day41-综合实战第一天

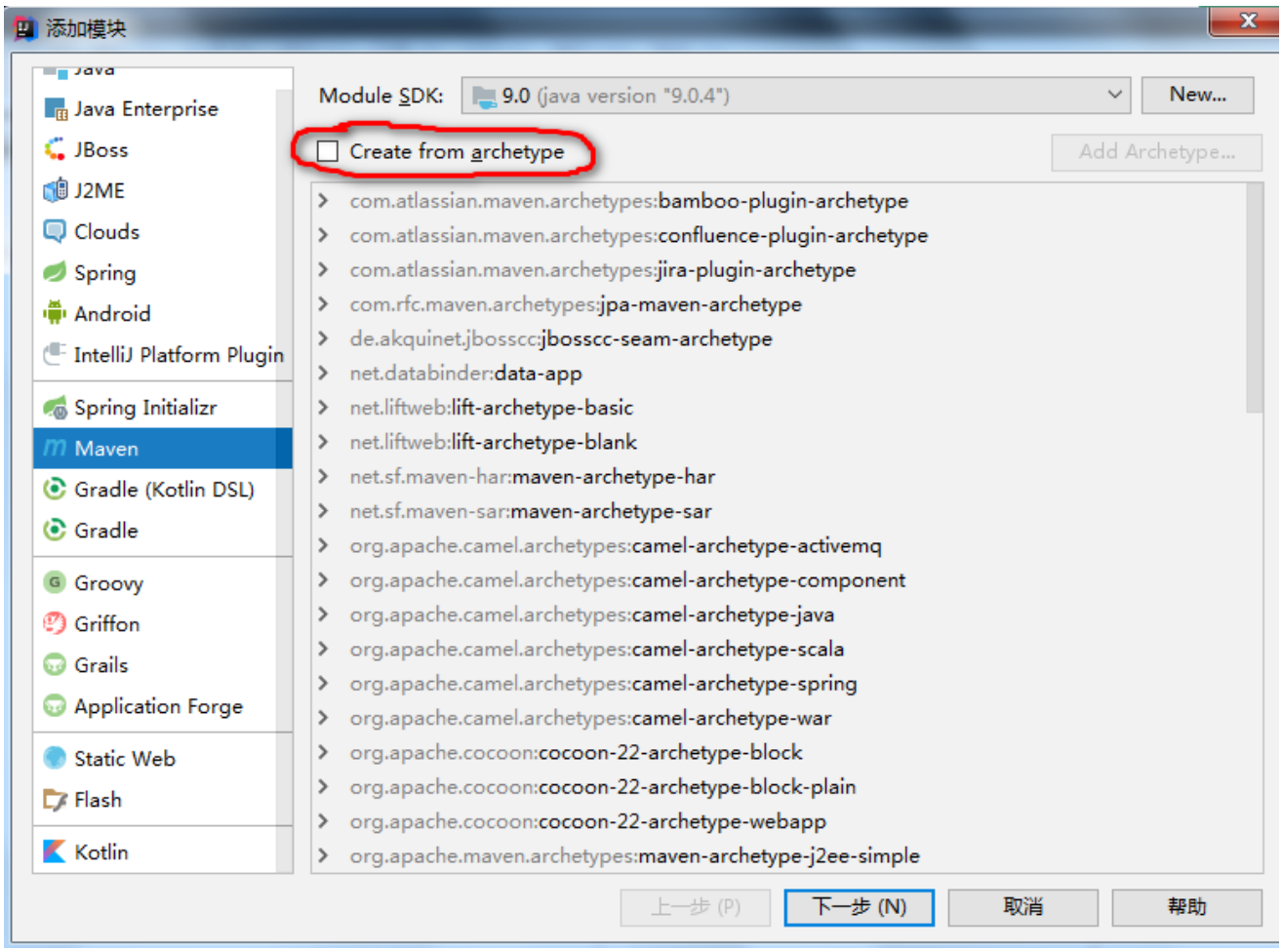
学习目标

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

不联网不使用骨架创建maven的方法

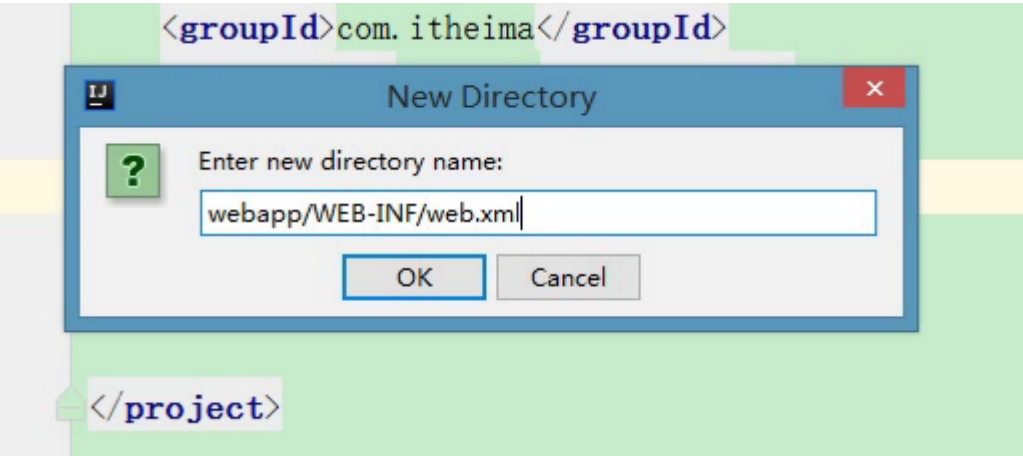
java项目



web项目:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.itheima</groupId>
    <artifactId>javaweb-demo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

