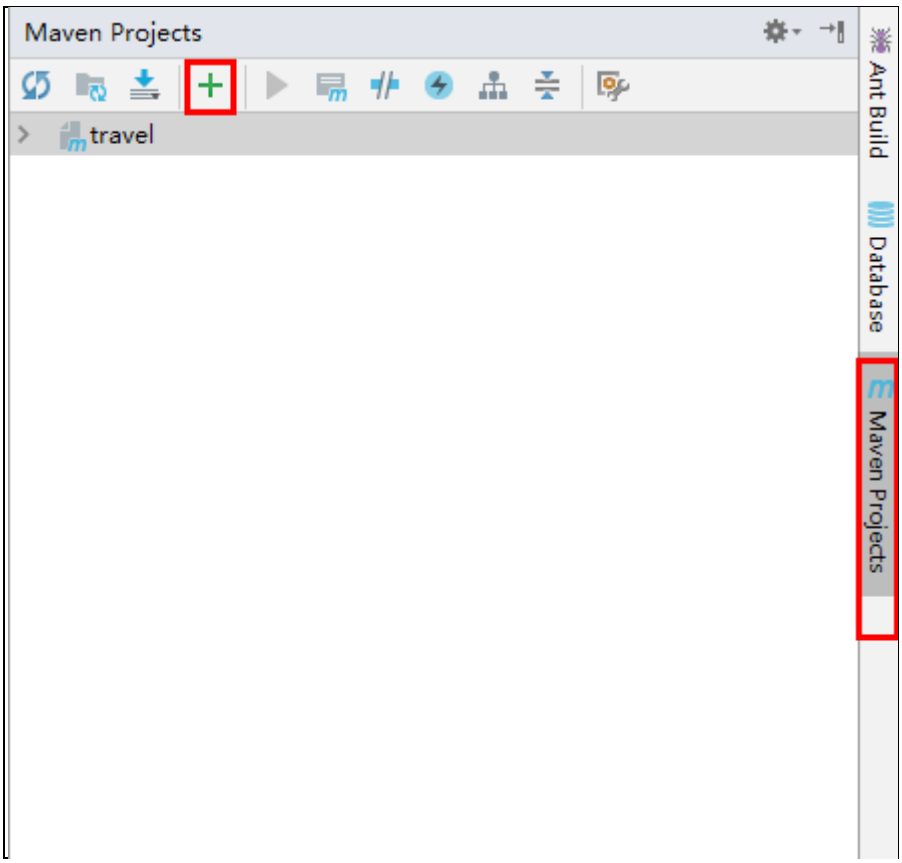
《黑马旅游网》综合案例

1 前言

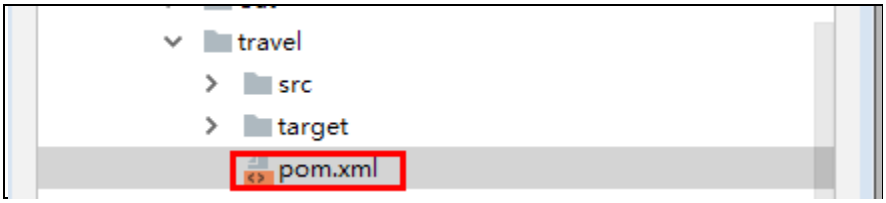
为了巩固 web 基础知识，提升综合运用能力，故而讲解此案例。要求，每位同学能够独立完成此案例。

2 项目导入

点击绿色＋按钮

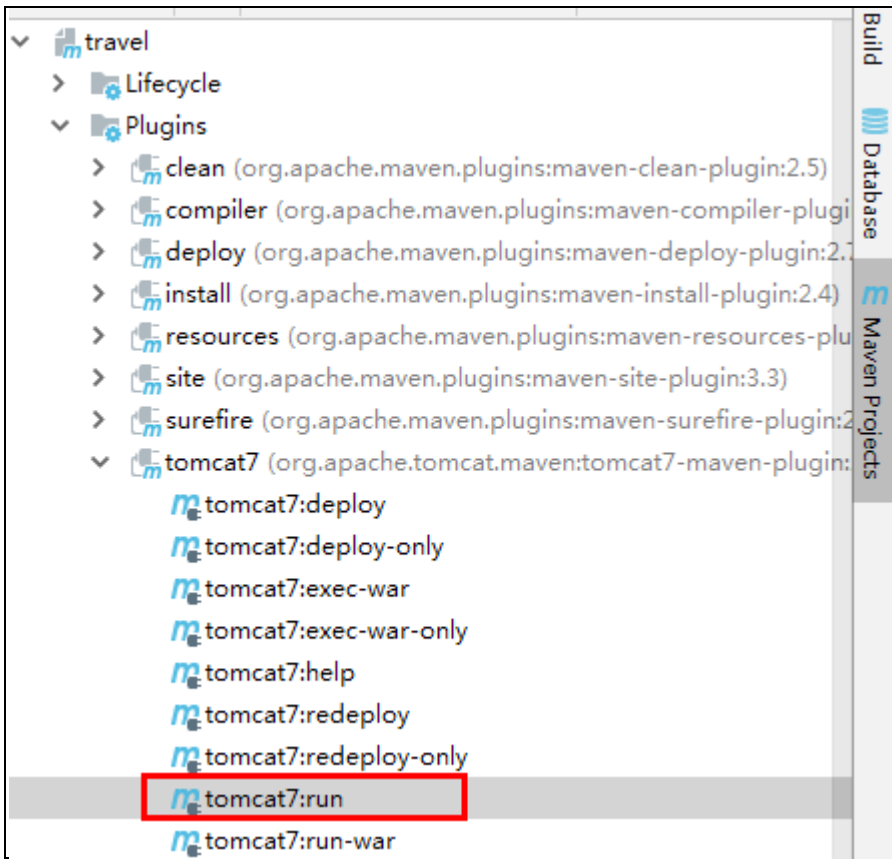


选择 travel 项目的 pom.xml 文件，点击 ok，完成项目导入。需要等待一小会，项目初始化完成。

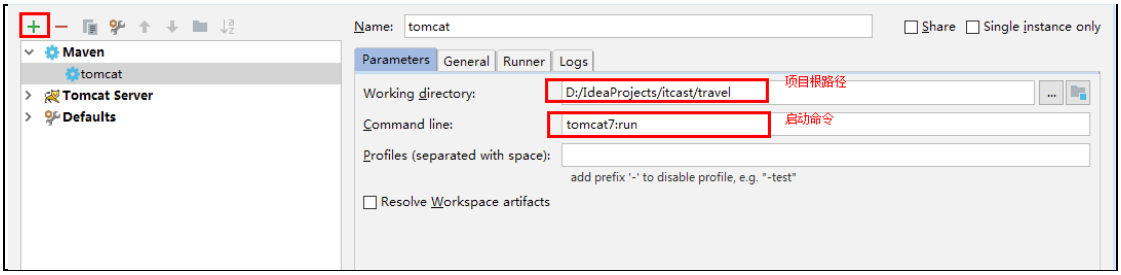


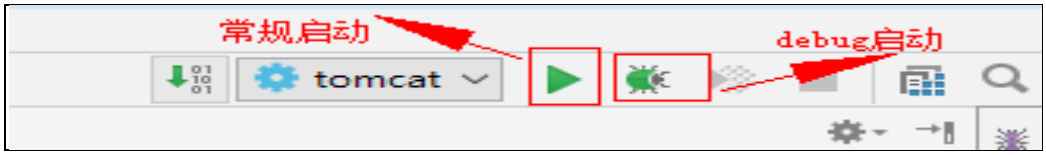
3 启动项目

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

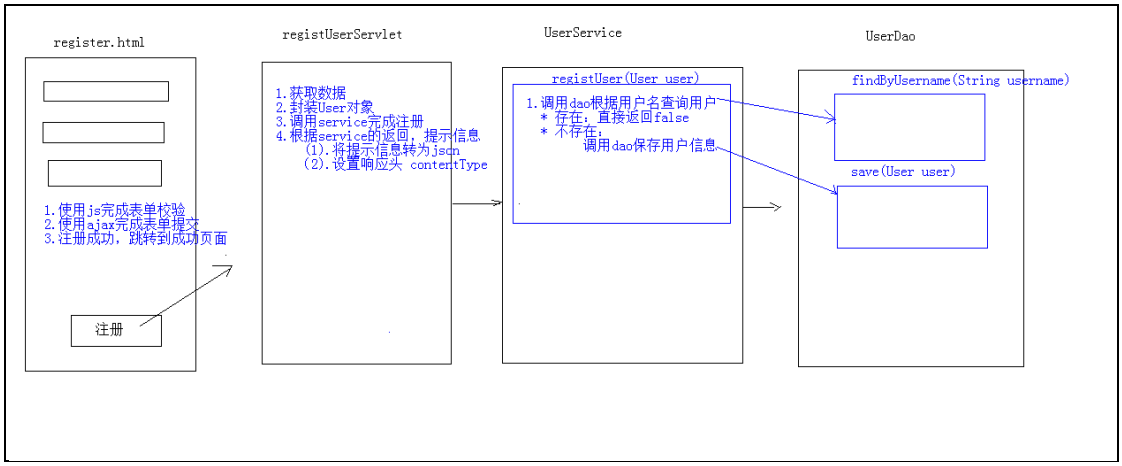
```
-- 创建数据库
CREATE DATABASE travel;
-- 使用数据库
USE travel;
--创建表
```

