day41-综合实战第一天

学习目标

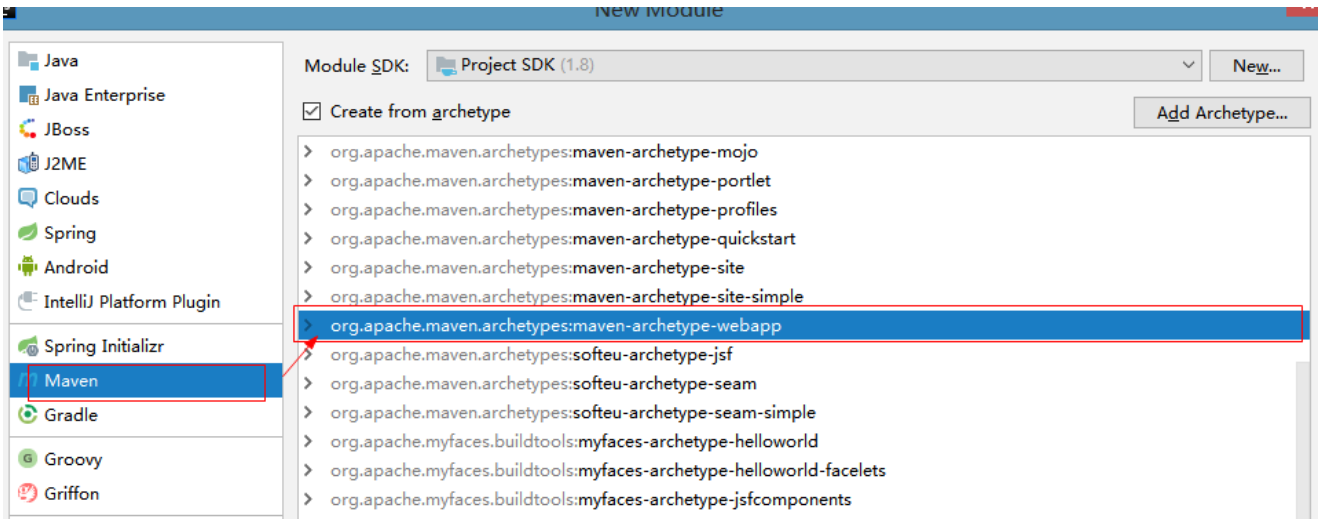
1. 能够完成用户注册案例
2. 能够完成用户登录与退出案例
3. 可以实现BaseServlet优化请求处理

项目的环境搭建

一,项目环境的搭建

1.创建Maven项目

- 创建Maven项目



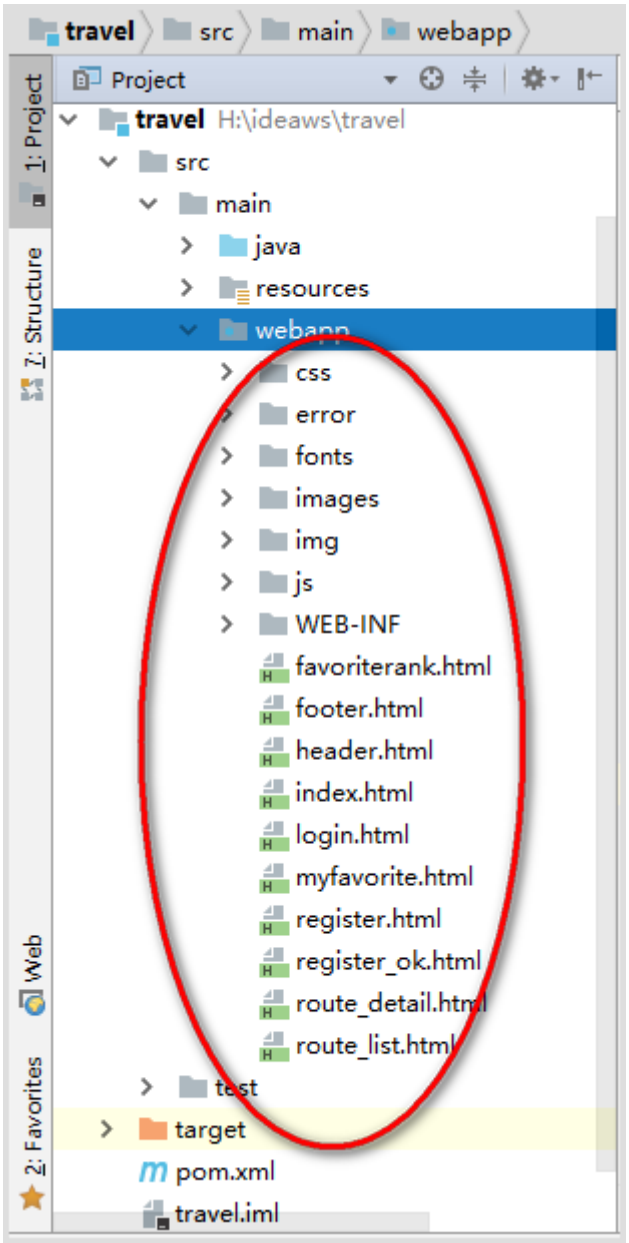
2.添加坐标依赖

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
    <!--servlet-->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
        <scope>provided</scope>
    </dependency>
    <!--mysql驱动-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.26</version>
        <scope>compile</scope>
    </dependency>
    <!--c3p0连接池-->
    <dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1.2</version>
    </dependency>
    <!--jdbcTemplate-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>4.1.2.RELEASE</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>4.1.2.RELEASE</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>4.1.2.RELEASE</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>4.1.2.RELEASE</version>
        <scope>compile</scope>
    </dependency>

    <dependency>
```

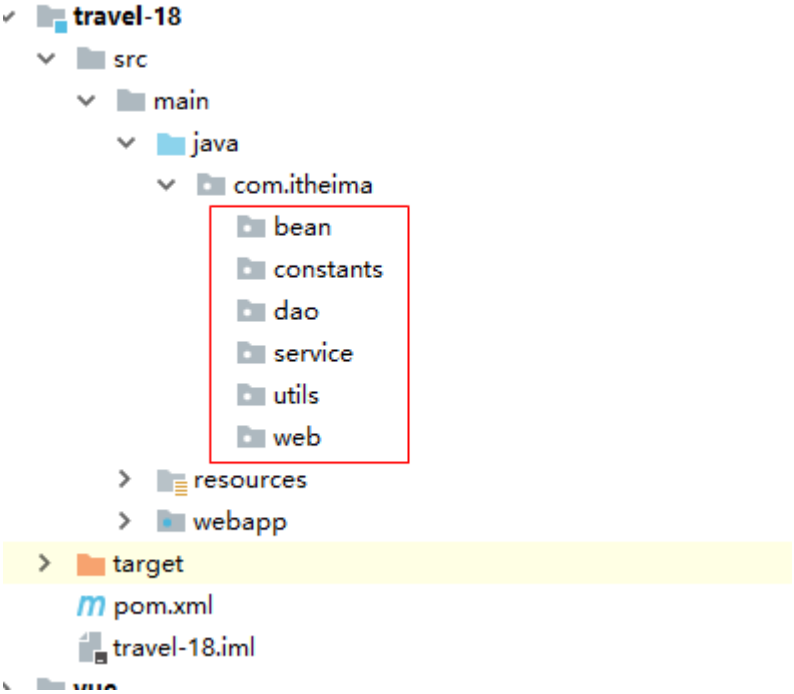
```
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.1.1</version>
            <scope>compile</scope>
        </dependency>
        <!--beanUtils-->
        <dependency>
            <groupId>commons-beanutils</groupId>
            <artifactId>commons-beanutils</artifactId>
            <version>1.9.2</version>
            <scope>compile</scope>
        </dependency>
        <!--jackson-->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.3.3</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.3.3</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-annotations</artifactId>
            <version>2.3.3</version>
        </dependency>

