

day43-项目项目第三天

学习目标

- 1. 能够完成头部搜索旅游线路分页展现数据案例
- 2. 能够完成查看旅游线路详情案例
- 3. 能够完成添加收藏案例
- 4. 能够进行面向接口编程

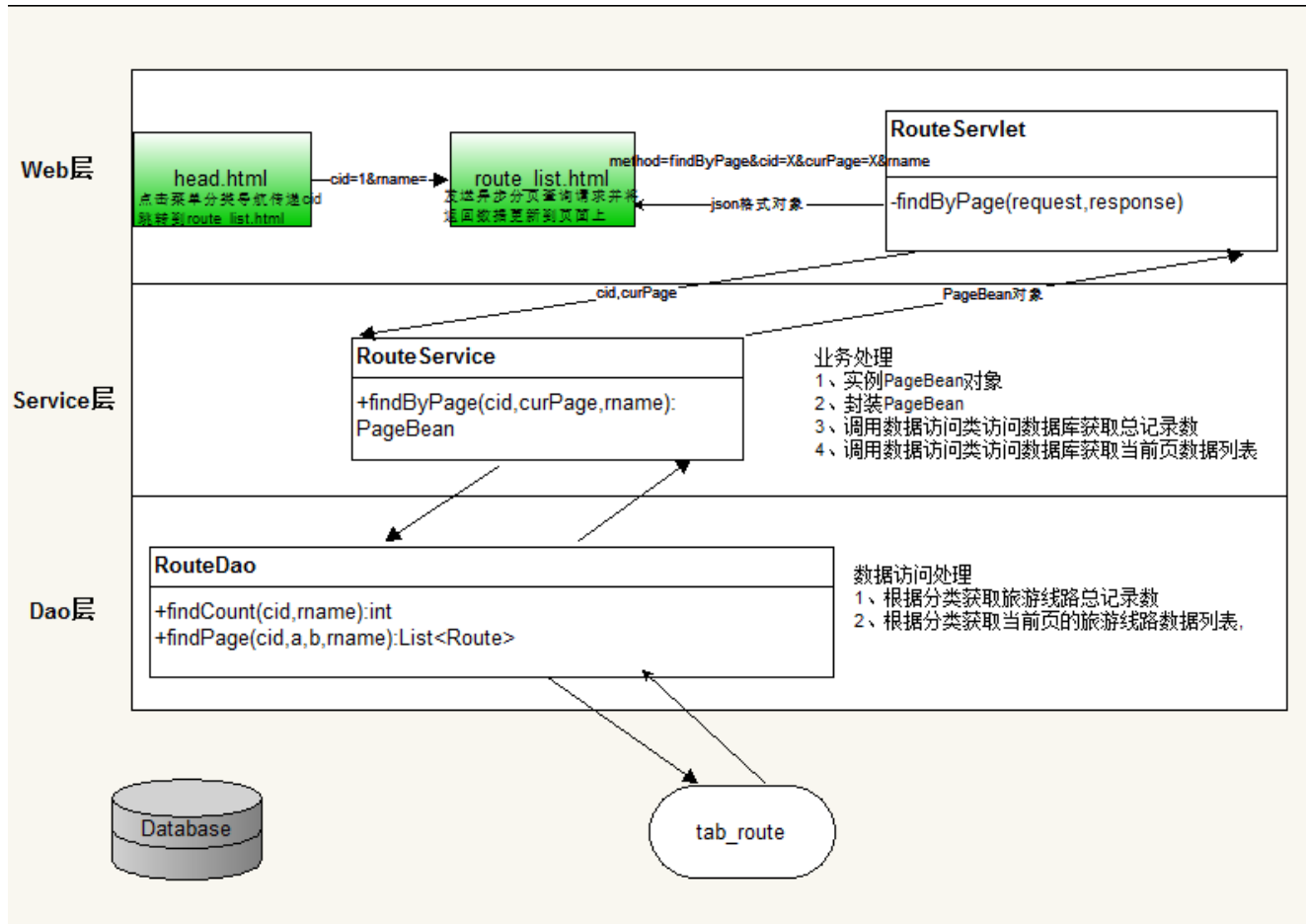
案例一 头部搜索旅游线路分页展现数据

一,案例需求

点击网页头部搜索框输入旅游线路名称，点击搜索安全进行模糊搜索，符合条件的结果进行分页展现数据列表



二,思路分析



1. header.html代码，点击搜索框提交搜索内容跳转到route_list.html并传递cid和搜索内容
2. route_list.html代码，获取搜索内容数据，并提交到分页查询数据请求中，保证每个分页请求都要增加搜索内容提交请求参数。
3. RouteServlet.java代码，获取到搜索内容数据，并查询分页数据方法增加搜索内容参数传递。
4. RouteService.java代码，获取总记录数据和当前页数据列表方法中增加搜索内容参数传递。
5. RouteDao.java代码，获取总记录数增加搜索内容数据条件过滤，获取当前页数据列表增加搜索内容数据条件过滤。