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)

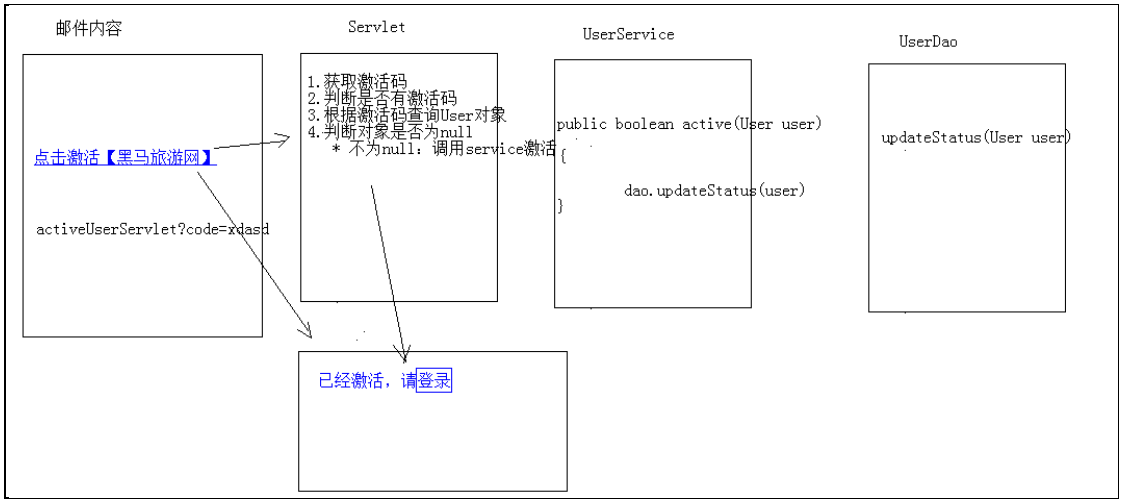
邮件工具类：MailUtils，调用其中 `sendMail` 方法可以完成邮件发送

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

```
@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

```
//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

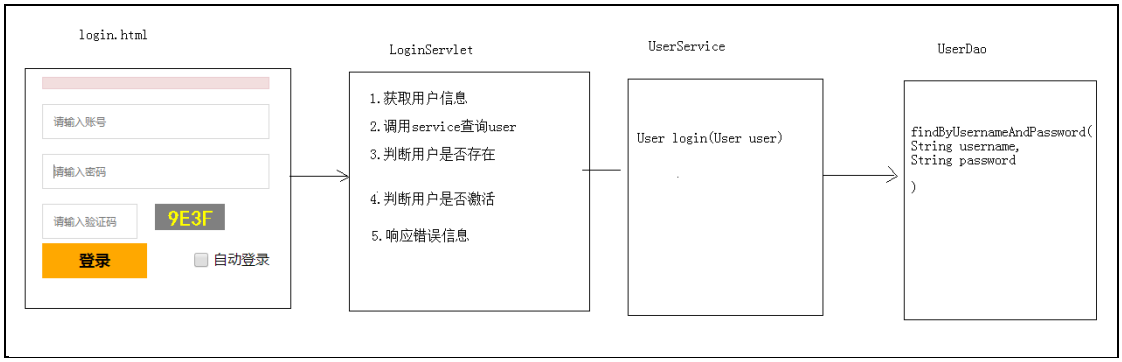
```
@Override
public boolean active(String code) {
    //1.根据激活码查询用户对象
    User user = userDao.findByCode(code);
    if(user != null){
        //2.调用 dao 的修改激活状态的方法
        userDao.updateStatus(user);
        return true;
    }else{
        return false;
    }

}
```

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());
}
```

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

```
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

```
$(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

```
//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>
```

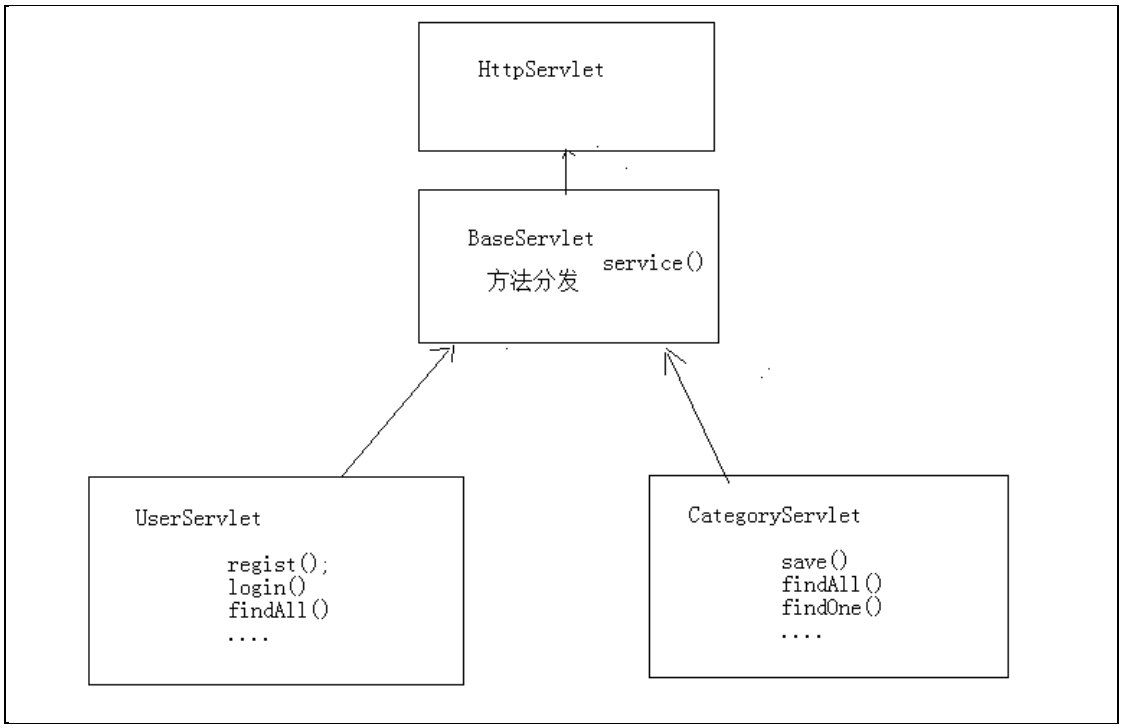
Servlet:

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

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

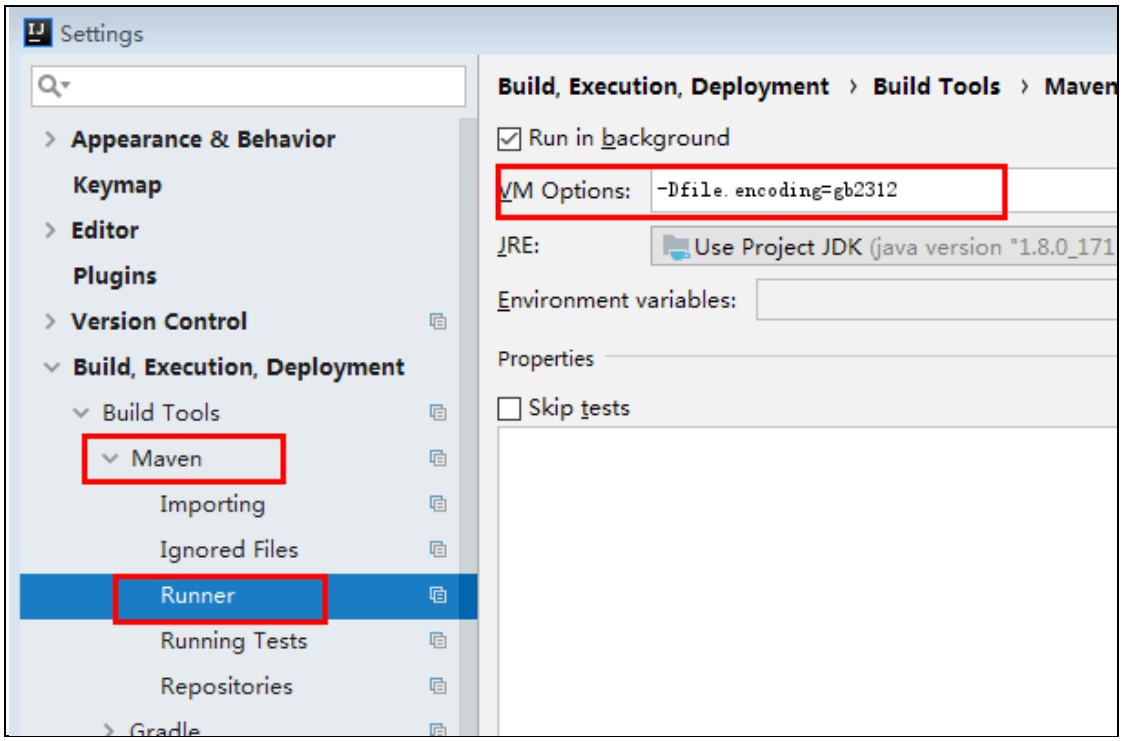
9 优化 Servlet

9.1 目的

减少 Servlet 的数量，现在是一个功能一个 Servlet，将其优化为一个模块一个 Servlet，相当于在数据库中一张表对应一个 Servlet，在 Servlet 中提供不同的方法，完成用户的请求。



Idea 控制台中文乱码解决：-Dfile.encoding=gb2312



9.2 BaseServlet 编写：