        <!--javaMail-->
        <dependency>
            <groupId>javax.mail</groupId>
            <artifactId>javax.mail-api</artifactId>
            <version>1.5.6</version>
        </dependency>
        <dependency>
            <groupId>com.sun.mail</groupId>
            <artifactId>javax.mail</artifactId>
            <version>1.5.3</version>
        </dependency>
        <!--jedis-->
        <dependency>
            <groupId>redis.clients</groupId>
            <artifactId>jedis</artifactId>
            <version>2.7.0</version>
        </dependency>
    </dependencies>
```

3.导入页面

将“资料/01-静态页面”导入到项目的webapp目录下，如图

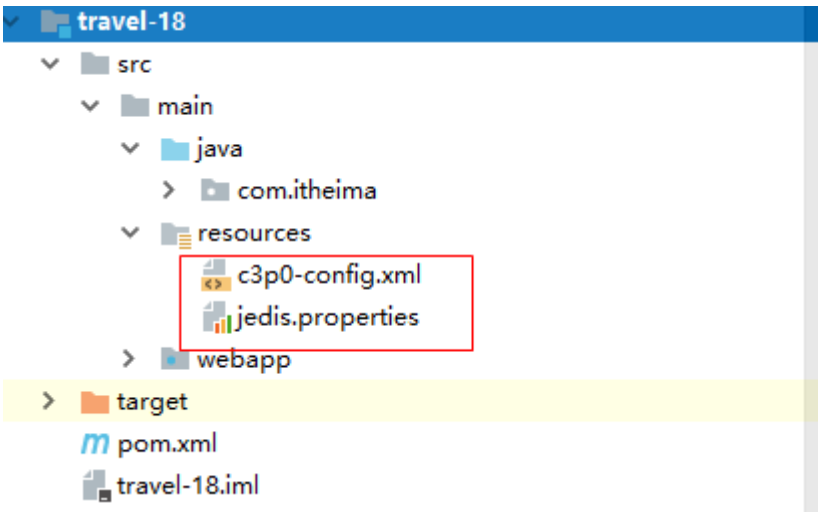

travel
src
main
webapp
Project
travel H:\ideaws\travel
src
main
java
resources
webapp
css
error
fonts
images
img
js
WEB-INF
favoriterank.html
footer.html
header.html
index.html
login.html
myfavorite.html
register.html
register_ok.html
route_detail.html
route_list.html
test
target
pom.xml
travel.iml
1: Project
7: Structure
Web
2: Favorites

4.创建包结构



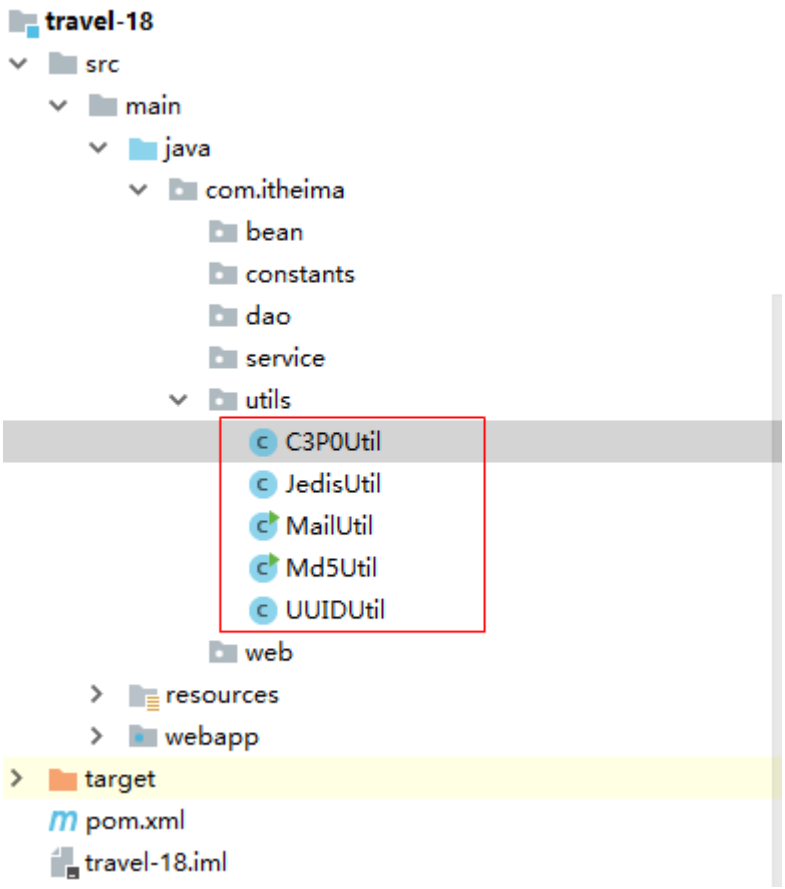
5. 导入配置文件

- 将“资料/02-配置文件”导入到resource资源目录中



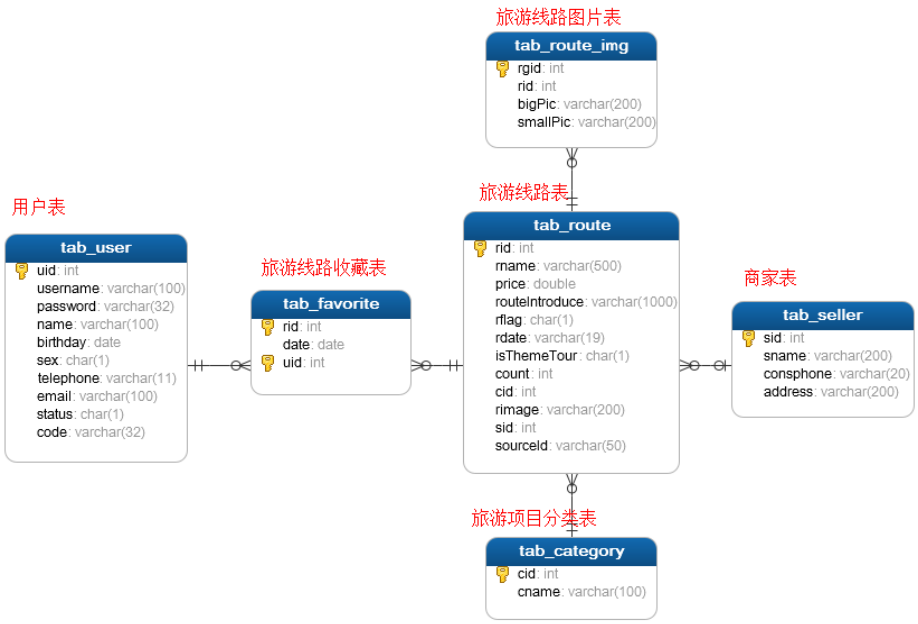
6. 导入工具类

- 将"资料/03-工具类"导入utils包下



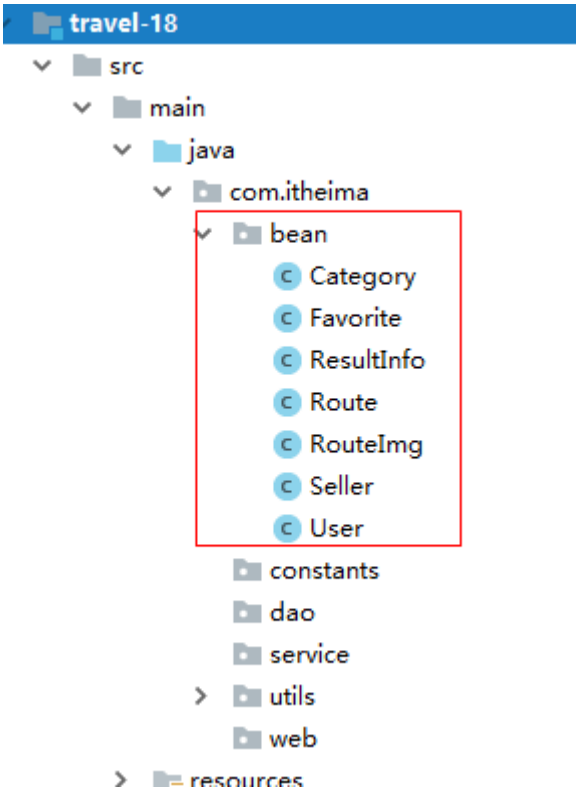
7 导入数据库脚本

到mysql数据库执行“资料/04-数据库脚本”，表之间的关系如下图



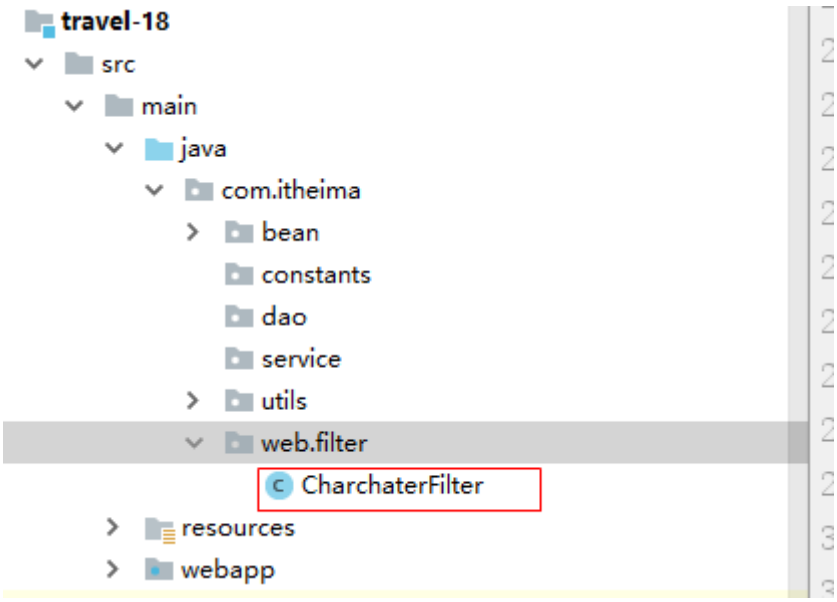
8 导入/创建实体

将“资料/05-实体类”打入到model包中



9 导入/创建其它公共类

将“资料/06-其他常用类”导入到如下包中



二,BaseServlet的抽取【重点】

1.BaseServlet的分析

传统方式的开发一个请求对应一个Servlet:这样的话会导致一个模块的Servlet过多,导致整个项目的Servlet都会很多.能不能做一个处理?让一个模块都用一个Servlet处理请求. 用户模块, 创建userServlet

注册:http://localhost:8080/day31/userServlet?method=regist

登录:http://localhost:8080/day31/userServlet?method=login

激活:http://localhost:8080/day31/userServlet?method=active

- 以"模块为单位"创建Servlet的方式