</project>
```

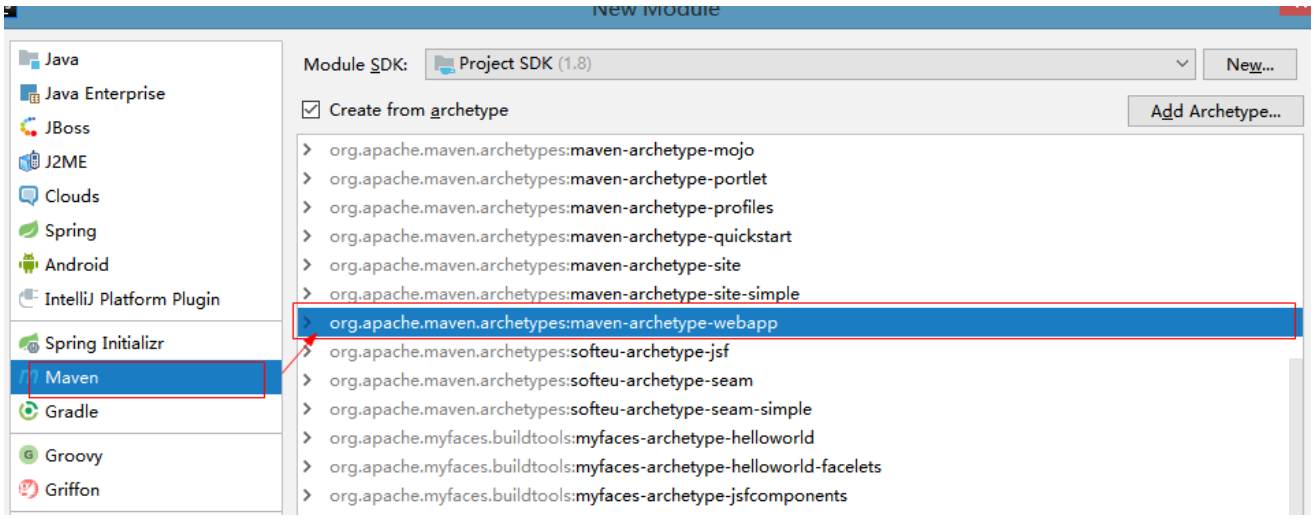
添加这个标签



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

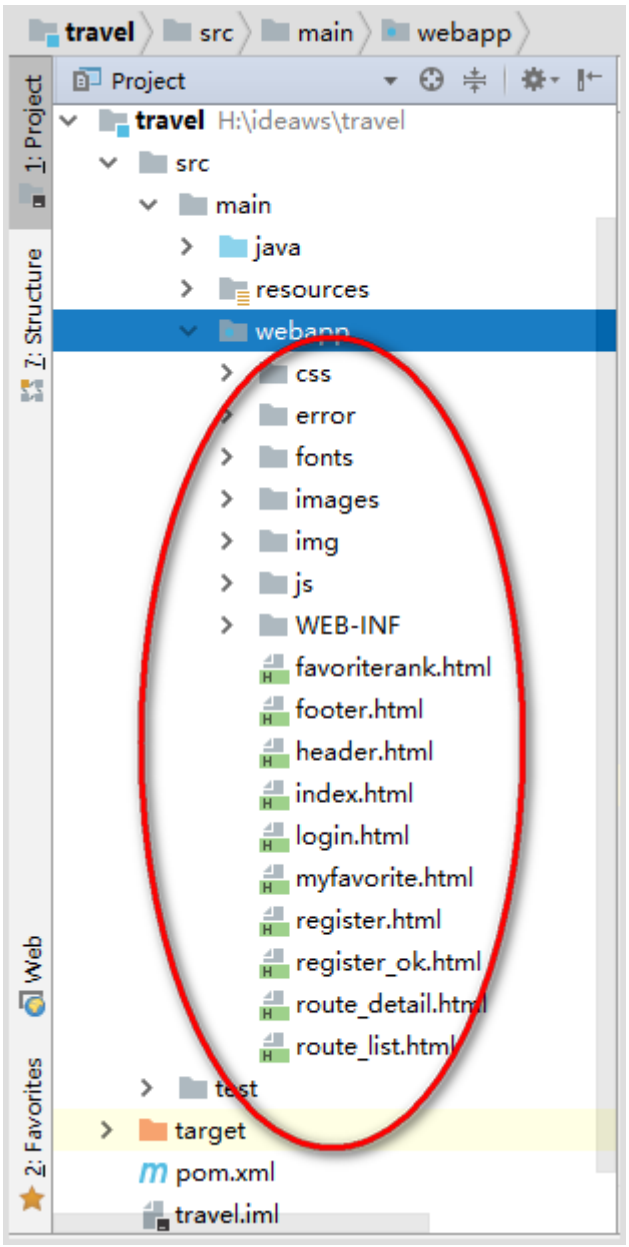
```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
    <!--servlet-->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</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>
```

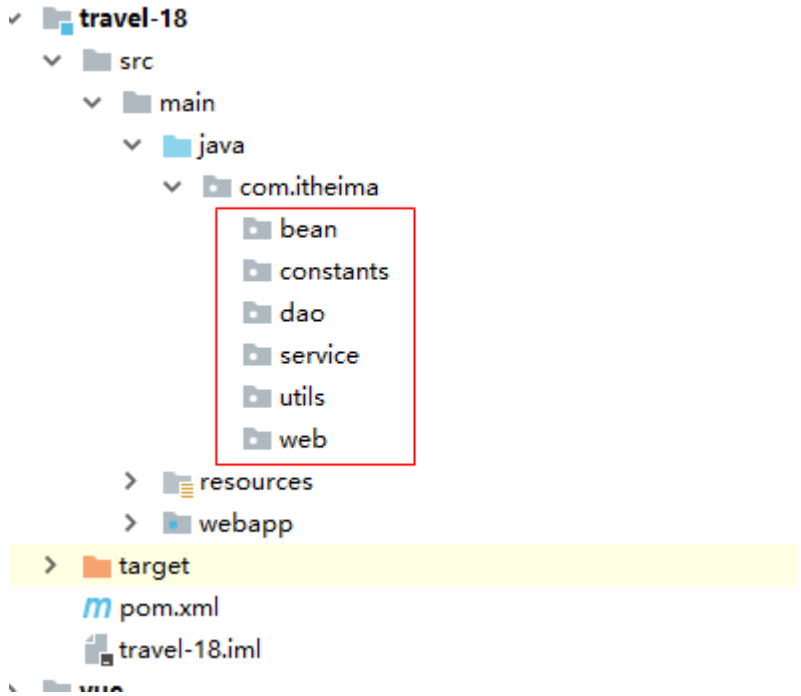
```
            <groupId>com.sun.mail</groupId>
            <artifactId>javax.mail</artifactId>
            <version>1.5.3</version>
        </dependency>
        <!--jedis-->
        <dependency>
            <groupId>redis.clients</groupId>
            <artifactId>jedis</artifactId>
            <version>2.7.0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

