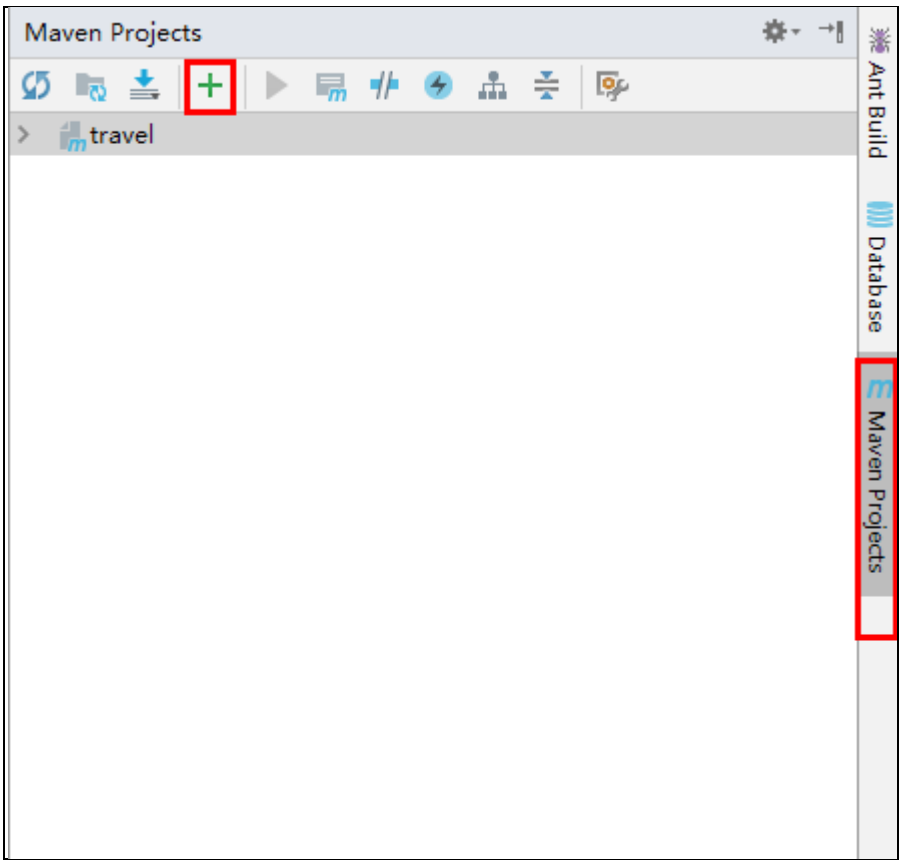
《黑马旅游网》综合案例

1 前言

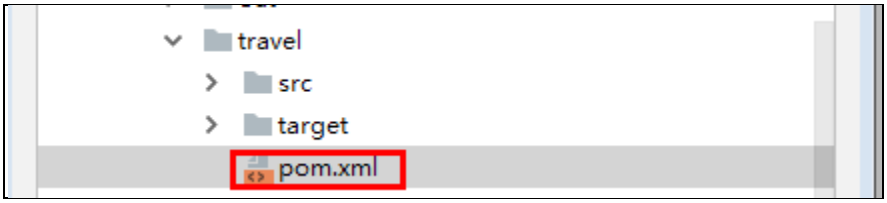
为了巩固 web 基础知识，提升综合运用能力，故而讲解此案例。要求，每位同学能够独立完成此案例。