```
class UserServlet extend HttpServlet{

    ... doGet(HttpServletRequest request, HttpServletResponse response){
        //1. 获得method请求参数的值
        String methodStr =  request.getParameter("method");
        //2. 判断对应的是哪一种请求(注册, 登录还是其它)
        if("regist".equal(methodStr)){
            //注册
            regist(request,response);
        }else if("login".equal(methodStr)){
            //登录
            login(request,response);
        }else if("active".equal(methodStr)){
            //激活
            active(request,response);
        }
        .....

    }

    public void regist(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }

     public void login(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }
    public void active(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }


}
```

发现在上面的doGet方法里面,有大量的if语句,能不能不写if语句

注册:http://localhost:8080/day31/userServlet?method=regist

登录:http://localhost:8080/day31/userServlet?method=login

激活:http://localhost:8080/day31/userServlet?method=active

- 优化后

```
class UserServlet extend HttpServlet{
    //1. 判断对应的是哪一个方法  2. 执行该方法
    ... doGet(HttpServletRequest request, HttpServletResponse response){
        //1. 获得method请求参数的值(方法名)  eg: regist
        String methodStr =  request.getParameter("method");
        //2. 获得字节码
        Class clazz = this.getClass();
        //3. 根据方法名 反射获得对应的method方法对象
        Method method =
clazz.getMethod(methodStr,HttpServletRequest.class,HttpServletResponse.class);
        //4. 让该方法执行
        method.invoke(this,request,response);


    }

    public void regist(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }

     public void login(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }
    public void active(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }


}
------------------------------
 商品模块 ProductServlet
    添加商品:http://localhost:8080/day31/productServlet?method=addProduct
    展示所有的商品:http://localhost:8080/day31/productServlet?method=showAll
    删除商品:http://localhost:8080/day31/productServlet?method=delete

class ProductServlet extend HttpServlet{
    //1. 判断对应的是哪一个方法  2. 执行该方法
    ... doGet(HttpServletRequest request, HttpServletResponse response){
        //1. 获得method请求参数的值(方法名)
        String methodStr =  request.getParameter("method");
        //2. 获得字节码
        Class clazz = this.getClass();
        //3. 根据方法名 反射获得对应的method方法对象
        Method method =
clazz.getMethod(methodStr,HttpServletRequest.class,HttpServletResponse.class);
        //4. 让该方法执行

        method.invoke(this,request,response);
```

```
    }

    public void addProduct(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }

     public void showAll(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }
    public void delete(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }


}

------------------------------
 订单模块 OrderServlet
    生成订单:http://localhost:8080/day31/orderServlet?method=saveOrder
    展示所有的订单:http://localhost:8080/day31/orderServlet?method=findAll


class OrderServlet extend HttpServlet{
    //1. 判断对应的是哪一个方法  2. 执行该方法
    ... doGet(HttpServletRequest request, HttpServletResponse response){
        //1. 获得method请求参数的值(方法名)
        String methodStr =  request.getParameter("method");
        //2. 获得字节码
        Class clazz = this.getClass();
        //3. 根据方法名 反射获得对应的method方法对象
        Method method =
clazz.getMethod(methodStr,HttpServletRequest.class,HttpServletResponse.class);
        //4. 让该方法执行
        method.invoke(this,request,response);