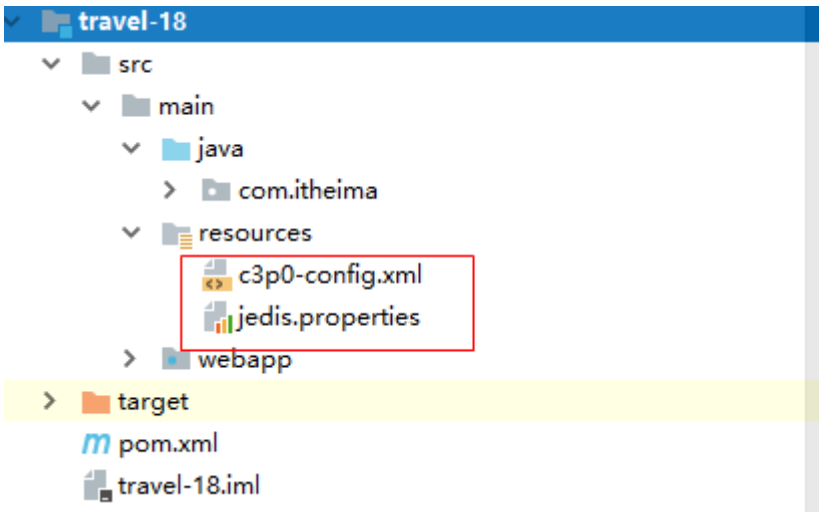


4.创建包结构



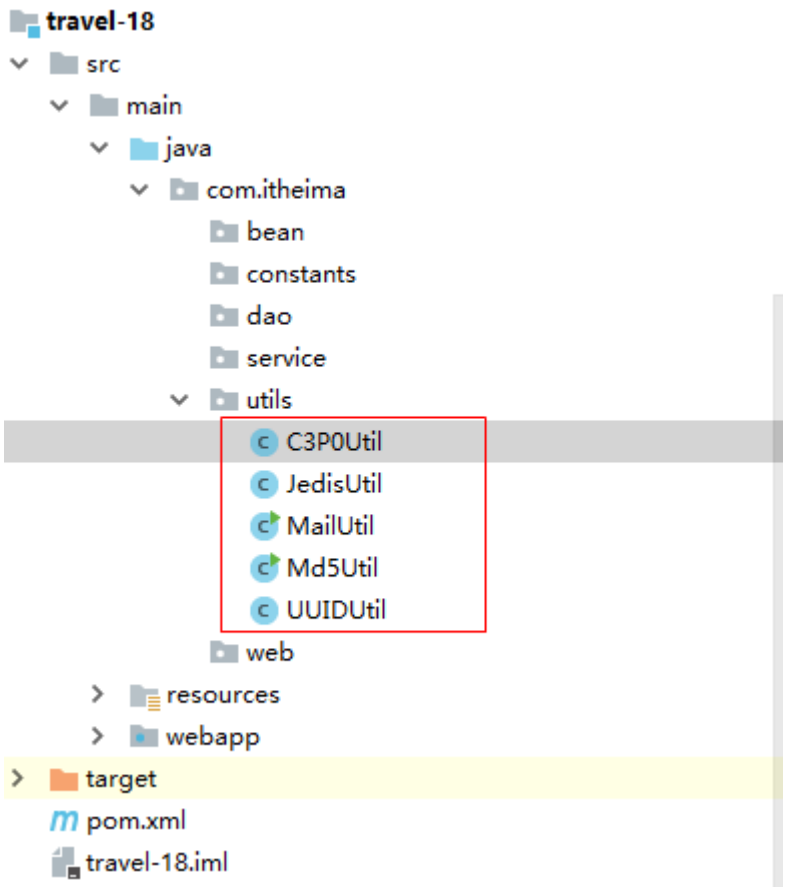
5. 导入配置文件

- 将“资料/02-配置文件”导入到resource资源目录中



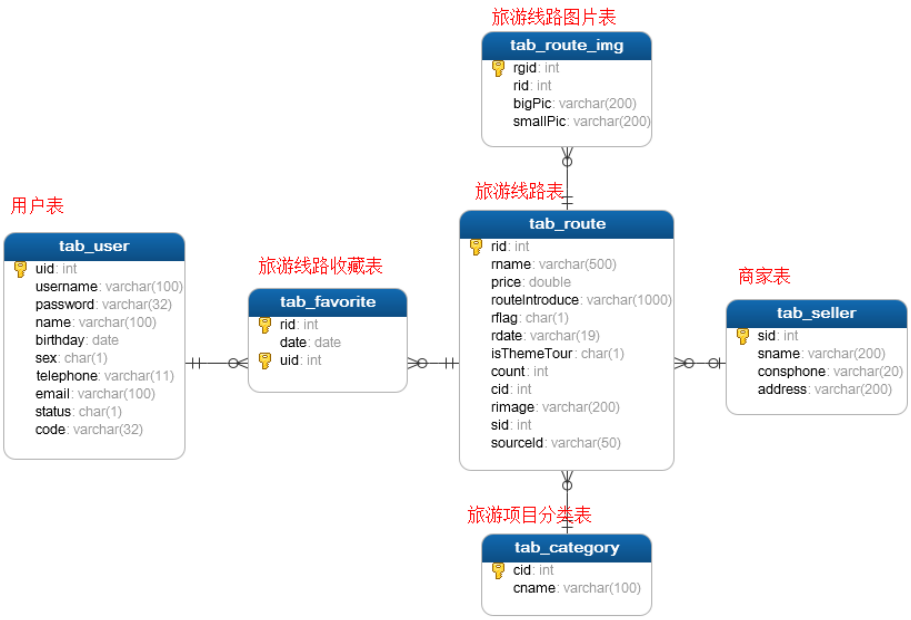
6. 导入工具类

- 将"资料/03-工具类"导入utils包下



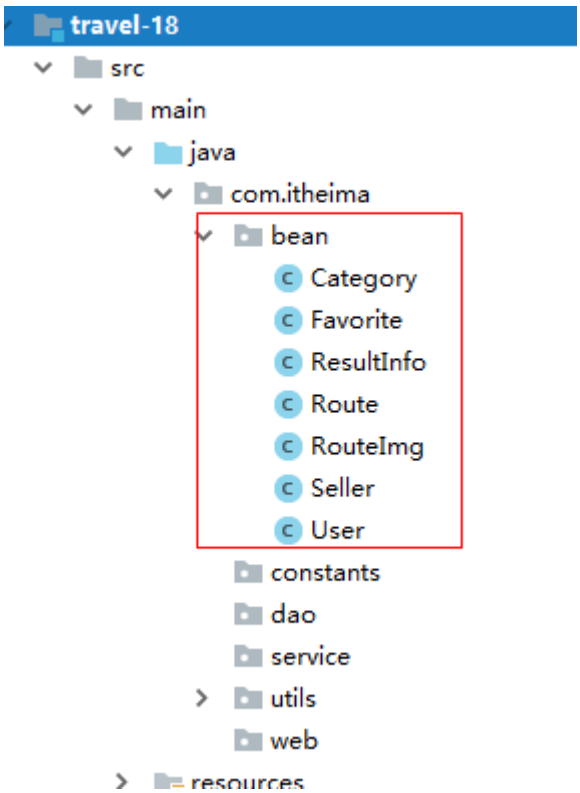
7 导入数据库脚本

到mysql数据库执行“资料/04-数据库脚本”，表之间的关系如下图



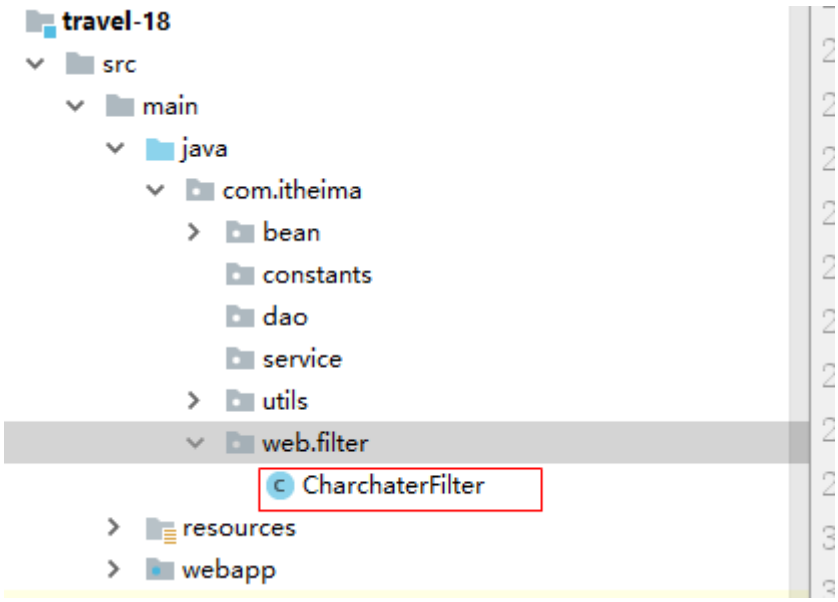
8 导入/创建实体

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

- 反射优化后

```
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
    }
}
```

- 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);

    }
}
```

this是BaseServlet还是UserServlet? 是UserServlet.

原因: ①在一般情况下this代表本类,在继承体系中,this:谁来调用我，我就指向谁; ②UserServlet继承了BaseServlet所以UserServlet也拥有service方法，也就是拥有this; ③请求的是userServlet,调用this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){
    }
     public void findAll(HttpServletRequest request, HttpServletResponse response){
    }
}
```

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

二,案例思路

BaseServlet的编写

```
@WebServlet("/base")
public class BaseServlet extends HttpServlet {
    protected void service(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        try {
            String method = request.getParameter("method");
            Class clazz = this.getClass();
            Method way = clazz.getMethod(method, HttpServletRequest.class,
HttpServletResponse.class);
            way.invoke(this,request,response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

三,代码实现

- ## register.html 注册页面

  根据id选择器通过jquery请求提交由.serialize()得到表单数据

  通过ajax的post请求第一个参数路径，第二个参数表单数据，第三个参数反馈结果，第四个参数指定结果类型json数据

  通过return false阻止页面整体提交

```
<script>
    $("#registerForm").submit(function () {
        var data=$("#registerForm").serialize();
        $.post("user",data,function (result) {
        if(result.flag){
location.href="register_ok.html";
        }else{
alert(result.msg);
        }
        },"json");
        return false;
    });
</script>
```

- ## login.html 登录页面

通过id选择器提交表单数据，serialize()获得表单数据

通过ajax的post请求发送给servlet 参数有 路径，表单数据，方法返回结果，返回值类型