2 项目导入

点击绿色＋按钮

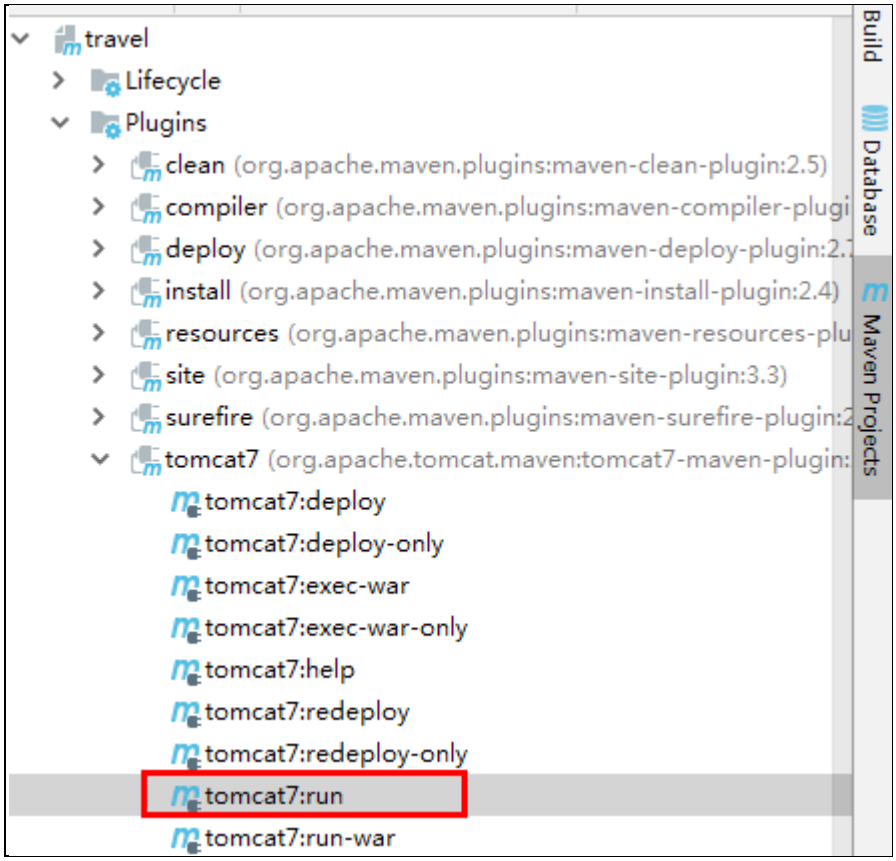


选择 travel 项目的 pom.xml 文件，点击 ok，完成项目导入。需要等待一小会，项目初始化完成。

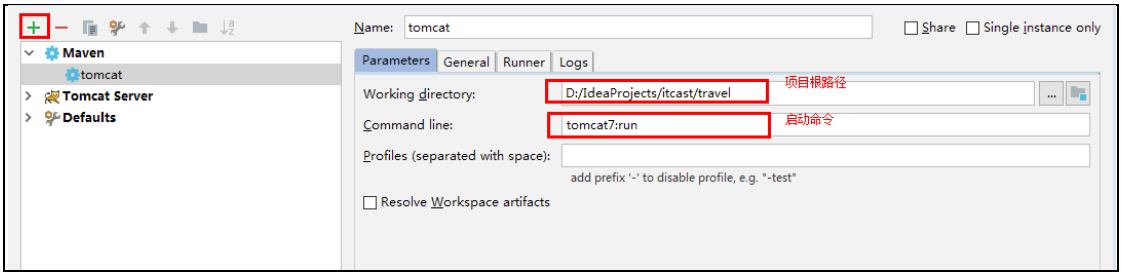


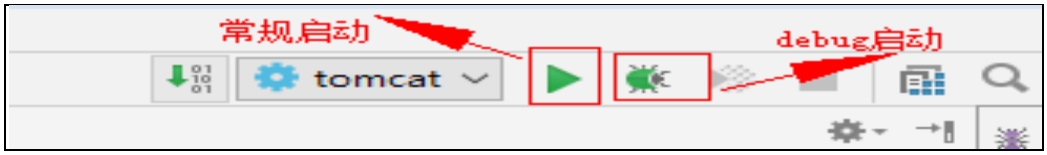
3 启动项目

3.1 方式一：



3.2 方式二：配置 maven 快捷启动





4 技术选型

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

5 创建数据库

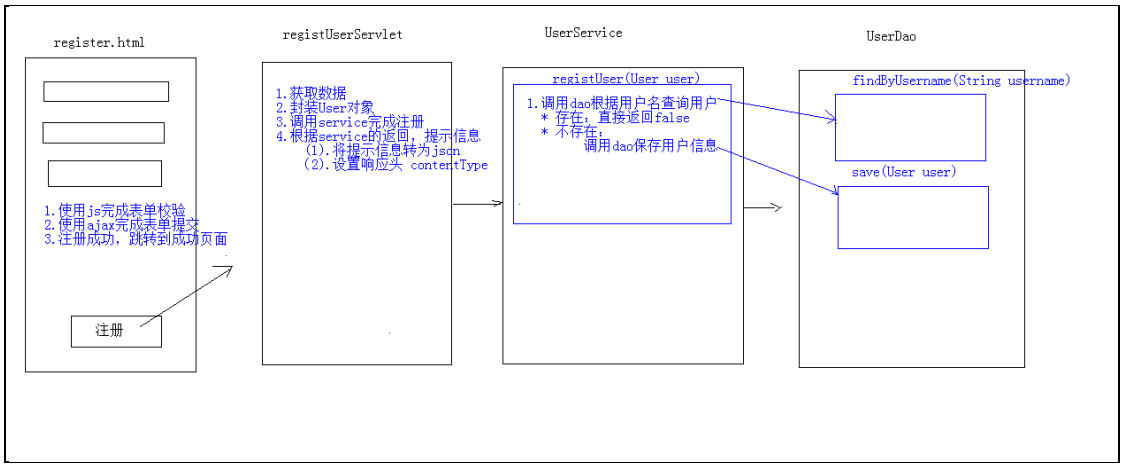
```
-- 创建数据库
CREATE DATABASE travel;
-- 使用数据库
USE travel;
--创建表
```

复制提供好的 sql

6 注册功能

6.1 页面效果

6.2 功能分析



6.3 代码实现

6.3.1 前台代码实现

6.3.2 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);


        });
```

6.3.3 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

```
//当表单提交时，调用所有的校验方法
$("#registerForm").submit(function(){
    //1.发送数据到服务器
    if(checkUsername() && checkPassword() && checkEmail()){