    }

    public void saveOrder(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }


     public void findAll(HttpServletRequest request, HttpServletResponse response){
```

```
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }



}

------------------------------
 商品模块 ProductServlet
    添加商品:http://localhost:8080/day31/productServlet?method=addProduct
    展示所有的商品:http://localhost:8080/day31/productServlet?method=showAll
    删除商品:http://localhost:8080/day31/productServlet?method=delete

class ProductServlet extend BaseServlet{

    public void addProduct(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }

     public void showAll(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }
    public void delete(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }


}
```

- 每一个模块对应一个Servlet,发现doGet()方法里面,代码都是重复的,所以抽取一个通用的BaseServlet基类, 让各个模块Servlet继承BaseServlet.通用的BaseServlet　好处: 少些代码, 把公共的代码抽取

```
class BaseServlet extend HttpServlet{
        //1. 判断对应的是哪一个方法  2. 执行该方法
    ... service(HttpServletRequest request, HttpServletResponse response){
        //1. 获得method请求参数的值(方法名)  eg: regist
        String methodStr =  request.getParameter("method");
        //2. 获得字节码
        Class clazz = this.getClass();
        //3. 根据方法名 反射获得对应的method方法对象
        Method method =
clazz.getMethod(methodStr,HttpServletRequest.class,HttpServletResponse.class);
        //4. 让该方法执行
        method.invoke(this,request,response);

    }
}

eg: 注册:http://localhost:8080/day31/userServlet?method=regist
    this是BaseServlet还是UserServlet? 是UserServlet. 对象是谁,this就是谁; 请求的是userServlet,服务
器给我们创建的就是UserServlet;

class UserServlet extend BaseServlet{



    public void regist(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }

     public void login(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }
    public void active(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }


}

------------------------------
 订单模块 OrderServlet
    生成订单:http://localhost:8080/day31/orderServlet?method=saveOrder
    展示所有的订单:http://localhost:8080/day31/orderServlet?method=findAll


class OrderServlet extend BaseServlet{
```

```
    public void saveOrder(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }

     public void findAll(HttpServletRequest request, HttpServletResponse response){
        //1. 接受请求参数
        //2. 调用业务
        //3. 分发转向
    }



}
```

2.BaseServlet的编写

```
@WebServlet("/base")
public class BaseServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        try {
            //1. 获得method请求参数的值(方法名)
            String methodStr = req.getParameter("method");
            //2. 获得字节码对象
            Class clazz = this.getClass();
            //3. 根据方法名, 反射得到当前方法对象
            Method method = clazz.getMethod(methodStr, HttpServletRequest.class,
HttpServletResponse.class);
            //4. 执行
            method.invoke(this,req,resp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

案例一-用户注册

一,案例需求

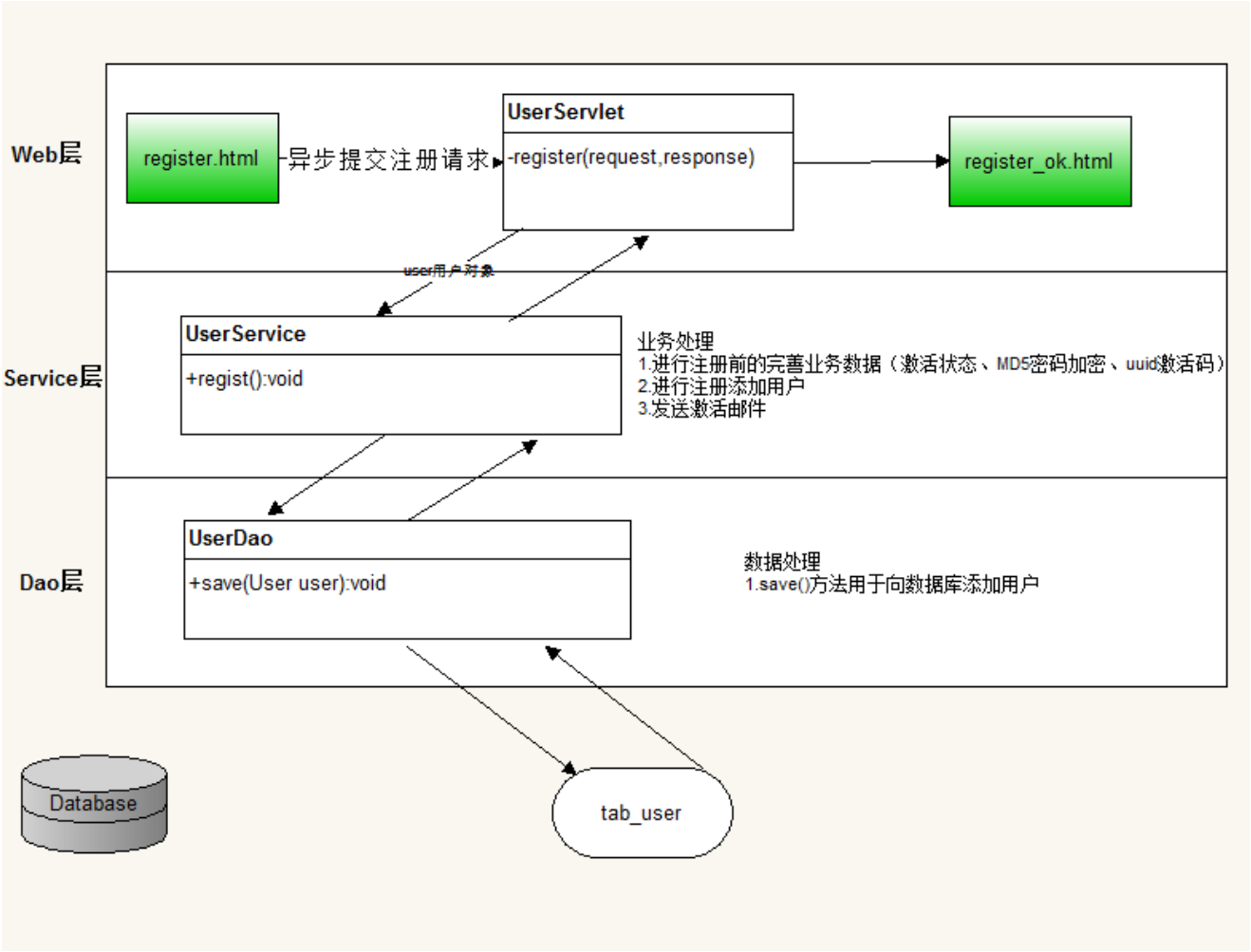
实现用户注册，要求前端发送异步请求注册。需要发送邮件激活码.