```
<script>
    $("#loginForm").submit(function () {
        var data=$("#loginForm").serialize();
        $.post("user",data,function (result) {
            if(result.flag){
                location.href="index.html";
            }else{
                $("#errorMsg").html(result.msg);
            }
        },"json");
        return false;
    });
</script>
```

- **header头部分**

发送ajax的post请求到servlet 参数有 路径,方法名数据,返回结果,返回值类型

返回参数如果为true则显示登录后的页面

```
<script>
$(".login").hide();
$.post("user",{method:"getlogin"},function (result) {
    if(result.flag){
$(".login_out").hide();
$(".login").show();
        $(".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=\"user?method=exitout\">退出</a>");
    }
},"json")
</script>
```

- **UserServlet继承基类反射方法，本类方法名要与request请求的方法名一致**

1，**注册** 服务器启动后进入首页，点击首页进入注册页面，通过上述方法将数据传给servlet

servlet通过request请求得到数据map，然后用BeanUtils工具将map转化为User对象 存入数据库

如果存入成功就响应true以及注册成功的信息

2，**激活** 在注册的时候，会得到一个激活码，存储到数据库同时发送链接携带激活码给邮箱。

在用户点击邮箱链接的时候，会走servlet中的active方法得到激活码，然后用激活码更新数据库将状态码N改为Y，如果用户存在，则servlet重定向到登录页面

3，**登录** 当用户点击登录之后，servlet会获得用户名和密码，通过用户名和密码查询数据库，如果存在并且状态码是Y，就将查到的User对象存到Session里面去，并且响应登录信息

4，**状态判断** 首页头部分发送过来ajax无参数请求，servlet获取服务器中的Session数据，如果Session中的user不为null,则响应回给浏览器登录过的状态信息，浏览器根据拿到的flag选择隐藏或者显示登录及用户界面

5，**退出** 当用户点击退出超链接时发动，超链接中带有路径和方法参数，直接在servlet中清空Session中的user数据并重定向到登录界面

```
@WebServlet("/user")
public class UserServlet extends BaseServlet {
    private UserService userService=new UserService();
    public void register(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        ResultInfo res=null;
        ObjectMapper objectMapper =   new ObjectMapper();

        try {
            Map<String, String[]> map = request.getParameterMap();
            User user=new User();
            BeanUtils.populate(user,map);
            userService.regist(user);
            res = new ResultInfo(true, null, "注册成功");
        } catch (Exception e) {
            e.printStackTrace();
            res = new ResultInfo(false, null, "注册失败");
        }finally{
            String s = objectMapper.writeValueAsString(res);
            response.getWriter().print(s);
        }
    }

    public void active(HttpServletRequest request, HttpServletResponse response) {
        try {
            String code = request.getParameter("code");
            boolean aflag = userService.Cactive(code);
            if(aflag){
            response.sendRedirect(request.getContextPath()+"/login.html");
            }else{
response.getWriter().print("激活失败");
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void login(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        ResultInfo res=null;
        ObjectMapper objectMapper =   new ObjectMapper();
        try {
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            UserService us=new UserService();
            User usera = us.loginuser(username, password);
            if(usera!=null){
                if(Constants.USER_ACTIVE.equals(usera.getStatus())){
                request.getSession().setAttribute("user",usera);
                    res=new ResultInfo(true,null,"登录成功");
```

```
                }else{
                    res=new ResultInfo(false,null,"尚未激活");
                }
            }else{
                res=new ResultInfo(false,null,"登录失败空");
            }
        } catch (Exception e) {
            e.printStackTrace();
            res=new ResultInfo(false,null,"登录失败异常");
        }finally{
String data=objectMapper.writeValueAsString(res);
            System.out.println(data+"");
response.getWriter().print(data);
        }
    }
    public void getlogin(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        ObjectMapper objectMapper =   new ObjectMapper();
        ResultInfo res=null;
        try {
            User user = (User) request.getSession().getAttribute("user");
            if(user!=null){
                res=new ResultInfo(true,user,"登录过了");
            }else{
                res=new ResultInfo(false,null,"未登录");
            }
        } finally {
            String data=objectMapper.writeValueAsString(res);
            System.out.println(data);
            response.getWriter().print(data);
        }
    }
    public void exitout(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        request.getSession().removeAttribute("user");
        response.sendRedirect(request.getContextPath()+"/login.html");
    }
}
```

- ## UserService

注册

设置user状态码为N和user激活码

将用户密码加密并赋值给user

然后存储数据到数据库中同时发送邮件携带路径激活码

激活 根据激活码修改数据库中的状态码N 改为Y 返回修改成功的数量

登录 将密码加密后赋值给user，查询数据库返回user对象

```
public class UserService {
```

```
        private UserDao userDao=new UserDao();
    public void regist(User user) throws Exception {
        user.setStatus(Constants.USER_NOT_ACTIVE);
        user.setCode(UUIDUtil.getUuid());
        String pass = Md5Util.encodeByMd5(user.getPassword());
        user.setPassword(pass);
        userDao.save(user);
        MailUtil.sendMail(user.getEmail(),"<a href='http://localhost/user?
method=active&code="+user.getCode()+"'>用户激活</a>");
    }
    public boolean Cactive(String code) throws Exception{
        int ac = userDao.active(code);
        return ac>0;
    }

    public User loginuser(String username, String password) throws Exception {
        String sub = Md5Util.encodeByMd5(password);
        password=sub;
        User userc = userDao.logindao(username, password);
        System.out.println(userc);
        return userc;
    }
}
```

- **UserDao**

注册 将user对象通过构造get方法存储到数据库中

激活 根据激活码修改状态码为Y

登录 根据用户名和密码查询数据库，得到user对象