```
        public class BaseServlet extends HttpServlet {


    @Override
    protected void service(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
        //System.out.println("baseServlet 的 service 方法被执行
了...");

        //完成方法分发
        //1.获取请求路径
        String uri = req.getRequestURI(); //   /travel/user/add
        System.out.println("请求 uri:"+uri);//  /travel/user/add
        //2.获取方法名称
        String methodName = uri.substring(uri.lastIndexOf('/') +
1);
        System.out.println("方法名称："+methodName);
        //3.获取方法对象 Method
        //谁调用我？我代表谁
        System.out.println(this);//UserServlet 的对象
cn.itcast.travel.web.servlet.UserServlet@4903d97e
        try {
            //获取方法
            Method method = this.getClass().getMethod(methodName,
HttpServletRequest.class, HttpServletResponse.class);
            //4.执行方法
            //暴力反射
            //method.setAccessible(true);
            method.invoke(this,req,resp);
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }


    }
}
```

9.3 UserServlet 改写

将之前的 Servlet 实现的功能，抽取到 UserServlet 中的不同方法中实现，并且将 UserService 创建抽取到成员变量位置

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

    //声明 UserService 业务对象
    private UserService service = new UserServiceImpl();

    /**
     * 注册功能
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void regist(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);

    }

    /**
     * 登录功能
```

```
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void login(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //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())){
            request.getSession().setAttribute("user",u);//登录成
功标记

            //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);
    }

    /**
     * 查询单个对象
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void findOne(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //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);
    }

    /**
     * 退出功能
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void exit(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //1.销毁 session
        request.getSession().invalidate();
```

```
        //2.跳转登录页面

response.sendRedirect(request.getContextPath()+"/login.html");
    }

    /**
     * 激活功能
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void active(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //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);
        }
    }
}
```

9.4 页面路径改写

register.html

```
$.post("user/regist",$(this).serialize(),function(data){
    //处理服务器响应的数据  data  {flag:true,errorMsg:"注册失败"}

    if(data.flag){
        //注册成功，跳转成功页面
        location.href="register_ok.html";
    }else{
        //注册失败,给errorMsg添加提示信息
        $("#errorMsg").html(data.errorMsg);

    }
```

login.html

```
    $.post("user/login",$("#loginForm").serialize(),function (data) {
        //data : {flag:false,errorMsg:''}
        if(data.flag){
            //登录成功
            location.href="index.html";
```

header.html

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

    });
});
```

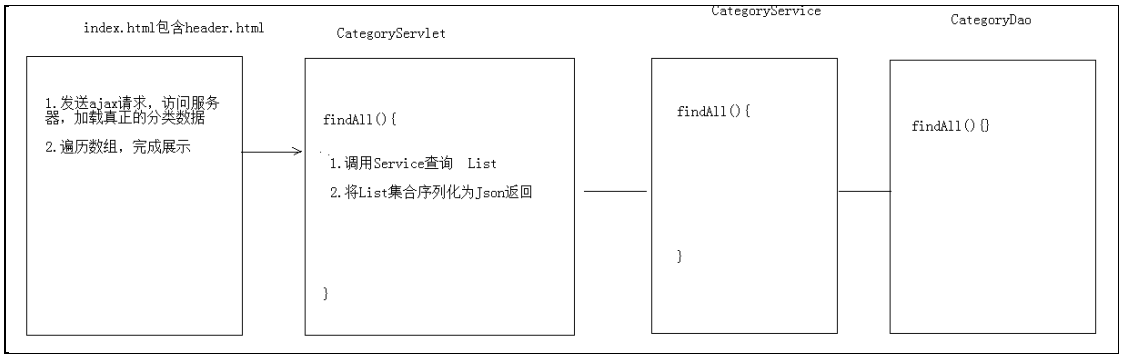
UserServiceImpl 发送邮件

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

10 分类数据展示

10.1 效果：

10.2　　分析：



10.3 代码实现：

10.3.1后台代码

CategoryServlet

```
@WebServlet("/category/*")
public class CategoryServlet extends BaseServlet {

    private CategoryService service = new CategoryServiceImpl();

    /**
     * 查询所有
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void findAll(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //1.调用 service 查询所有
        List<Category> cs = service.findAll();
        //2.序列化 json 返回
       /* ObjectMapper mapper = new ObjectMapper();

response.setContentType("application/json;charset=utf-8");
        mapper.writeValue(response.getOutputStream(),cs);*/
       writeValue(cs,response);
```

```
    }

}
```

CategoryService

```
public class CategoryServiceImpl implements CategoryService {

    private CategoryDao categoryDao = new CategoryDaoImpl();

    @Override
    public List<Category> findAll() {
        return categoryDao.findAll();
    }
}
```

CategoryDao

```
public class CategoryDaoImpl implements CategoryDao {

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

    @Override
    public List<Category> findAll() {
        String sql = "select * from tab_category ";
        return template.query(sql,new
BeanPropertyRowMapper<Category>(Category.class));
    }
}
```

在 BaseServlet 中封装了序列化 json 的方法

```
/**
 * 直接将传入的对象序列化为 json，并且写回客户端
 * @param obj
 */
public void writeValue(Object obj,HttpServletResponse response)
throws IOException {
    ObjectMapper mapper = new ObjectMapper();
    response.setContentType("application/json;charset=utf-8");
    mapper.writeValue(response.getOutputStream(),obj);
}

/**
 * 将传入的对象序列化为 json，返回
```

```
 * @param obj
 * @return
 */
public String writeValueAsString(Object obj) throws
JsonProcessingException {
    ObjectMapper mapper = new ObjectMapper();
    return mapper.writeValueAsString(obj);
}
```

10.3.2 前台代码

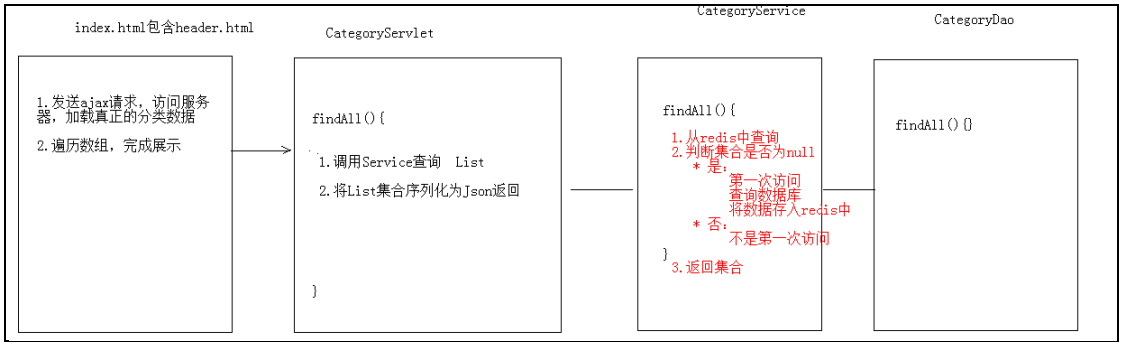
hader.html 加载后，发送 ajax 请求，请求 category/findAll