用户注册成功后，跳转到注册成功页面, 提示用户登录邮箱去激活;

如果注册失败,在当前页面提示用户注册失败


域名重定向 http://localhost:8080/travel/register.html
注册页面
新用户注册
USER REGISTER
已有账号？ 立即登录
用户名 zhangsan
密码 123456
Email zhangsan@itheima.com
姓名 张三
手机号 12345678901
性别 男 女
出生日期 2018/02/13
验证码 g3DC
g3DC
注册
注册成功，跳转到注册成功页面
注册失败，在当前页面显示错误消息
域名重定向 http://localhost:8080/travel/register_ok.html
注册成功页面
酒店+门票双人套餐
旅游
快乐每一程
黑马
亲子周边旅游节
黑马程序员™
www.itheima.com
请输入路线名称
搜索
首页 门票 酒店 香港车票 出境游 国内游 港澳游 抱团定制 全球自由行
恭喜，注册成功！请登录您的注册邮箱进行激活您的账号，激活后
注册成功的显示消息
产品齐全
产品全自主选，随心买
便利快捷
24小时不打烊，随时买
安全支付
知名支付工具，放心买
贴心服
客服全年

二,案例思路



三,代码实现

- register.jsp

```
    <script>
        //给表单设置提事件
        $("#registerForm").submit(function () {
            //序列化表单提交的数据
            var data =  $(this).serialize();
            alert("data="+data);
            //发送Ajax请求服务器
            $.post("user",data,function (result) {
                if (result.flag){
                    //注册成功
                    location.href = "register_ok.html";
                }else{
                   //注册失败
                    alert(result.msg);
                }

            },"json");
            //阻止表单同步提交
            return false;
        });

    </script>
```

- UserServlet

```
@WebServlet("/user")
public class UserServlet extends  BaseServlet {

    /**
     * 用户注册
     * @param request
     * @param response
     */
    public void register(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String data = null;
        ResultInfo resultInfo = null;
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            //1. 获得请求参数
            Map<String, String[]> map = request.getParameterMap();
            User user = new User();
            BeanUtils.populate(user,map);
            //2. 调用业务
            UserService userService = new UserService();
            userService.regist(user);

            //3. 响应数据
            resultInfo = new ResultInfo(true,null,"注册成功!");
        } catch (Exception e) {
            e.printStackTrace();
            resultInfo = new ResultInfo(false,null,"注册失败!");
        }finally {
            data = objectMapper.writeValueAsString(resultInfo);
            response.getWriter().print(data);
        }
    }

}
```

- UserService.java

```
public class UserService {

    private UserDao userDao =   new UserDao();
    /**
     * 用户注册
     * @param user
     */
    public void regist(User user) throws Exception {
        user.setStatus("N");
        user.setCode(UUIDUtil.getUuid());
        userDao.save(user);
        MailUtil.sendMail(user.getEmail(),"尊敬的:"+user.getName()+"欢迎注册黑马旅游网!请点击如下超链接进行激活!<a href='http://localhost:8080/travel-18/user?method=active&code="+user.getCode()+"'>点击激活</a>");

    }
}
```

- UserDao

```
public class UserDao {

    private JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Util.getDataSource());

    /**
     * 保存用户
     * @param user
     */
    public void save(User user) {
        //定义插入用户sql语句
        String sql = "INSERT INTO tab_user VALUES(NULL,?,?,?,?,?,?,?,?,?,?)";
        //执行sql语句,返回影响行数
        jdbcTemplate.update(sql,
                user.getUsername(),
                user.getPassword(),
                user.getName(),
                user.getBirthday(),
                user.getSex(),
                user.getTelephone(),
                user.getEmail(),
                user.getStatus(),
                user.getCode()
        );
    }
}
```

四,案例扩展

1.使用MD5对密码进行加密

我们在实际开发里面,为了保证用户密码的保密性, 基本上都会先加密,再存到数据库里面. 用的比较多的就是MD5加密. 在业务层使用MD5对密码进行加密

```
public class UserService {

    private UserDao userDao =   new UserDao();
    /**
     * 用户注册
     * @param user
     */
    public void regist(User user) throws Exception {
        user.setStatus("N");
        user.setCode(UUIDUtil.getUuid());
        user.setPassword(Md5Util.encodeByMd5(user.getPassword()));//进行加密
        userDao.save(user);
        MailUtil.sendMail(user.getEmail(),"尊敬的:"+user.getName()+"欢迎注册黑马旅游网!请点击如下超链接进行激活!<a href='http://localhost:8080/travel-18/user?method=active&code="+user.getCode()+"'>点击激活</a>");

    }
}
```