三,代码实现

- header.jsp

```
<div class="search">
  <input name="" id="rnameId" type="text" placeholder="请输入路线名称" class="search_input" autocomplete="off">
  <a href="javascript:search();" class="search-button">搜索</a>
</div>
```

```

<script>
    //创建函数响应搜索按钮的点击
    function search() {

        //获得用户输入的路线名
        var rname = $("#rnameId").val();
        //获得当前类别cid
        var cid = getParameter("cid");
        alert("cid="+cid);
        //请求route_list.html
        location.href = "route_list.html?cid="+cid+"&rname="+rname+"&curPage=1";
    }

</script>

```

- 修改route_list.html添加rname参数

```

//获得rname
var rname = getParameter("rname");
if (rname != null && rname != "") {
    //进行解码
    rname = decodeURI(rname);
}

//获得header页面, 传递过来的请求参数(cid, curPage)
var cid = getParameter("cid");
var curPage = getParameter("curPage");
//发送Ajax请求, 分页查询
$.post("route", {method: "findByPage", curPage: curPage, cid: cid, rname: rname}, function (result) {
    if (result.flag) {

```

- 修改RouteServlet.java

```

/**
 * 根据类别分页查询旅游路线
 * @param request
 * @param response
 */
public void findByPage(HttpServletRequest request, HttpServletResponse response) throws
IOException {
    String data = null;
    ResultInfo resultInfo = null;
    ObjectMapper objectMapper = new ObjectMapper();

    try {
        //1. 获得请求参数(cid,curPage,rname)
        int cid = Integer.parseInt(request.getParameter("cid"));
        int curPage = Integer.parseInt(request.getParameter("curPage"));
        String rname = request.getParameter("rname");

        //2. 调用业务
        PageBean<Route> pageBean = routeService.findByPage(cid,curPage,rname);
        //3.响应数据
        resultInfo = new ResultInfo(true,pageBean,"分页查询成功!");
    } catch (Exception e) {
        e.printStackTrace();
        resultInfo = new ResultInfo(false,null,"分页查询失败!");
    }finally {
        data = objectMapper.writeValueAsString(resultInfo);
        System.out.println("data=" + data);
        response.getWriter().print(data);
    }
}

```

- 修改RouteService.java

```

/**
 * 分页查询数据
 * @param cid
 * @param curPage
 * @param rname
 * @return
 */
public PageBean<Route> findByPage(int cid, int curPage, String rname) throws Exception {
    int curSize = Constants.ROUTE_CUR_SIZE;
    int count = routeDao.findCount(cid,rname);
    int sumPage = 0;
    if(count % curSize == 0){
        sumPage = count / curSize;
    }else{
        sumPage = count / curSize + 1;
    }

    int b = Constants.ROUTE_CUR_SIZE;
    int a = (curPage -1)*b;
    List<Route> list = routeDao.findPage(cid,a,b,rname);

    PageBean<Route> pageBean = new PageBean<>();
    pageBean.setCurPage(curPage);
    pageBean.setCurSize(curSize);
    pageBean.setCount(count);
    pageBean.setSumPage(sumPage);
    pageBean.setList(list);
    return pageBean;
}