```
//查询分类数据
 $.get("category/findAll",{},function (data) {
     //[{cid:1,cname:国内游},{},{}]
     var lis = '<li class="nav-active"><a href="index.html">首页
</a></li>';
     //遍历数组,拼接字符串(<li>)
     for (var i = 0; i < data.length; i++) {
         var li = '<li><a
href="route_list.html">'+data[i].cname+'</a></li>';

         lis += li;

     }
     //拼接收藏排行榜的 li,<li><a href="favoriterank.html">收藏排
行榜</a></li>

     lis+= '<li><a href="favoriterank.html">收藏排行榜</a></li>';

     //将 lis 字符串，设置到 ul 的 html 内容中
     $("#category").html(lis);
 });
```

10.4 对分类数据进行缓存优化

分析发现，分类的数据在每一次页面加载后都会重新请求数据库来加载，对数据库的压力比较大，而且分类的数据不会经常产生变化，所有可以使用 redis 来缓存这个数据。

分析：



10.5 优化代码实现

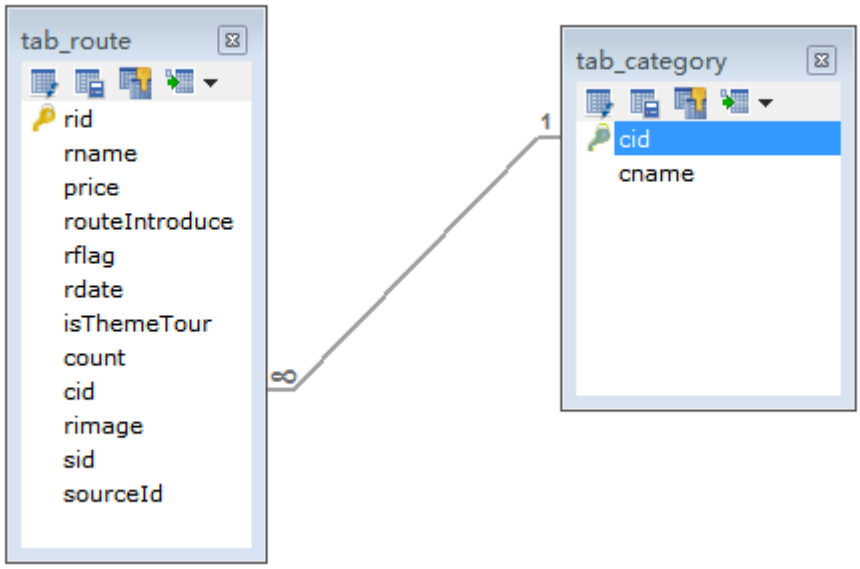
期望数据中存储的顺序就是将来展示的顺序，使用 redis 的 sortedset

```
@Override
public List<Category> findAll() {
    //1.从 redis 中查询
    //1.1 获取 jedis 客户端
    Jedis jedis = JedisUtil.getJedis();
    //1.2 可使用 sortedset 排序查询
    Set<String> categorys = jedis.zrange("category", 0, -1);
    List<Category> cs = null;
    //2.判断查询的集合是否为空
    if (categorys == null || categorys.size() == 0) {

        System.out.println("从数据库查询....");
        //3.如果为空,需要从数据库查询,在将数据存入 redis
        //3.1 从数据库查询
        cs = categoryDao.findAll();
        //3.2 将集合数据存储到 redis 中的 category 的 key
        for (int i = 0; i < cs.size(); i++) {

            jedis.zadd("category", cs.get(i).getCid(),
cs.get(i).getCname());
        }
    } else {
        System.out.println("从 redis 中查询.....");

        //4.如果不为空,将 set 的数据存入 list
        cs = new ArrayList<Category>();
        for (String name : categorys) {
            Category category = new Category();
            category.setCname(name);
            cs.add(category);
```

```
        }
    }


    return cs;
}
```

11 旅游线路的分页展示

点击了不同的分类后，将来看到的旅游线路不一样的。通过分析数据库表结构，发现，旅游线路表和分类表时一个多对一的关系



查询不同分类的旅游线路 sql

Select * from tab_route where cid = ?;

11.1 类别 id 的传递

Redis 中查询 score（cid）

```
public class CategoryServiceImpl implements CategoryService {

    private CategoryDao categoryDao = new CategoryDaoImpl();

    @Override
    public List<Category> findAll() {
```

```
        //1.从 redis 中查询
        //1.1 获取 jedis 客户端
        Jedis jedis = JedisUtil.getJedis();
        //1.2 可使用 sortedset 排序查询
        //Set<String> categorys = jedis.zrange("category", 0, -1);
        //1.3 查询 sortedset 中的分数(cid)和值(cname)
        Set<Tuple> categorys = jedis.zrangeWithScores("category",
0, -1);

        List<Category> cs = null;
        //2.判断查询的集合是否为空
        if (categorys == null || categorys.size() == 0) {

            System.out.println("从数据库查询....");
            //3.如果为空,需要从数据库查询,在将数据存入 redis
            //3.1 从数据库查询
            cs = categoryDao.findAll();
            //3.2 将集合数据存储到 redis 中的 category 的 key
            for (int i = 0; i < cs.size(); i++) {

                jedis.zadd("category", cs.get(i).getCid(),
cs.get(i).getCname());
            }
        } else {
            System.out.println("从 redis 中查询.....");

            //4.如果不为空,将 set 的数据存入 list
            cs = new ArrayList<Category>();
            for (Tuple tuple : categorys) {
                Category category = new Category();
                category.setCname(tuple.getElement());
                category.setCid((int)tuple.getScore());
                cs.add(category);

            }
        }


        return cs;
    }
}
```

页面传递 cid

header.html 传递 cid

```
var li = '<li><a
href="route_list.html?cid='+data[i].cid+'">'+data[i].cname+'</
a></li>';
```

获取 cid

```
$(function () {
   var search = location.search;
   //alert(search);//?id=5
    // 切割字符串，拿到第二个值
    var cid = search.split("=")[1];
});
```

11.2 根据 id 查询不同类别的旅游线路数据

分页展示旅游线路数据：

11.2.1 分析

11.2.2编码

1. 客户端代码编写

```
$(function () {
    var search = location.search;
        // 切割字符串，拿到第二个值
```

```
    var cid = search.split("=")[1];

    //当页码加载完成后，调用 load 方法，发送 ajax 请求加载数据
    load(cid);
});

function load(cid ,currentPage){
    //发送 ajax 请求，请求 route/pageQuery,传递 cid

$.get("route/pageQuery",{cid:cid,currentPage:currentPage},func
tion (pb) {
        //解析 pagebean 数据，展示到页面上

        //1.分页工具条数据展示
        //1.1 展示总页码和总记录数
        $("#totalPage").html(pb.totalPage);
        $("#totalCount").html(pb.totalCount);

                var lis = "";

        var fristPage = '<li onclick="javascipt:load('+cid+')"><a
href="javascript:void(0)">首页</a></li>';

        //计算上一页的页码
        var beforeNum =  pb.currentPage - 1;
        if(beforeNum <= 0){
            beforeNum = 1;
        }

        var beforePage = '<li
onclick="javascipt:load('+cid+','+beforeNum+')"
class="threeword"><a href="javascript:void(0)">上一页</a></li>';

        lis += fristPage;
        lis += beforePage;
        //1.2 展示分页页码
        /*
            1.一共展示 10 个页码，能够达到前 5 后 4 的效果
            2.如果前边不够 5 个，后边补齐 10 个
            3.如果后边不足 4 个，前边补齐 10 个
        */

        // 定义开始位置 begin,结束位置 end
        var begin; // 开始位置
```

```
        var end ; //  结束位置


        //1.要显示 10 个页码
        if(pb.totalPage < 10){
            //总页码不够 10 页

            begin = 1;
            end = pb.totalPage;
        }else{
            //总页码超过 10 页

            begin = pb.currentPage - 5 ;
            end = pb.currentPage + 4 ;