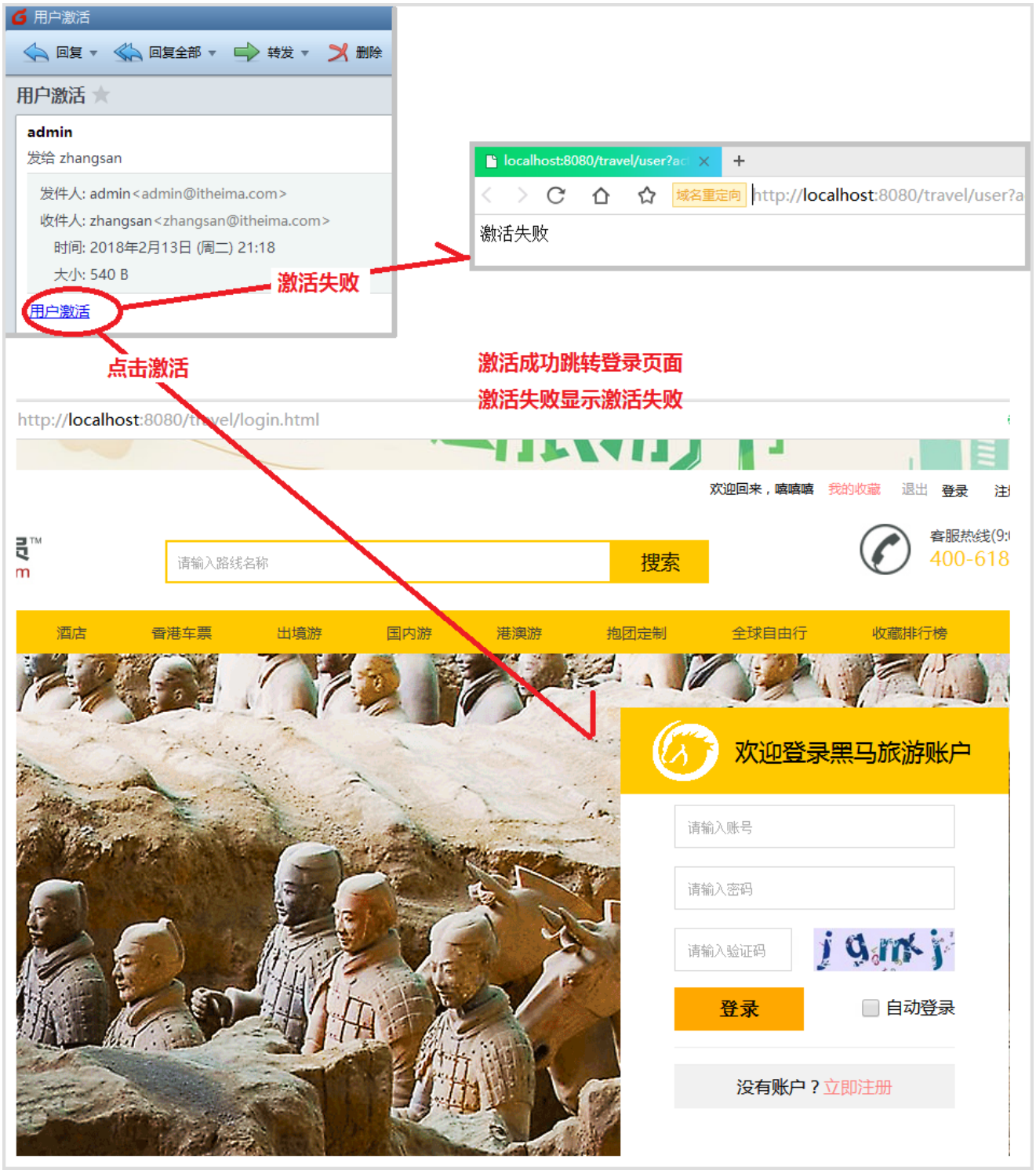
案例二-用户激活

一,案例需求

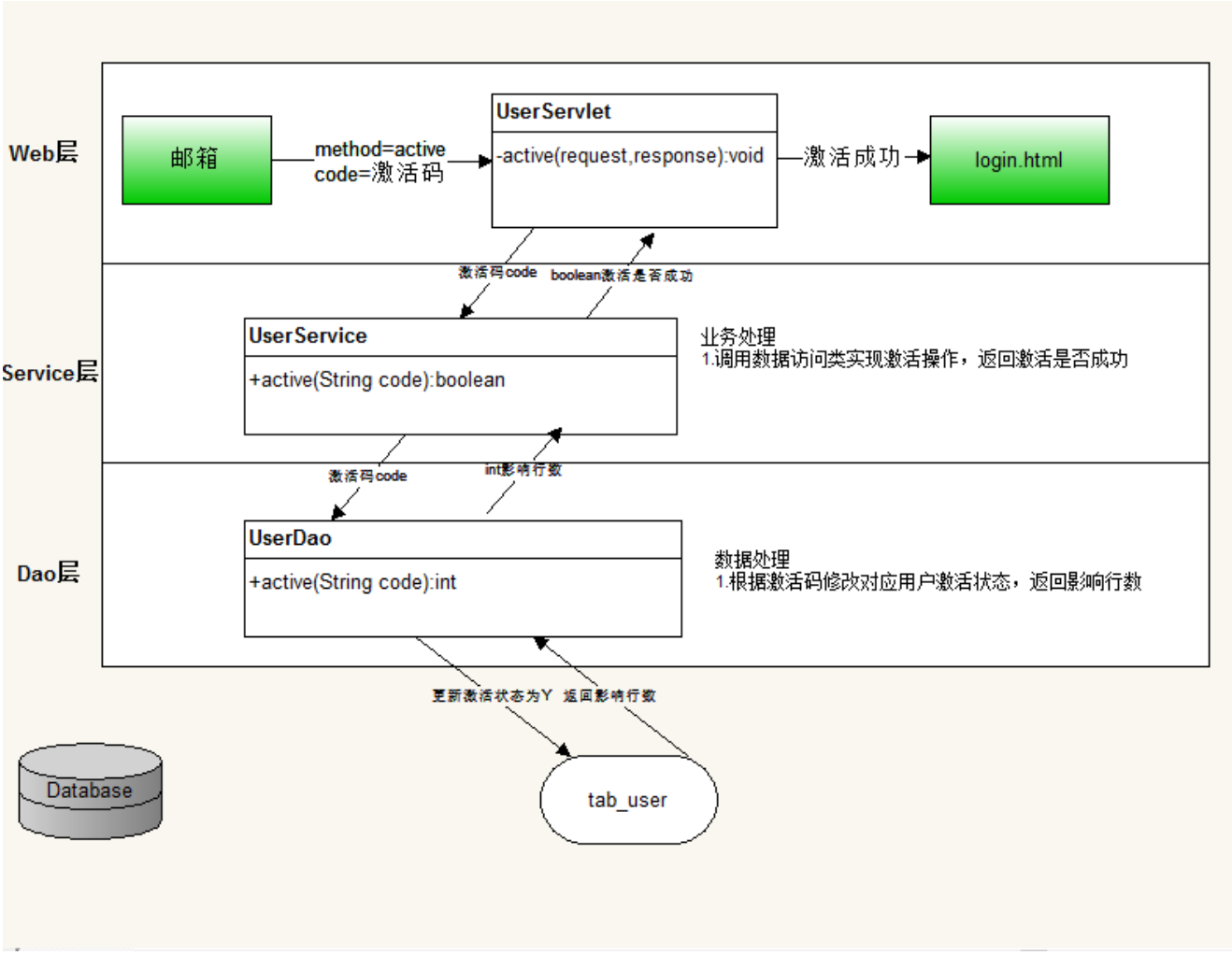
用户登录邮箱, 点击激活超链接,

如果激活成功,重定向到登录页面;

如果激活失败,给用户提示激活失败


用户激活
回复
回复全部
转发
删除
用户激活
admin
发给 zhangsan
发件人: admin<admin@itheima.com>
收件人: zhangsan<zhangsan@itheima.com>
时间: 2018年2月13日 (周二) 21:18
大小: 540 B
用户激活
激活失败
localhost:8080/travel/user?a
域名重定向
http://localhost:8080/travel/user?a
激活失败
点击激活
激活成功跳转登录页面
激活失败显示激活失败
http://localhost:8080/travel/login.html
欢迎回来，嘻嘻嘻
我的收藏
退出
登录
请输入路线名称
搜索
客服热线(9:
400-618
酒店
香港车票
出境游
国内游
港澳游
抱团定制
全球自由行
收藏排行榜
欢迎登录黑马旅游账户
请输入账号
请输入密码
请输入验证码
登录
自动登录
没有账户？立即注册

二,案例思路



- 修改发送邮件的超链接路径

```
 MailUtil.sendMail(user.getEmail(),"<a href='http://localhost:8080/travel-43/user?
methd=active&code="+user.getCode()+"'>用户激活</a>");
```

- 在UserServlet里面创建active

```
public void active(request,response){
        //1. 获得激活码
        //2.调用业务，判断是的激活成功
        //3.给前端响应

}
```

- 在UserService里面

```
public boolean active(String code){
        //调用Dao，进行激活的处理
        //判断是否激活成功，返回boolean值

}
```

- 在UserDao里面

```
public int active(String code){
                //操作数据库 更新状态
                //返回几行受影响
}
```

三,代码实现

- UserServlet.java

```
    /**
     * 用户激活
     * @param request
     * @param response
     */
    public void active(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        try {
            //1.获得激活码
            String code = request.getParameter("code");
            //2.根据激活码激活
            boolean isActive = userService.active(code);
            //3. 判断是否激活成功
            if(isActive){
                response.sendRedirect(request.getContextPath()+"/login.html");
            }else{
                response.getWriter().print("<h1>激活失败</h1>");
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().print("<h1>激活失败</h1>");
        }
    }
```

- UserService.java

```
    /**
     * 用户激活
     * @param code
     * @return
     */
    public boolean active(String code) throws Exception {
       int rows =  userDao.active(code);
        return  rows > 0;
    }
```

- UserDao.java

```
    public int active(String code) throws  Exception {
        //定义插入用户sql语句
        String sql = "update tab_user set status = ?  where code = ?";
        return  jdbcTemplate.update(sql,"Y",code);
    }
```

案例三-用户登录

一,案例需求

在登录页面,点击登录按钮, 进行登录.

