolean** active(String code) {  
 //1.根据激活码查询用户对象  
 User user = **userDao**.findByCode(code);  
 **if**(user != **null**){  
 //2.调用dao的修改激活状态的方法  
 **userDao**.updateStatus(user);  
 **return true**;  
 }**else**{  
 **return false**;  
 }  
  
}

UserDao：findByCode updateStatus

/\*\*  
 \* 根据激活码查询用户对象  
 \* **@param *code*** \* **@return** \*/  
@Override  
**public** User findByCode(String code) {  
 User user = **null**;  
 **try** {  
 String sql = **"select** *\** **from tab\_user where code = ?"**;  
  
 user = **template**.queryForObject(sql,**new** BeanPropertyRowMapper<User>(User.**class**),code);  
 } **catch** (DataAccessException e) {  
 e.printStackTrace();  
 }  
  
 **return** user;  
}  
  
/\*\*  
 \* 修改指定用户激活状态  
 \* **@param *user*** \*/  
@Override  
**public void** updateStatus(User user) {  
 String sql = **" update tab\_user set status = 'Y' where uid=?"**;  
 **template**.update(sql,user.getUid());  
}

# 登录

## 分析



## 代码实现

### 前台代码



### 后台代码

LoginServlet

//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);

UserService

**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;  
}

### index页面中用户姓名的提示信息功能

效果：



header.html代码



Servlet代码

//从session中获取登录用户  
Object user = request.getSession().getAttribute(**"user"**);  
//将user写回客户端  
  
ObjectMapper mapper = **new** ObjectMapper();  
response.setContentType(**"application/json;charset=utf-8"**);  
mapper.writeValue(response.getOutputStream(),user);

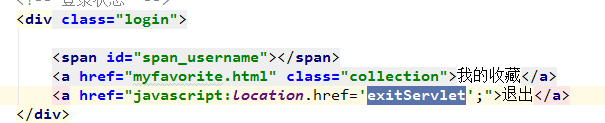
# 退出

什么叫做登录了？session中有user对象。

实现步骤：

* + 1. 访问servlet，将session销毁
    2. 跳转到登录页面

代码实现：

Header.html

Servlet:

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