```

- 修改RouteDao.java

```

/**
 * 统计当前类别下的旅游线路总数
 * @param cid
 * @param rname
 * @return
 */
public int findCount(int cid, String rname) throws Exception {
    String sql="SELECT count(*) FROM tab_route WHERE rflag='1'";

    List<Object> params = new ArrayList<Object>();
    if(cid != 0){
        sql += " AND cid = ?";
        params.add(cid);
    }

    if (rname != null && !"".equals(rname)){
        sql+=" AND rname LIKE ?";
        params.add("%"+rname+"%");
    }

    Long n = jdbcTemplate.queryForObject(sql, Long.class,params.toArray());
    return n.intValue();
}

/**
 * 查询当前类别下的旅游线路列表
 * @param cid
 * @param a
 * @param b
 * @param rname
 * @return
 */
public List<Route> findPage(int cid, int a, int b, String rname) throws Exception{
    String sql="SELECT * FROM tab_route WHERE rflag='1'";
    List<Object> params = new ArrayList<Object>();
    if(cid != 0){
        sql += " AND cid = ?";
        params.add(cid);
    }

    if (rname != null && !"".equals(rname)){
        sql+=" AND rname LIKE ?";
        params.add("%"+rname+"%");
    }
    sql += " limit ?,?";
    params.add(a);
    params.add(b);

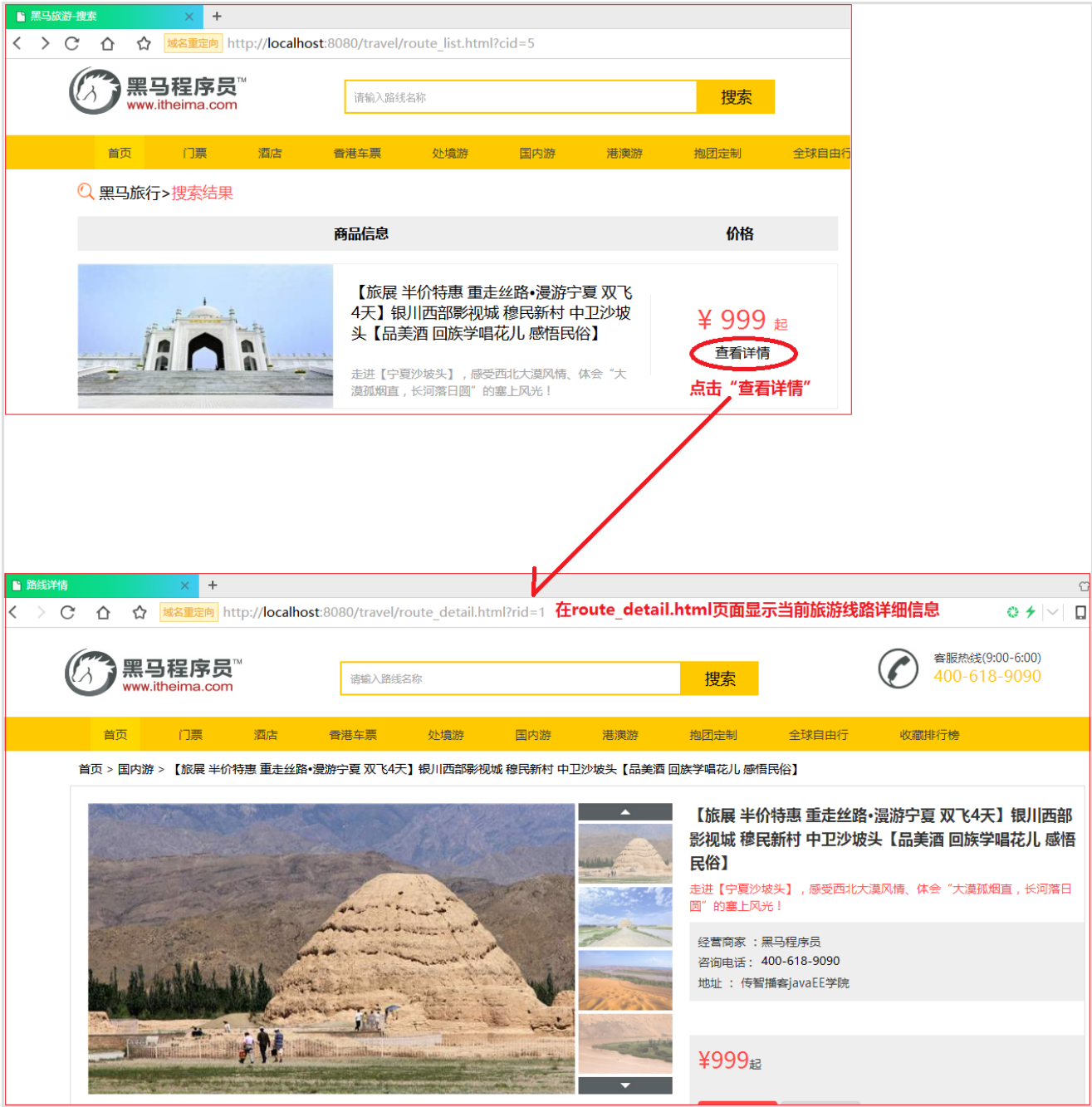
    return jdbcTemplate.query(sql,new BeanPropertyRowMapper<>
(Route.class),params.toArray());
}