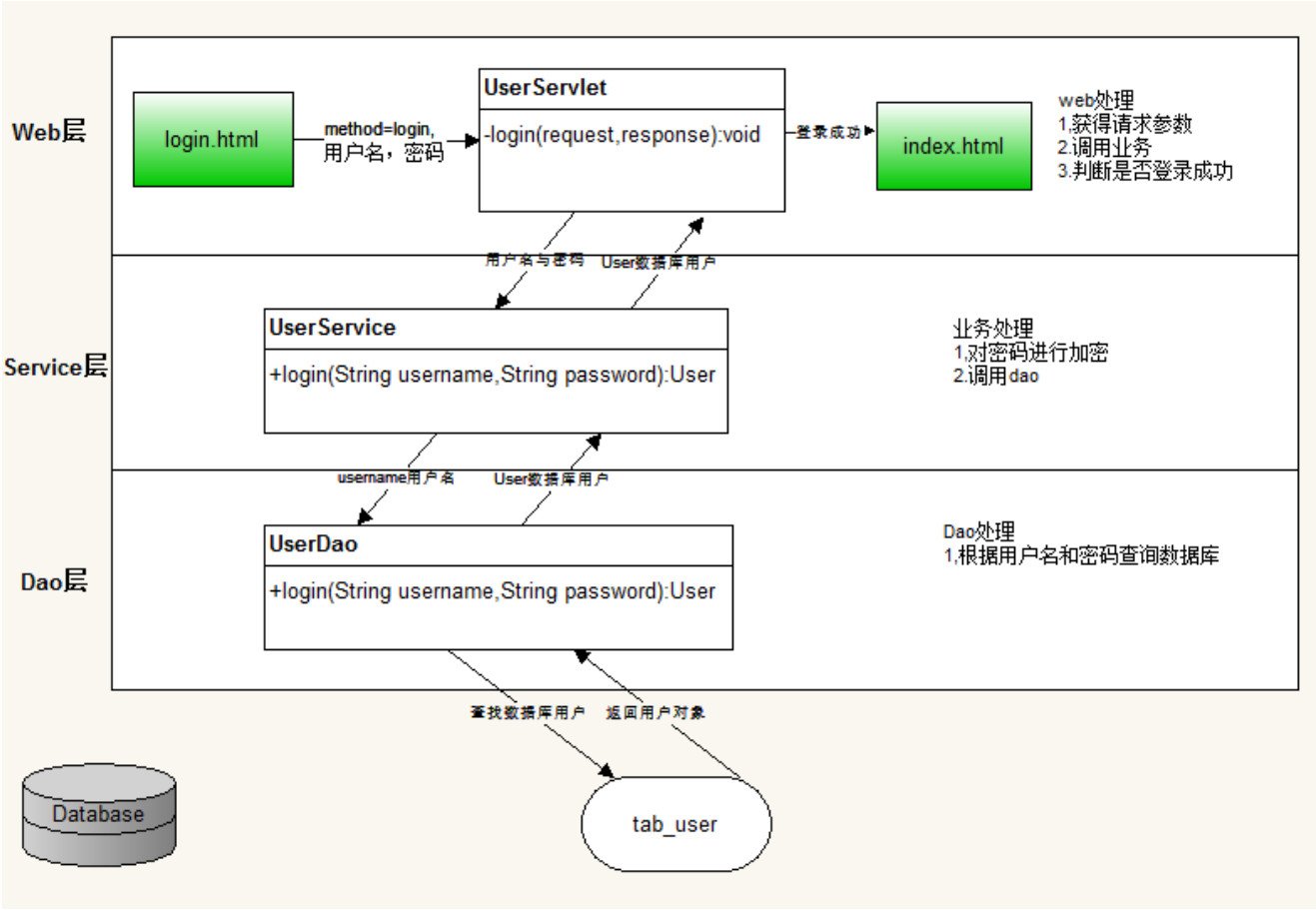
如果登录成功,跳转到网站首页;

如果登录失败, 在当前页面(登录页面)提示用户


欢迎登录黑马旅游账户
请输入账号
请输入密码
请输入验证码
345D
登录
自动登录
没有账户？立即注册
登录成功，显示登录信息
http://localhost:8080/travel/index.html
欢迎回来，zhangsan 我的收藏 退出
请输入路线名称
搜索
酒店 香港车票 出境游 国内游 港澳游 抱团定制 全球自由行 收藏排行榜
没有登录显示这样
http://localhost:8080/travel/index.html
登录 注册
客服热线(9:00-6:00)
400-618-9090
请输入路线名称
搜索
酒店 香港车票 出境游 国内游 港澳游 抱团定制 全球自由行 收藏排行榜

二,案例思路

1.登录思路



- 在login页面, 点击登录按钮, 会触发表单的提交, 给表单设置提交事件, 创建函数响应这个事件
- 在函数里面:

```
function(){
        //1. 获得表单里面需要提交的数据(用户名,密码.method方法名) data
        //2. 发送Ajax请求, 把数据提交过去
        $.post("user",data,function(result){

        },"json");

        //3.阻止表单自身的提交
        return false;


}
```

- 在UserServlet里面 创建 login()方法

```
public void login(request,response){
        //1. 获得用户名和密码
        //2. 调用业务, 进行登录 获得User对象
        //3. 判断是否登录成功, 给前端响应
}
```

- 在UserService

```
public User login(String username,String password){
        //把password加密一下(注册的时候加密的)
        //调用Dao, 根据用户名和密码查询数据库

}
```

- UserDao

```
public User login(String username,String password){
        // 根据用户名和密码查询数据库
}
```

2.实现header位置显示登录数据功能

三,代码实现

1.登录代码实现

- login.html

```
<script>
    //给表单设置提交事件
    $("#loginForm").submit(function () {
        //给表单数据序列化
        var data =  $("#loginForm").serialize();
        //发送Ajax请求
        $.post("user",data,function (result) {
            if (result.flag){
                //登录成功
                 location.href = "index.html";
            }else{
                //登录失败
                 $("#errorMsg").html(result.msg);
            }

        },"json");

        //阻止表单同步提交
        return false;
    });

</script>
```

- UserServlet.java

```
    /**
     * 用户登录
     * @param request
     * @param response
     */
    public void login(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String data = null;
        ResultInfo resultInfo = null;
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            //1. 获得请求参数
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            //2. 调用业务
            User user = userService.login(username, password);
            //3. 判断是否登录成功
            if(user != null){
                if("Y".equals(user.getStatus())){
                    //登录成功
                    request.getSession().setAttribute("user",user);
                    resultInfo = new ResultInfo(true,null,"登录成功...");
                }else{
                    resultInfo = new ResultInfo(false,null,"您还没有激活...");
                }

            }else{
                resultInfo = new ResultInfo(false,null,"用户名和密码不一致...");
            }
        } catch (Exception e) {
            e.printStackTrace();
            resultInfo = new ResultInfo(false,null,"服务器异常,登录失败...");
        }finally {
            data = objectMapper.writeValueAsString(resultInfo);
            response.getWriter().print(data);
        }

    }
```

- UserService.java

```
    /**
     * 用户登录
     * @param username
     * @param password
     * @return
     */
    public User login(String username, String password) throws Exception {
        password = Md5Util.encodeByMd5(password);
        return  userDao.login(username,password);

    }
```

- UserDao.java

```
    /**
     * 用户登录
     * @param username
     * @param password
     * @return
     */
    public User login(String username, String password) throws Exception {
        String sql="select * from tab_user where username = ? and password = ?";
        User user = jdbcTemplate.queryForObject(sql, new BeanPropertyRowMapper<>(User.class),
username, password);
        return  user;
    }
```