        //校验通过,发送ajax请求，提交表单的数据   username=zhangsan&password=123

        $.post("registUserServlet",$(this).serialize(),function(data){
            //处理服务器响应的数据   data
        });

    }
    //2.不让页面跳转
    return false;
    //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
});
```

6.3.4 后台代码实现

6.3.5 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);
    }
}
```

6.3.6 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;
```

```
    }
}
```

6.3.7 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());
```

```
    }
}
```

6.3.8 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。

6.3.9 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)

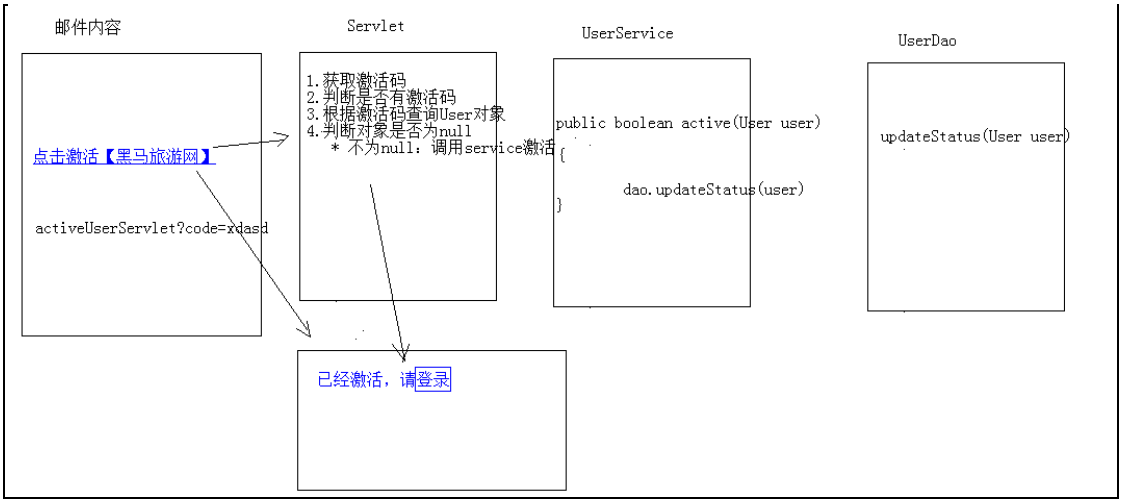
邮件工具类：MailUtils，调用其中 `sendMail` 方法可以完成邮件发送