```

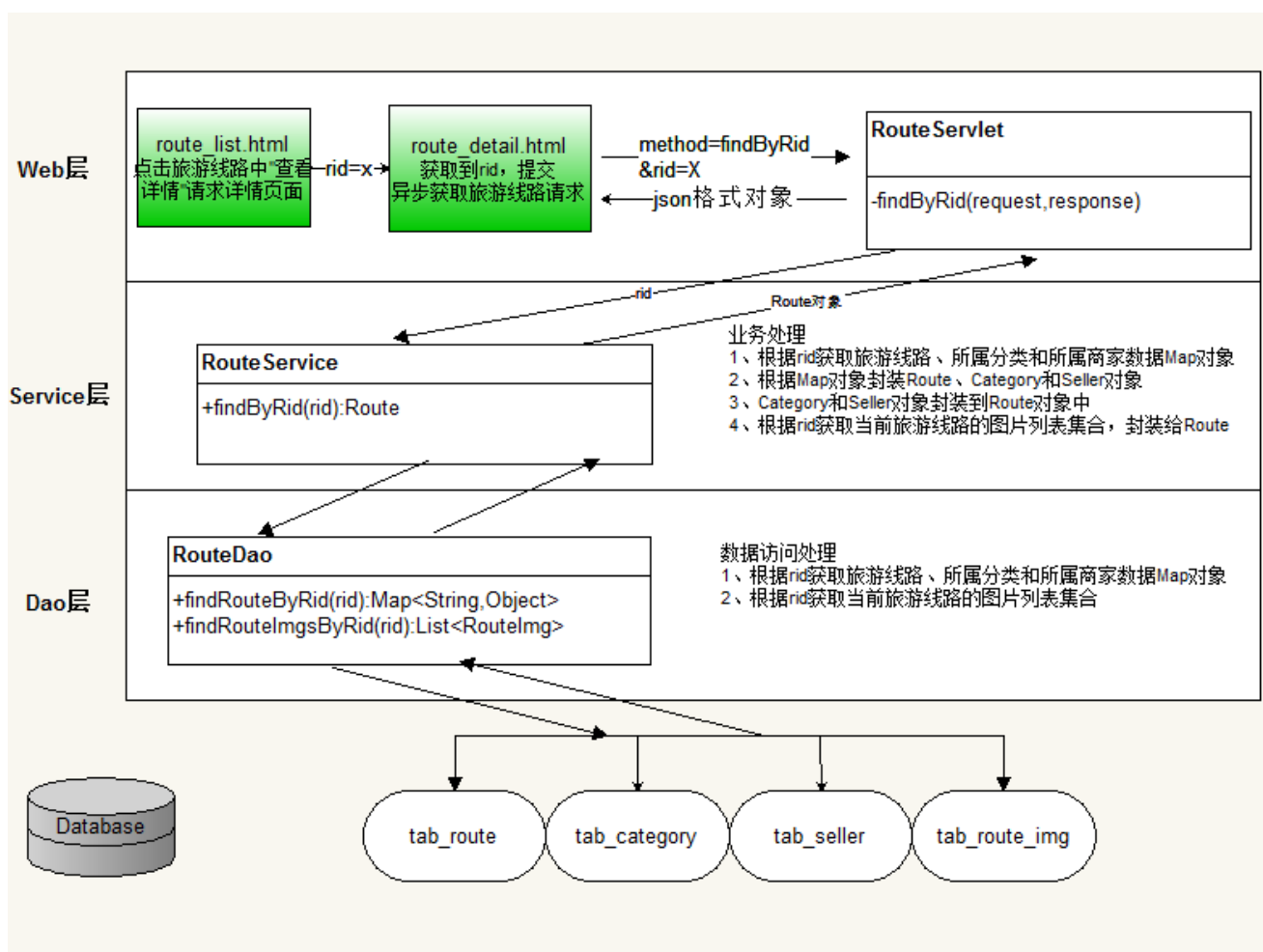
案例二：查看旅游线路详情

一,案例需求

点击route_list.html搜索列表中旅游线路的“查看详情”进入route_detail.html旅游线路详情页面并显示相关数据



二, 思路分析



1. 在route_list.html修改 [查看详情](#) 连接, 携带 rid
2. 在 route_detail.html页面:

```
//1. 获得rid
//2. 发送Ajax请求, 获得当前旅游线路的数据 $.post("route",{method:"findByRid",rid:rid}....)
//3. 解析数据, 填充页面
```