```
public class UserDao {
    private JdbcTemplate jt=new JdbcTemplate(C3P0Util.getDataSource());
    public void save(User user)throws Exception {
        Object[]para={

user.getUsername(),user.getPassword(),user.getName(),user.getBirthday(),user.getSex(),
                user.getTelephone(),user.getEmail(),user.getStatus(),user.getCode()
        };
        jt.update("insert into tab_user values (null,?,?,?,?,?,?,?,?,?,?)",para);
    }
    public int active(String code)throws Exception{
        int upd = jt.update("update tab_user set status ='Y'where code=?", code);
        return upd;
    }

    public User logindao(String username, String password)throws Exception {
        System.out.println(username+""+password);
        User userb = jt.queryForObject("select * from tab_user where username=? and password=?",
new BeanPropertyRowMapper<>(User.class), username, password);
        System.out.println(userb);

        return userb;
```

```
    }
}
```

- **Constants 状态码常量**

```
public interface Constants {
    String USER_NOT_ACTIVE ="N";
    String USER_ACTIVE="Y";
}
```

- **User**

```
public class User implements Serializable {
    private int uid;
    private String username;
    private String password;
    private String name;
    private String birthday;
    private String sex;
    private String telephone;
    private String email;
    private String status;
    private String code;
    构造get/set
    toString方法
    }
```

- **ResultInfo**

```
public class ResultInfo implements Serializable {
    private boolean flag;
    private Object data; //后端返回结果数据对象
    private String msg;

    public ResultInfo() {
    }
    public ResultInfo(boolean flag) {
        this.flag = flag;
    }

    public ResultInfo(boolean flag, String msg) {
        this.flag = flag;
        this.msg = msg;
    }

    public ResultInfo(boolean flag, Object data, String msg) {
        this.flag = flag;
        this.data = data;
        this.msg = msg;