            //2.如果前边不够 5 个，后边补齐 10 个
            if(begin < 1){
                begin = 1;
                end = begin + 9;
            }

            //3.如果后边不足 4 个，前边补齐 10 个
            if(end > pb.totalPage){
                end = pb.totalPage;
                begin = end - 9 ;
            }
        }


        for (var i = begin; i <= end ; i++) {
            var li;
            //判断当前页码是否等于 i
            if(pb.currentPage == i){

                li = '<li class="curPage"
onclick="javascipt:load('+cid+','+i+')"><a
href="javascript:void(0)">'+i+'</a></li>';

            }else{
                //创建页码的 li
                li = '<li
onclick="javascipt:load('+cid+','+i+')"><a
href="javascript:void(0)">'+i+'</a></li>';
            }
```

```
            //拼接字符串
            lis += li;
        }



         var lastPage = '<li class="threeword"><a
href="javascript:;">末页</a></li>';
        var nextPage = '<li class="threeword"><a
href="javascript:;">下一页</a></li>';

        lis += nextPage;
        lis += lastPage;


        //将 lis 内容设置到 ul
        $("#pageNum").html(lis);




        //2.列表数据展示
        var route_lis = "";

        for (var i = 0; i < pb.list.length; i++) {
            //获取{rid:1,rname:"xxx"}
            var route = pb.list[i];

            var li = '<li>\n' +
                '                        <div class="img"><img
src="'+route.rimage+'" style="width: 299px;"></div>\n' +
                '                        <div class="text1">\n' +
                '
<p>'+route.rname+'</p>\n' +
                '                            <br/>\n' +
                '
<p>'+route.routeIntroduce+'</p>\n' +
                '                        </div>\n' +
                '                        <div class="price">\n' +
                '                            <p
class="price_num">\n' +
                '
```

```
<span>&yen;</span>\n' +
                '
<span>'+route.price+'</span>\n' +
                '                                    <span>起</span>\n'
+
                '                                </p>\n' +
                '                                <p><a
href="route_detail.html">查看详情</a></p>\n' +
                '                            </div>\n' +
                '                    </li>';
            route_lis += li;
        }
        $("#route").html(route_lis);

        //定位到页面顶部
        window.scrollTo(0,0);
    });

}
```

2. 服务器端代码编写
   a) 创建 PageBean 对象

```
public class PageBean<T> {

    private int totalCount;//总记录数
    private int totalPage;//总页数
    private int currentPage;//当前页码
    private int pageSize;//每页显示的条数

    private List<T> list;//每页显示的数据集合

    public int getTotalCount() {
        return totalCount;
    }

    public void setTotalCount(int totalCount) {
        this.totalCount = totalCount;
    }

    public int getTotalPage() {
        return totalPage;
    }
```

```
    public void setTotalPage(int totalPage) {
        this.totalPage = totalPage;
    }

    public int getCurrentPage() {
        return currentPage;
    }

    public void setCurrentPage(int currentPage) {
        this.currentPage = currentPage;
    }

    public int getPageSize() {
        return pageSize;
    }

    public void setPageSize(int pageSize) {
        this.pageSize = pageSize;
    }

    public List<T> getList() {
        return list;
    }

    public void setList(List<T> list) {
        this.list = list;
    }
}
```

b) RouteServlet

```
@WebServlet("/route/*")
public class RouteServlet extends BaseServlet {

    private RouteService routeService = new
RouteServiceImpl();

    /**
     * 分页查询
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void pageQuery(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
```

```
IOException {
        //1.接受参数
        String currentPageStr =
request.getParameter("currentPage");
        String pageSizeStr =
request.getParameter("pageSize");
        String cidStr = request.getParameter("cid");

        int cid = 0;//类别 id
        //2.处理参数
        if(cidStr != null && cidStr.length() > 0){
            cid = Integer.parseInt(cidStr);
        }
        int currentPage = 0;//当前页码，如果不传递，则默认为
第一页
        if(currentPageStr != null &&
currentPageStr.length() > 0){
            currentPage = Integer.parseInt(currentPageStr);
        }else{
            currentPage = 1;
        }

        int pageSize = 0;//每页显示条数，如果不传递，默认每页
显示 5 条记录
        if(pageSizeStr != null && pageSizeStr.length() > 0){
            pageSize = Integer.parseInt(pageSizeStr);
        }else{
            pageSize = 5;
        }

        //3. 调用 service 查询 PageBean 对象
        PageBean<Route> pb = routeService.pageQuery(cid,
currentPage, pageSize);

        //4. 将 pageBean 对象序列化为 json，返回
        writeValue(pb,response);

    }

}
```

c) RouteService

```
public class RouteServiceImpl implements RouteService {
    private RouteDao routeDao = new RouteDaoImpl();
```

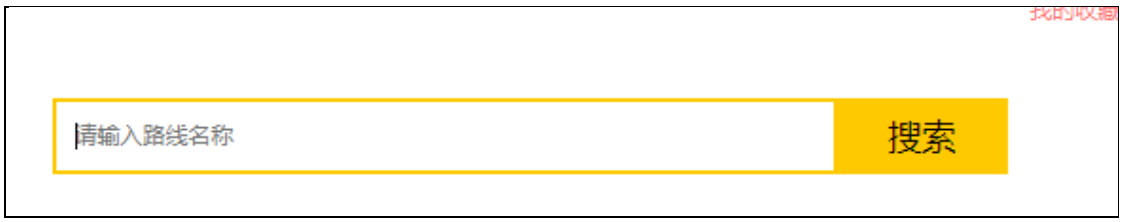
```
    @Override
    public PageBean<Route> pageQuery(int cid, int
currentPage, int pageSize) {
        //封装 PageBean
        PageBean<Route> pb = new PageBean<Route>();
        //设置当前页码
        pb.setCurrentPage(currentPage);
        //设置每页显示条数
        pb.setPageSize(pageSize);

        //设置总记录数
        int totalCount = routeDao.findTotalCount(cid);
        pb.setTotalCount(totalCount);
        //设置当前页显示的数据集合
        int start = (currentPage - 1) * pageSize;//开始的记
录数
        List<Route> list =
routeDao.findByPage(cid,start,pageSize);
        pb.setList(list);

        //设置总页数 = 总记录数/每页显示条数
        int totalPage = totalCount % pageSize == 0 ?
totalCount / pageSize :(totalCount / pageSize) + 1 ;
        pb.setTotalPage(totalPage);


        return pb;
    }
}
```

d) RouteDao

```
public class RouteDaoImpl implements RouteDao {
    private JdbcTemplate template = new
JdbcTemplate(JDBCUtils.getDataSource());

    @Override
    public int findTotalCount(int cid) {
        String sql = "select count(*) from tab_route where
cid = ?";
        return
template.queryForObject(sql,Integer.class,cid);
    }

    @Override
```

```
    public List<Route> findByPage(int cid, int start, int
pageSize) {
        String sql = "select * from tab_route where cid = ?
limit ? , ?";

        return template.query(sql,new
BeanPropertyRowMapper<Route>(Route.class),cid,start,pageS
ize);
    }
}
```

12 旅游线路名称查询



12.1 查询参数的传递

在 header.html 中

```
$("#search-button").click(function () {
    //线路名称
    var rname = $("#search_input").val();

    var cid = getParameter("cid");
    // 跳转路径 http://localhost/travel/route_list.html?cid=5，拼
接上 rname=xxx

location.href="http://localhost/travel/route_list.html?cid="+c
id+"&rname="+rname;
});
```

在 route_list.html

```
var cid = getParameter("cid");
 //获取 rname 的参数值
 var rname = getParameter("rname");
 //判断 rname 如果不为 null 或者""
 if(rname){
```

```
    //url 解码
    rname = window.decodeURIComponent(rname);
}
```

12.2 修改后台代码

Servlet

```
@WebServlet("/route/*")
public class RouteServlet extends BaseServlet {

    private RouteService routeService = new RouteServiceImpl();

    /**
     * 分页查询
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void pageQuery(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //1.接受参数
        String currentPageStr =
request.getParameter("currentPage");
        String pageSizeStr = request.getParameter("pageSize");
        String cidStr = request.getParameter("cid");