3. 在RouteServlet里面创建findByRid()方法, 在这个方法里面:

获得请求参数(rid)

调用业务, 获得Route对象

封装ResultInfo, 响应数据

4. 在RouteService里面创建findByRid()

调用Dao, 封装Route

5. 在RouteDao 查询数据库

三,代码实现

- route_list.html


```

$(list).each(function (i,obj) {
    listHtml+="- <n" +
    "
    <div class='img'><img src='"+obj.rimage+"' width='299' height='169' alt='\"
    <div class='text1'>\n" +
    <p'+obj.rname.substring(0,20)+'...</p>\n" +
    <br/>\n" +
    <p'+obj.routeIntroduce.substring(0,60)+'...</p>\n" +
    </div>\n" +
    <div class='price'>\n" +
    <p class='price_num'>\n" +
    <span>&yen;</span>\n" +
    <span>'+obj.price+'</span>\n" +
    <span>起</span>\n" +
    </p>\n" +
    <p><a href='route_detail.html?rid='+obj.rid+'>查看详情</a></p>\n" +
    </div>\n" +
    </li>";
});
$("#listUId").html(listHtml);

```

- route_detail.html

```

<script>
$(function() {
    //1. 获得rid
    var rid = getParameter("rid");
    //2. 发送Ajax请求服务器
    $.post("route",{method:"findByRid",rid:rid},function (result) {
        if (result.flag){

            var route = result.data;

            //a 更新面包屑
            $(".bread_box").html("<a href=\\\"/\\\">首页</a>\\n" +
                "            <span> &gt;</span>\\n" +
                "            <a href=\\\"#\\\">"+route.category.cname+"</a><span> &gt;</span>\\n"
+
                "            <a href=\\\"#\\\">"+route.rname+"</a>");

            //b.更新标题
            $(".pros_title").html(route.rname);

            //c.更新介绍
            $(".hot").html(route.routeIntroduce);

            //d.更新商家信息
            $(".pros_other").html(" <p>经营商家 : "+route.seller.sname+"</p>\\n" +
                "                <p>咨询电话 : "+route.seller.consphone+"</p>\\n" +
                "                <p>地址 : "+route.seller.address+"</p> ");

            //e更新价格
            $(".price strong").html("¥"+route.price);

            //f更新收藏次数
            $("#collectSpanId").html("已收藏"+route.count+"次");

            //e.更新图片
            var routeImgList = route.routeImgList;
            //大图
            var imgListHtml = " <dt>\\n" +
                "                <img alt=\\\"\" class=\\\"big_img\\\"
src=\\\""+routeImgList[0].bigPic+"\\\">\\n" +
                "                </dt>";

            //小图
            //小图开始
            imgListHtml += "<dd><a class=\\\"up_img up_img_disable\\\"></a>";

            //中间遍历
            for(var i = 0; i < routeImgList.length;i++){
                var routeImg = routeImgList[i];

                if (i < 4){

                    imgListHtml+= " <a title=\\\"\" class=\\\"little_img cur_img\\\" data-

```

```

bigpic=\""+routeImg.bigPic+"\">\n" +
        "
        "                <img src=\""+routeImg.smallPic+"\">\n" +
        "                </a>";
    }else{
        imgListHtml+=" <a title=\"\" style=\"display:none;\" class=\"little_img
cur_img\" data-bigpic=\""+routeImg.bigPic+"\">\n" +
        "
        "                <img src=\""+routeImg.smallPic+"\">\n" +
        "                </a>";
    }

}

//小图结束
imgListHtml+="<a class=\"down_img down_img_disable\" style=\"margin-bottom: 0;\">
</a></dd>";

$(".prosum_left").html(imgListHtml);

imgGo();

}else{
    alert(result.msg);
}
}, "json");

});

function imgGo() {

    //焦点图效果
    //点击图片切换图片
    $('.little_img').on('mousemove', function() {
        $('.little_img').removeClass('cur_img');
        var big_pic = $(this).data('bigpic');
        $('.big_img').attr('src', big_pic);
        $(this).addClass('cur_img');
    });
    //上下切换
    var picindex = 0;
    var nextindex = 4;
    $('.down_img').on('click', function(){
        var num = $('.little_img').length;
        if((nextindex + 1) <= num){
            $('.little_img:eq('+picindex+')').hide();
            $('.little_img:eq('+nextindex+')').show();
            picindex = picindex + 1;
            nextindex = nextindex + 1;
        }
    });
}