    }
```

```
    构造get/set
    toString()方法
}
```

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

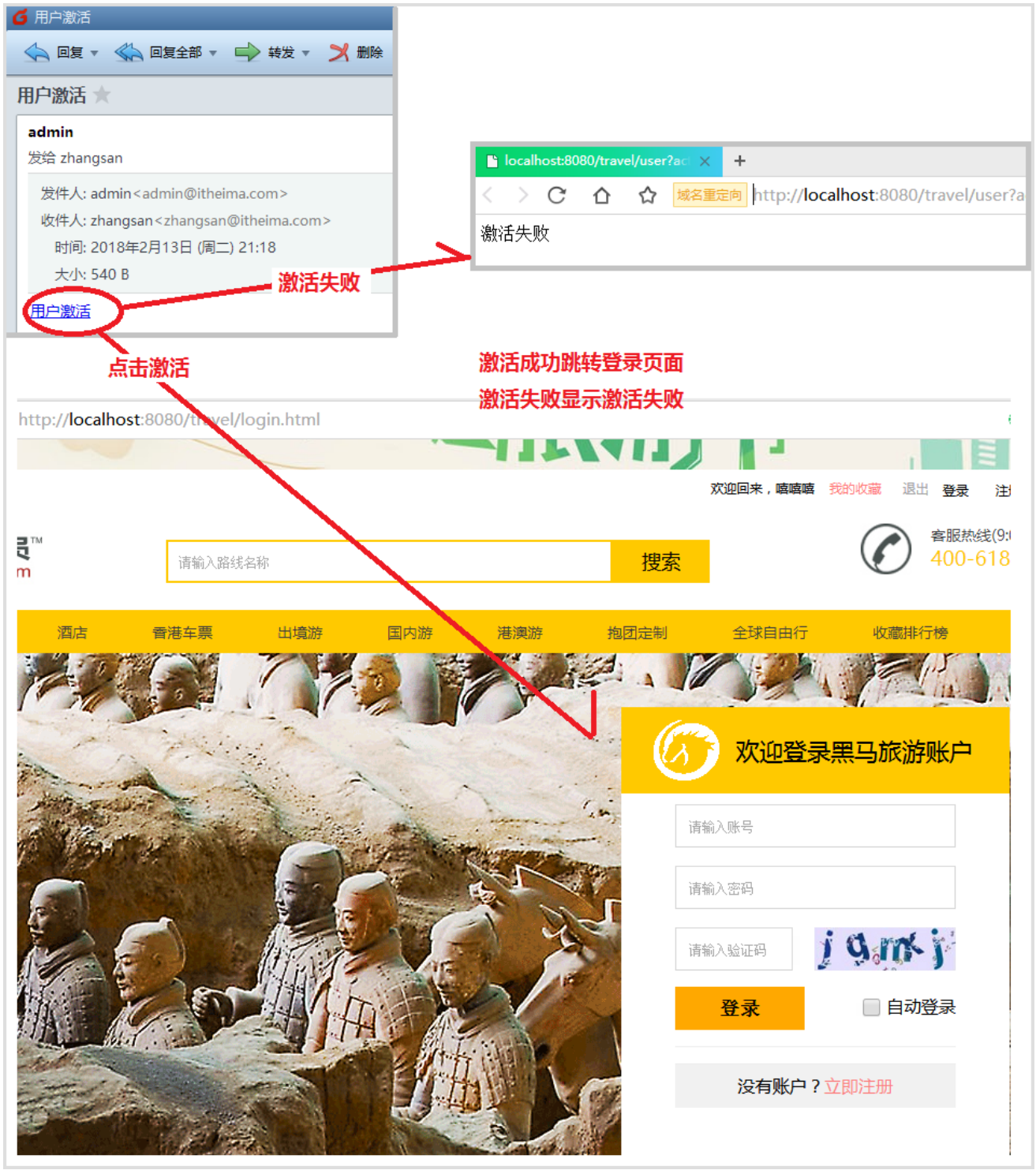
案例二-用户激活

一,案例需求　注意方法千万别私有

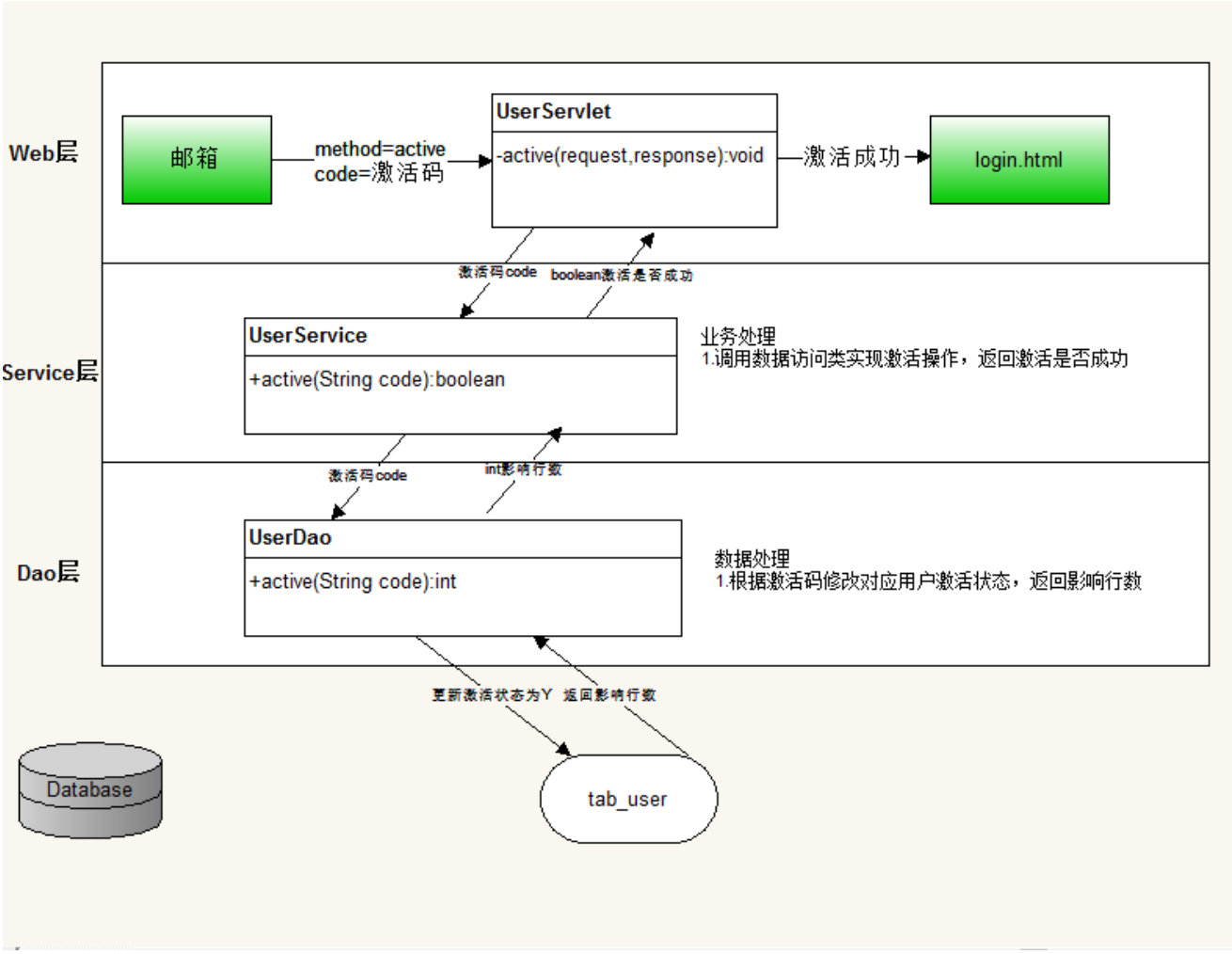
用户登录邮箱, 点击激活超链接,

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?ac
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



三,代码实现

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.

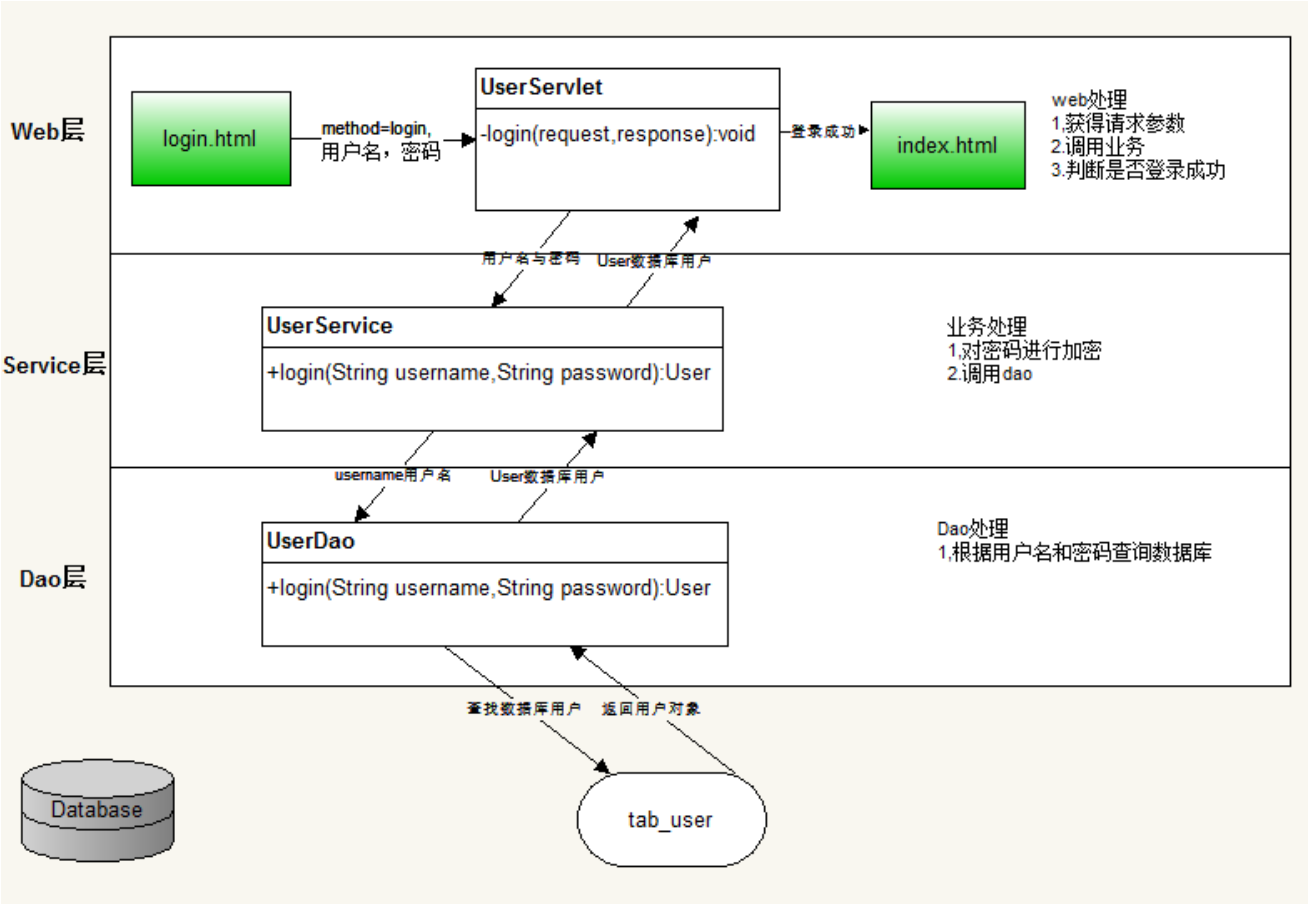