        //接受 rname 线路名称
        String rname = request.getParameter("rname");
        rname = new String(rname.getBytes("iso-8859-1"),"utf-8");


        int cid = 0;//类别 id
        //2.处理参数
        if(cidStr != null && cidStr.length() > 0){
            cid = Integer.parseInt(cidStr);
        }
        int currentPage = 0;//当前页码，如果不传递，则默认为第一页
        if(currentPageStr != null && currentPageStr.length() > 0){
            currentPage = Integer.parseInt(currentPageStr);
        }else{
            currentPage = 1;
```

```
        }

        int pageSize = 0;//每页显示条数，如果不传递，默认每页显示 5
条记录
        if(pageSizeStr != null && pageSizeStr.length() > 0){
            pageSize = Integer.parseInt(pageSizeStr);
        }else{
            pageSize = 5;
        }

        //3. 调用 service 查询 PageBean 对象
        PageBean<Route> pb = routeService.pageQuery(cid,
currentPage, pageSize,rname);

        //4. 将 pageBean 对象序列化为 json，返回
        writeValue(pb,response);

    }

}
```

Service

```
public PageBean<Route> pageQuery(int cid, int currentPage, int
pageSize,String rname ) {
    //封装 PageBean
    PageBean<Route> pb = new PageBean<Route>();
    //设置当前页码
    pb.setCurrentPage(currentPage);
    //设置每页显示条数
    pb.setPageSize(pageSize);

    //设置总记录数
    int totalCount = routeDao.findTotalCount(cid,rname);
    pb.setTotalCount(totalCount);
    //设置当前页显示的数据集合
    int start = (currentPage - 1) * pageSize;//开始的记录数
    List<Route> list =
routeDao.findByPage(cid,start,pageSize,rname);
    pb.setList(list);

    //设置总页数 = 总记录数/每页显示条数
    int totalPage = totalCount % pageSize == 0 ? totalCount /
pageSize :(totalCount / pageSize) + 1 ;
    pb.setTotalPage(totalPage);
```

```
    return pb;
}
```

Dao

```
@Override
public int findTotalCount(int cid,String rname) {
    //String sql = "select count(*) from tab_route where cid = ?";
    //1.定义 sql 模板
    String sql = "select count(*) from tab_route where 1=1 ";
    StringBuilder sb = new StringBuilder(sql);

    List params = new ArrayList();//条件们
    //2.判断参数是否有值
    if(cid != 0){
        sb.append( " and cid = ? ");

        params.add(cid);//添加? 对应的值
    }

    if(rname != null && rname.length() > 0){
        sb.append(" and rname like ? ");

        params.add("%"+rname+"%");
    }

    sql = sb.toString();


    return
template.queryForObject(sql,Integer.class,params.toArray());
}

@Override
public List<Route> findByPage(int cid, int start, int
pageSize,String rname) {
    //String sql = "select * from tab_route where cid = ? and rname
like ?  limit ? , ?";
    String sql = " select * from tab_route where 1 = 1 ";
    //1.定义 sql 模板
    StringBuilder sb = new StringBuilder(sql);
```

```
    List params = new ArrayList();//条件们
    //2.判断参数是否有值
    if(cid != 0){
        sb.append( " and cid = ? ");

        params.add(cid);//添加？对应的值
    }

    if(rname != null && rname.length() > 0){
        sb.append(" and rname like ? ");

        params.add("%"+rname+"%");
    }
    sb.append(" limit ? , ? ");//分页条件

    sql = sb.toString();

    params.add(start);
    params.add(pageSize);


    return template.query(sql,new
BeanPropertyRowMapper<Route>(Route.class),params.toArray());
}
```

12.3 修改前台代码

```
$(function () {
   /* var search = location.search;
    //alert(search);//?id=5
    // 切割字符串，拿到第二个值
    var cid = search.split("=")[1];*/
   //获取 cid 的参数值
   var cid = getParameter("cid");
    //获取 rname 的参数值
    var rname = getParameter("rname");
    //判断 rname 如果不为 null 或者""
    if(rname){
        //url 解码
        rname = window.decodeURIComponent(rname);
    }
```

```
    //当页码加载完成后，调用 load 方法，发送 ajax 请求加载数据
    load(cid,null,rname);
});

function load(cid ,currentPage,rname){
    //发送 ajax 请求，请求 route/pageQuery,传递 cid

$.get("route/pageQuery",{cid:cid,currentPage:currentPage,rname
:rname},function (pb) {
        //解析 pagebean 数据，展示到页面上

        //1.分页工具条数据展示
        //1.1 展示总页码和总记录数
        $("#totalPage").html(pb.totalPage);
        $("#totalCount").html(pb.totalCount);

        /*
                <li><a href="">首页</a></li>
                <li class="threeword"><a href="#">上一页</a></li>
                <li class="curPage"><a href="#">1</a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
                <li><a href="#">6</a></li>
                <li><a href="#">7</a></li>
                <li><a href="#">8</a></li>
                <li><a href="#">9</a></li>
                <li><a href="#">10</a></li>
                <li class="threeword"><a href="javascript:;">下一
页</a></li>
                <li class="threeword"><a href="javascript:;">末页
</a></li>


         */
        var lis = "";

        var fristPage = '<li
onclick="javascipt:load('+cid+',1,\''+rname+'\')"><a
href="javascript:void(0)">首页</a></li>';

        //计算上一页的页码
```

```
        var beforeNum =  pb.currentPage - 1;
        if(beforeNum <= 0){
            beforeNum = 1;
        }

        var beforePage = '<li
onclick="javascipt:load('+cid+','+beforeNum+',\''+rname+'\')"
class="threeword"><a href="javascript:void(0)">上一页</a></li>';

        lis += fristPage;
        lis += beforePage;
        //1.2 展示分页页码
        /*
            1.一共展示 10 个页码，能够达到前 5 后 4 的效果
            2.如果前边不够 5 个，后边补齐 10 个
            3.如果后边不足 4 个，前边补齐 10 个
        */

        // 定义开始位置 begin,结束位置 end
        var begin; // 开始位置
        var end ; //  结束位置


        //1.要显示 10 个页码
        if(pb.totalPage < 10){
            //总页码不够 10 页

            begin = 1;
            end = pb.totalPage;
        }else{
            //总页码超过 10 页

            begin = pb.currentPage - 5 ;
            end = pb.currentPage + 4 ;

            //2.如果前边不够 5 个，后边补齐 10 个
            if(begin < 1){
                begin = 1;
                end = begin + 9;
            }

            //3.如果后边不足 4 个，前边补齐 10 个
            if(end > pb.totalPage){
                end = pb.totalPage;
```

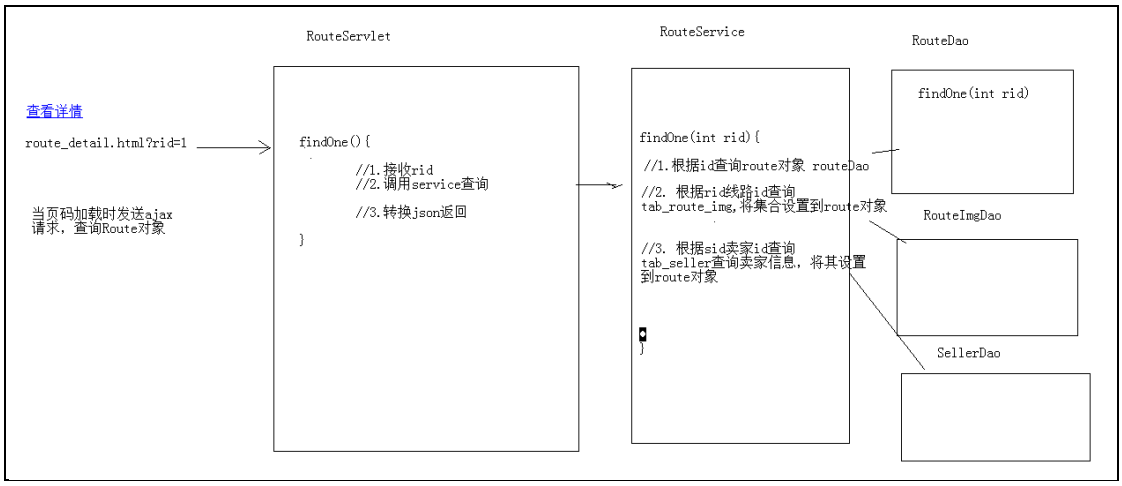
```
                begin = end - 9 ;
            }
        }


        for (var i = begin; i <= end ; i++) {
            var li;
            //判断当前页码是否等于 i
            if(pb.currentPage == i){

                li = '<li class="curPage"
onclick="javascipt:load('+cid+','+i+',\''+rname+'\')"><a
href="javascript:void(0)">'+i+'</a></li>';