6.3.10 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’

```java
public class User implements Serializable {
    private int uid;//用户id
    private String username;//用户名，账号
    private String password;//密码
    private String name;//真实姓名
    private String birthday;//出生日期
    private String sex;//男或女
    private String telephone;//手机号
    private String email;//邮箱
    private String status;//激活状态，Y代表激活，N代表未激活
    private String code;//激活码（要求唯一）
```

分析：



发送邮件代码：

```java
@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
    //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/activeUserServlet?code="+user.getCode(

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

    return true;
}
```

修改保存 Dao 代码，加上存储 status 和 code 的代码逻辑

```java
@Override
public void save(User user) {
    //1.定义sql
    String sql = "insert into tab_user(username,password,name,birthday,sex,telephone,email,status,code) values(?,?,?,?,?,?,?,?,?)";
    //2.执行sql

    template.update(sql,user.getUsername(),
                user.getPassword(),
            user.getName(),
            user.getBirthday(),
            user.getSex(),
            user.getTelephone(),
            user.getEmail(),
            user.getStatus(),
            user.getCode()
            );
}
```

激活代码实现：

ActiveUserServlet

```java
//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

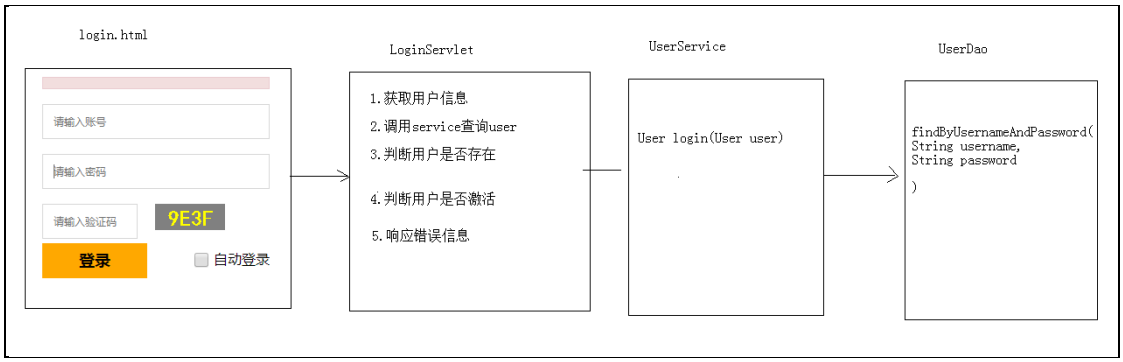
```java
@Override
public boolean active(String code) {
    //1.根据激活码查询用户对象
    User user = userDao.findByCode(code);
    if(user != null){
        //2.调用 dao 的修改激活状态的方法
        userDao.updateStatus(user);
        return true;
    }else{
        return false;
    }

}
```

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());
}
```

7 登录

7.1 分析



7.2 代码实现

7.2.1 前台代码

```
$(function () {
    //1.给登录按钮绑定单击事件
    $("#btn_sub").click(function () {
        //2.发送ajax请求，提交表单数据
        $.post("loginServlet",$("#loginForm").serialize(),function (data) {
            //data : {flag:false,errorMsg:''}
            if(data.flag){
                //登录成功
                location.href="index.html";
            }else{
                //登录失败
                $("#errorMsg").html(data.errorMsg);
            }
        });
    });
});
```

7.2.2 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.getP
assword());
}
```

UserDao

```java
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;
}
```

7.2.3 index 页面中用户姓名的提示信息功能

效果：

header.html 代码

```javascript
$(function () {
    $.get("findUserServlet",{},function (data) {
        //{uid:1,name:'李四'}
        var msg = "欢迎回来，"+data.name;
        $("#span_username").html(msg);

    });
})
```

Servlet 代码

```java
//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);
```

8 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

```
<div class="login">

    <span id="span_username"></span>
    <a href="myfavorite.html" class="collection">我的收藏</a>
    <a href="javascript:location.href='exitServlet';">退出</a>
</div>
```

Servlet:

```
//1.销毁 session
request.getSession().invalidate();

//2.跳转登录页面
response.sendRedirect(request.getContextPath()+"/login.html");
```