如果登录成功,跳转到网站首页;

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
客服热线(9:00
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
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



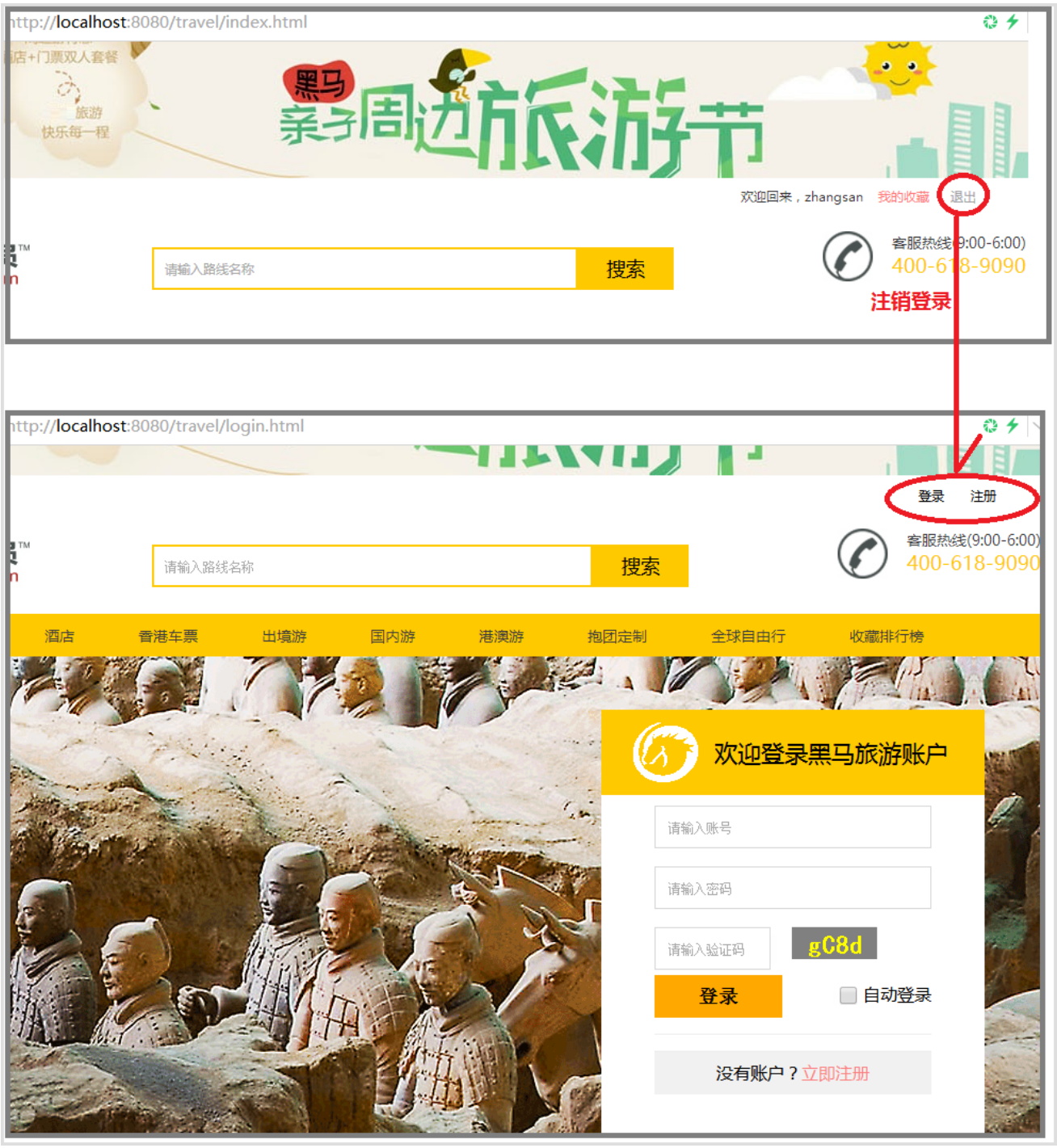
2.实现header位置显示登录数据功能

三,代码实现

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
黑马
亲子周边旅游节
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
method=logout
UserServlet
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