            }else{
                //创建页码的 li
                li = '<li
onclick="javascipt:load('+cid+','+i+',\''+rname+'\')"><a
href="javascript:void(0)">'+i+'</a></li>';
            }
            //拼接字符串
            lis += li;
        }




       /* for (var i = 1; i <= pb.totalPage ; i++) {
            var li;
            //判断当前页码是否等于 i
            if(pb.currentPage == i){

                li = '<li class="curPage"
onclick="javascipt:load('+cid+','+i+')"><a
href="javascript:void(0)">'+i+'</a></li>';

            }else{
                //创建页码的 li
                li = '<li
onclick="javascipt:load('+cid+','+i+')"><a
href="javascript:void(0)">'+i+'</a></li>';
            }
            //拼接字符串
```

```
            lis += li;
        }*/
        var lastPage = '<li class="threeword"><a
href="javascript:;">末页</a></li>';
        var nextPage = '<li class="threeword"><a
href="javascript:;">下一页</a></li>';

        lis += nextPage;
        lis += lastPage;


        //将 lis 内容设置到 ul
        $("#pageNum").html(lis);



        /*
            <li>
                <div class="img"><img
src="images/04-search_03.jpg" alt=""></div>
                <div class="text1">
                    <p>【减 100 元 含除夕/春节出发】广州增城三英温泉度
假酒店/自由行套票</p>
                    <br/>
                    <p>1-2 月出发,网付立享￥1099/2 人起!爆款位置有限,
抢完即止! </p>
                </div>
                <div class="price">
                    <p class="price_num">
                        <span>&yen;</span>
                        <span>299</span>
                        <span>起</span>
                    </p>
                    <p><a href="route_detail.html">查看详情
</a></p>
                </div>
            </li>

         */

        //2.列表数据展示
        var route_lis = "";

        for (var i = 0; i < pb.list.length; i++) {
```

```
            //获取{rid:1,rname:"xxx"}
            var route = pb.list[i];

            var li = '<li>\n' +
                '                        <div class="img"><img
src="'+route.rimage+'" style="width: 299px;"></div>\n' +
                '                        <div class="text1">\n' +
                '
<p>'+route.rname+'</p>\n' +
                '                            <br/>\n' +
                '
<p>'+route.routeIntroduce+'</p>\n' +
                '                        </div>\n' +
                '                        <div class="price">\n' +
                '                            <p
class="price_num">\n' +
                '
<span>&yen;</span>\n' +
                '
<span>'+route.price+'</span>\n' +
                '                                <span>起</span>\n'
+
                '                            </p>\n' +
                '                            <p><a
href="route_detail.html">查看详情</a></p>\n' +
                '                        </div>\n' +
                '                    </li>';
            route_lis += li;
        }
        $("#route").html(route_lis);

        //定位到页面顶部
        window.scrollTo(0,0);
    });

}
```

13　旅游线路的详情展示

13.1　分析



13.2　代码实现

13.2.1后台代码

Servlet

```
/**
 * 根据 id 查询一个旅游线路的详细信息
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
```

```
 */
public void findOne(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {

    //1.接收 id
    String rid = request.getParameter("rid");
    //2.调用 service 查询 route 对象
    Route route = routeService.findOne(rid);
    //3.转为 json 写回客户端
    writeValue(route,response);
}
```

Service

```
@Override
public Route findOne(String rid) {
    //1.根据 id 去 route 表中查询 route 对象
    Route route = routeDao.findOne(Integer.parseInt(rid));

    //2.根据 route 的 id 查询图片集合信息
    List<RouteImg> routeImgList =
routeImgDao.findByRid(route.getRid());
    //2.2 将集合设置到 route 对象
    route.setRouteImgList(routeImgList);
    //3.根据 route 的 sid（商家 id）查询商家对象
    Seller seller = sellerDao.findById(route.getSid());
    route.setSeller(seller);

    return route;
}
```

SellerDao

```
public class SellerDaoImpl implements SellerDao {

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



    @Override
    public Seller findById(int id) {
```

```
        String sql = "select * from tab_seller where sid = ? ";
        return template.queryForObject(sql,new
BeanPropertyRowMapper<Seller>(Seller.class),id);
    }
}
```

routeDao

```
@Override
public Route findOne(int rid) {
    String sql = "select * from tab_route where rid = ?";
    return template.queryForObject(sql,new
BeanPropertyRowMapper<Route>(Route.class),rid);
}
```

RouteImgDao

```
public class RouteImgDaoImpl implements RouteImgDao {

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


    @Override
    public List<RouteImg> findByRid(int rid) {
        String sql = "select * from tab_route_img where rid = ? ";
        return template.query(sql,new
BeanPropertyRowMapper<RouteImg>(RouteImg.class),rid);
    }
}
```

13.2.2前台代码

Route_detail.html 中加载后

1. 获取 rid
2. 发送 ajax 请求，获取 route 对象
3. 解析对象的数据

```
//1.获取 rid
var rid = getParameter("rid");

//2.发送请求请求 route/findOne
 $.get("route/findOne",{rid:rid},function (route) {
     //3.解析数据填充 html
```

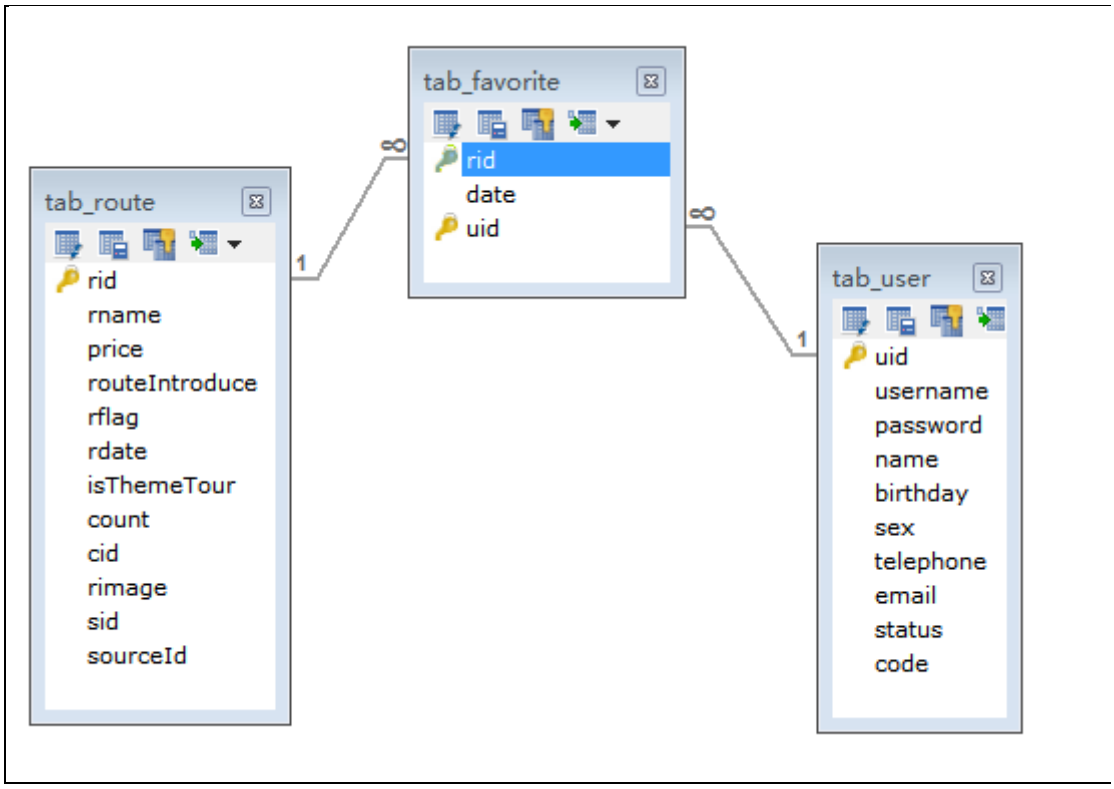
```
    $("#rname").html(route.rname);
    $("#routeIntroduce").html(route.routeIntroduce);
    $("#price").html("¥"+route.price);
    $("#sname").html(route.seller.sname);
    $("#consphone").html(route.seller.consphone);
    $("#address").html(route.seller.address);

    //图片展示

    var ddstr = '<a class="up_img up_img_disable"></a>';