2.实现header位置显示登录数据功能

- header.html

```
<script src="js/jquery-3.3.1.js"></script>
<script>
    $(function () {
        //先隐藏登录的界面
        $(".login").hide();
        //发送Ajax请求获得用户的登录状态
        $.post("user",{method:"getLoginInfo"},function (result) {
            if (result.flag){
                //a.隐藏没有登录界面
                $(".login_out").hide();
                //b.展示登录的界面
                $(".login").html(" <span>欢迎回来,"+result.data.username+"</span>\n" +
                    "<a href=\"myfavorite.html\" class=\"collection\">我的收藏</a>\n" +
                    " <a href=\"javascript:;\">退出</a>");
                $(".login").show();
                }else{
                //隐藏登录的解码
                $(".login").hide();
                //显示没有登录界面
                $(".login_out").show();
            }

        },"json");
    });

</script>
```

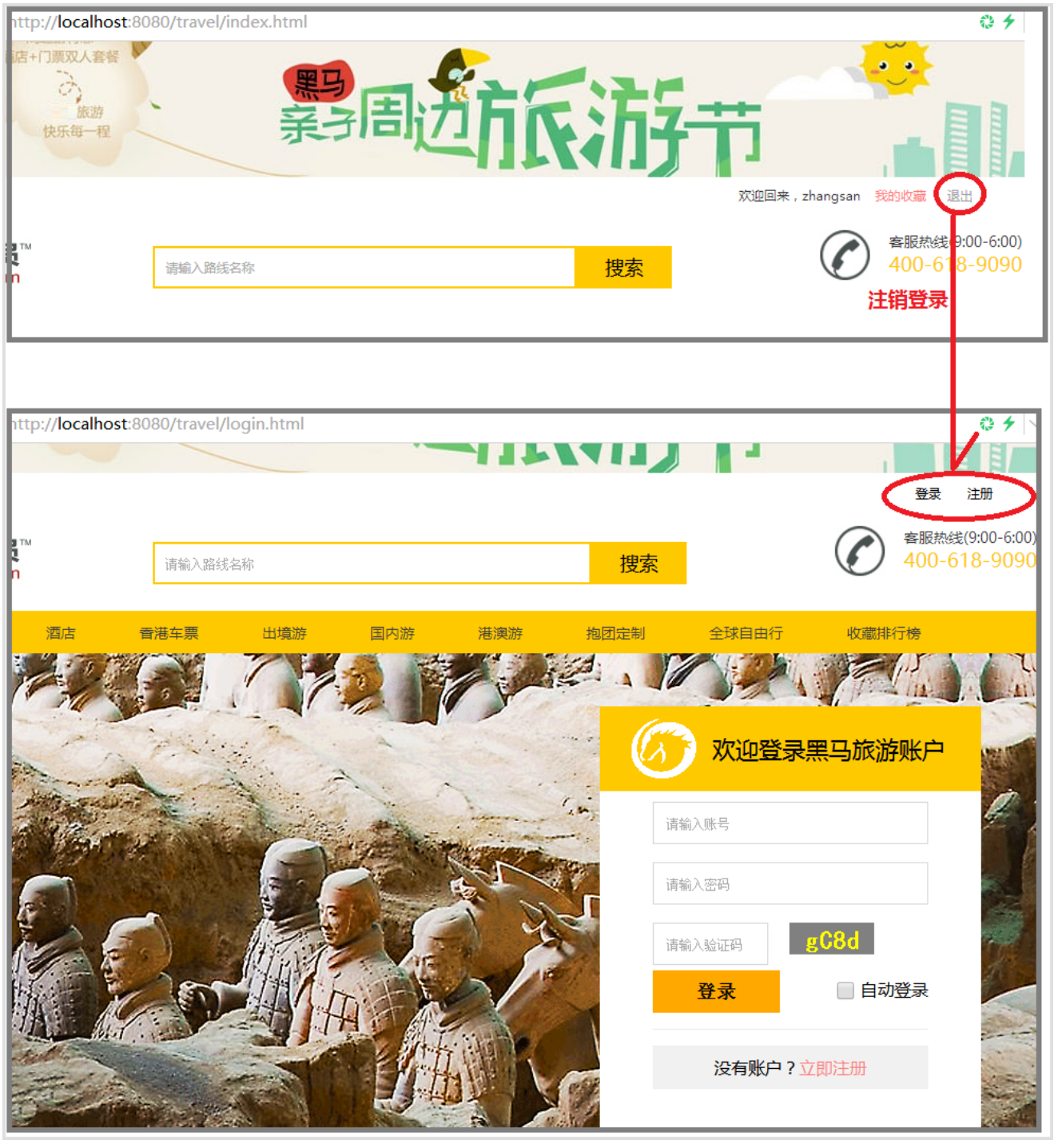
- UserServlet.java

```
    /**
     * 获得登录状态
     * @param request
     * @param response
     */
    public void getLoginInfo(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String data = null;
        ResultInfo resultInfo = null;
        ObjectMapper objectMapper = new ObjectMapper();

        User user  = (User) request.getSession().getAttribute("user");
        if(user != null){
            //是登录状态
            resultInfo = new ResultInfo(true,user,"已经登录");
        }else{
            //不是登录状态
            resultInfo = new ResultInfo(false,null,"未登录");
        }

        data =  objectMapper.writeValueAsString(resultInfo);
        System.out.println("data="+data);
        response.getWriter().print(data);
    }
```

案例四-退出登录

一,案例需求

在网站首页, 点击退出, 注销当前用户, 跳转到登录页面


http://localhost:8080/travel/index.html
欢迎回来，zhangsan
我的收藏
退出
请输入路线名称
搜索
客服热线(9:00-6:00)
400-618-9090
注销登录
http://localhost:8080/travel/login.html
登录
注册
请输入路线名称
搜索
客服热线(9:00-6:00)
400-618-9090
酒店
香港车票
出境游
国内游
港澳游
抱团定制
全球自由行
收藏排行榜
欢迎登录黑马旅游账户
请输入账号
请输入密码
请输入验证码
gC8d
登录
自动登录
没有账户？立即注册

二,案例思路


Web层
header.html
页面加载时请求UserServlet
method=logout
UserServlet
-logout(request,response):void
1.销毁session
2.跳转到登录页面
获取成功
login.html

三,代码实现

- header.jsp

```
<script>
    $(function() {
        $(".login").hide();

        $.post("user",{method:"getLoginUserData"},function(result){
            if (result.flag){
                $(".login").html(" <span>欢迎回来，"+result.data.name+"</span>\n" +
                    "  <a href=\"myfavorite.html\" class=\"collection\">我的收藏</a>\n" +
                    " <a href=\"user?method=logout\">退出</a>");
                $(".login").show();
                $(".login_out").hide();
            }else{
                $(".login_out").show();
            }

        },"json");
    });
</script>
```

请求userServlet，方法为logout

- UserServlet.java

```
    /**
     * 退出登录
     * @param request
     * @param response
     */
    public void logout(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        //移除登录状态
        request.getSession().removeAttribute("user");
        //重定向到登录页面
        response.sendRedirect(request.getContextPath()+"/login.html");

    }
```