    //遍历 routeImgList
    for (var i = 0; i < route.routeImgList.length; i++)
{
        var astr ;
        if(i >= 4){
            astr = '<a title="" class="little_img"
data-bigpic="'+route.routeImgList[i].bigPic+'"
style="display:none;">\n' +
                '                        <img
src="'+route.routeImgList[i].smallPic+'">\n' +
                '                    </a>';
        }else{
            astr = '<a title="" class="little_img"
data-bigpic="'+route.routeImgList[i].bigPic+'">\n' +
                '                        <img
src="'+route.routeImgList[i].smallPic+'">\n' +
                '                    </a>';
        }


        ddstr += astr;
    }
    ddstr+='<a class="down_img down_img_disable"
style="margin-bottom: 0;"></a>';

    $("#dd").html(ddstr);

    //图片展示和切换代码调用
    goImg();
 });
```

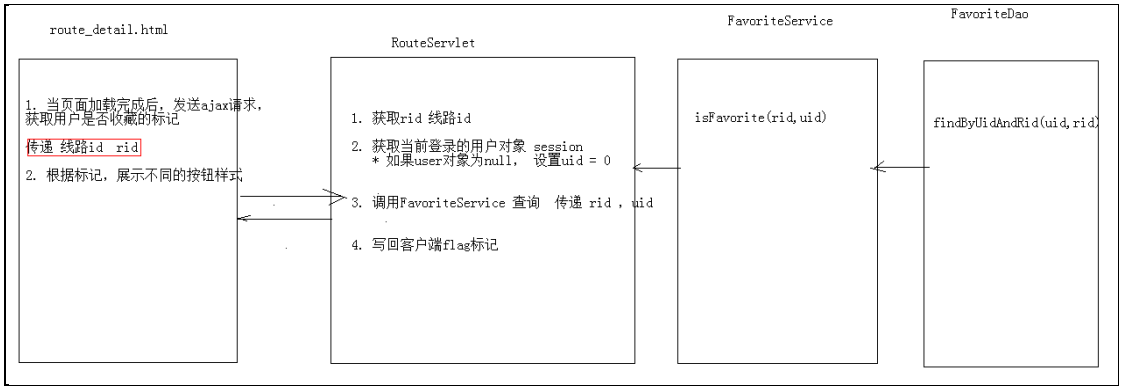
14　旅游线路收藏功能

14.1 分析

14.1.1 判断当前登录用户是否收藏过该线路

当页面加载完成后，发送 ajax 请求，获取用户是否收藏的标记
根据标记，展示不同的按钮样式





14.2 编写代码

14.2.1 后台代码

RouteServlet：

```
public void isFavorite(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    //1. 获取线路 id
    String rid = request.getParameter("rid");

    //2. 获取当前登录的用户 user
    User user = (User) request.getSession().getAttribute("user");
    int uid;//用户 id
    if(user == null){
        //用户尚未登录
        uid = 0;
    }else{
        //用户已经登录
        uid = user.getUid();
    }

    //3. 调用 FavoriteService 查询是否收藏
    boolean flag = favoriteService.isFavorite(rid, uid);

    //4. 写回客户端
    writeValue(flag,response);
}
```

FavoriteService

```
@Override
public boolean isFavorite(String rid, int uid) {

    Favorite favorite =
favoriteDao.findByRidAndUid(Integer.parseInt(rid), uid);

    return favorite != null;//如果对象有值，则为 true，反之，则为 false
}
```

FavoriteDao

```
@Override
public Favorite findByRidAndUid(int rid, int uid) {
    Favorite favorite = null;
    try {
        String sql = " select * from tab_favorite where rid = ? and
uid = ?";
        favorite = template.queryForObject(sql, new
BeanPropertyRowMapper<Favorite>(Favorite.class), rid, uid);
    } catch (DataAccessException e) {
        e.printStackTrace();
    }
    return favorite;
}
```

14.2.2 前台代码

route_detail.html

```
$(function () {
   // 发送请求，判断用户是否收藏过该线路
    var rid = getParameter("rid");
    $.get("route/isFavorite",{rid:rid},function (flag) {
        if(flag){
            // 用户已经收藏过
            //<a  class="btn already" disabled="disabled">
            //设置收藏按钮的样式
            $("#favorite").addClass("already");
            $("#favorite").prop("disabled",disabled);



        }else{
            // 用户没有收藏
        }
    });
```

14.3 收藏次数的动态展示

前台：

```
//设置收藏次数
$("#favoriteCount").html("已收藏"+route.count+"次");
```

后台：

RouteService

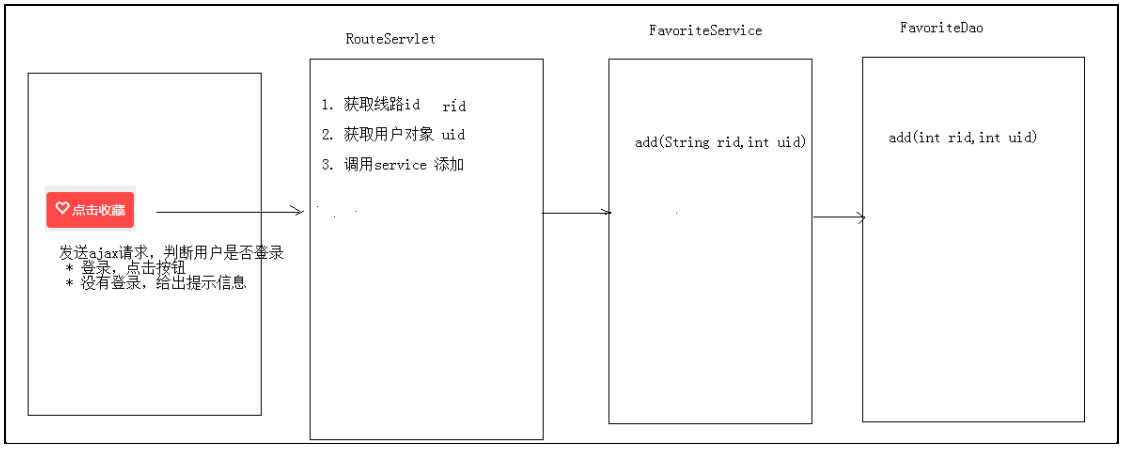
```
//4. 查询收藏次数
int count = favoriteDao.findCountByRid(route.getRid());
route.setCount(count);
```

FavoriteDao

```
@Override
public int findCountByRid(int rid) {
    String sql = "SELECT COUNT(*) FROM tab_favorite WHERE rid = ?";

    return template.queryForObject(sql,Integer.class,rid);
}
```

14.4 点击按钮收藏线路

14.4.1 分析：



14.4.2 编码

前台代码

```
        $(function () {
    // 发送请求，判断用户是否收藏过该线路
    var rid = getParameter("rid");
    $.get("route/isFavorite",{rid:rid},function (flag) {
        if(flag){
            // 用户已经收藏过
            //<a  class="btn already" disabled="disabled">
            //设置收藏按钮的样式
            $("#favorite").addClass("already");
            $("#favorite").attr("disabled","disabled");

            //删除按钮的点击事件
            $("#favorite").removeAttr("onclick");
        }else{
            // 用户没有收藏
        }
    });



});

//点击收藏按钮触发的方法
function addFavorite(){
    var rid = getParameter("rid");
    //1. 判断用户是否登录
    $.get("user/findOne",{},function (user) {
       if(user){
           //用户登录了
            //添加功能
            $.get("route/addFavorite",{rid:rid},function () {

                //代码刷新页面
                location.reload();
            });

       }else{
           //用户没有登录
           alert("您尚未登录，请登录");
           location.href="http://localhost/travel/login.html";
       }
    })
}
```

后台代码

RouteServlet

```
public void addFavorite(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    //1. 获取线路 rid
    String rid = request.getParameter("rid");
    //2. 获取当前登录的用户
    User user = (User) request.getSession().getAttribute("user");
    int uid;//用户 id
    if(user == null){
        //用户尚未登录
        return ;
    }else{
        //用户已经登录
        uid = user.getUid();
    }


    //3. 调用 service 添加
    favoriteService.add(rid,uid);

}
```

FavoriteService

```
@Override
public void add(String rid, int uid) {
    favoriteDao.add(Integer.parseInt(rid),uid);
}
```

FavoriteDao

```
@Override
public void add(int rid, int uid) {
    String sql = "insert into tab_favorite values(?,?,?)";

    template.update(sql,rid,new Date(),uid);
}
```