```

```

});
$('.up_img').on('click',function(){
    var num = $('.little_img').length;
    if(picindex > 0){
        $('.little_img:eq('+nextindex-1+')').hide();
        $('.little_img:eq('+picindex-1+')').show();
        picindex = picindex - 1;
        nextindex = nextindex - 1;
    }
});
//自动播放
// var timer = setInterval("auto_play()", 5000);

}

//自动轮播方法
function auto_play() {
    var cur_index = $('.prosum_left dd').find('a.cur_img').index();
    cur_index = cur_index - 1;
    var num = $('.little_img').length;
    var max_index = 3;
    if ((num - 1) < 3) {
        max_index = num - 1;
    }
    if (cur_index < max_index) {
        var next_index = cur_index + 1;
        var big_pic = $('.little_img:eq(' + next_index + ')').data('bigpic');
        $('.little_img').removeClass('cur_img');
        $('.little_img:eq(' + next_index + ')').addClass('cur_img');
        $('.big_img').attr('src', big_pic);
    } else {
        var big_pic = $('.little_img:eq(0)').data('bigpic');
        $('.little_img').removeClass('cur_img');
        $('.little_img:eq(0)').addClass('cur_img');
        $('.big_img').attr('src', big_pic);
    }
}
}
</script>

```

- RouteServiceProvider.java

```

/**
 * 根据rid查询旅游线路详情
 * @param request
 * @param response
 */
public void findByRid(HttpServletRequest request, HttpServletResponse response) throws
IOException {
    String data = null;
    ResultInfo resultInfo = null;
    ObjectMapper objectMapper = new ObjectMapper();
    try {
        //1. 获得rid
        String rid = request.getParameter("rid");
        //2. 调用业务
        Route route = routeService.findByRid(rid);
        resultInfo = new ResultInfo(true, route, "旅游线路详情查询成功!");

    } catch (Exception e) {
        e.printStackTrace();
        resultInfo = new ResultInfo(false, null, "旅游线路详情查询失败!");
    } finally {
        data = objectMapper.writeValueAsString(resultInfo);
        System.out.println("data=" + data);
        response.getWriter().print(data);
    }
}

```

- RouteService.java

```

/**
 * 根据rid查询旅游线路详情
 */
public Route findById(String rid) throws Exception{

    //1.根据rid查询route(包含商家,类别)
    Map <String,Object> map = routeDao.findRouteByRid(rid);
    //a. 封装route
    Route route = new Route();
    BeanUtils.populate(route,map);
    //b.封装所属分类对象
    Category category = new Category();//所属分类对象
    BeanUtils.populate(category,map);
    //c.封装所属商家对象
    Seller seller = new Seller();
    BeanUtils.populate(seller,map);

    route.setSeller(seller);
    route.setCategory(category);

    //2.根据rid查询旅游详情图片
    List<RouteImg> routeImgList = routeDao.findRouteImgsByRid(rid);
    route.setRouteImgList(routeImgList);
    return route;
}

```

- RouteDao.java

```

/**
 * 根据rid查询旅游线路(包含商家,类别)
 * @param rid
 * @return
 */
public Map<String,Object> findRouteByRid(String rid)throws Exception{
    String sql = "SELECT * FROM tab_route r, tab_category c, tab_seller s WHERE r.cid = c.cid
AND r.sid = s.sid AND r.rflag = '1' AND r.rid = ?";
    Map<String, Object> map = jdbcTemplate.queryForMap(sql, rid);
    return map;

}

/**
 * 根据rid查询旅游详情图片
 * @param rid
 * @return
 */
public List<RouteImg> findRouteImgsByRid(String rid) throws Exception {
    String sql="SELECT * FROM tab_route_img WHERE rid=?";
    return jdbcTemplate.query(sql, new BeanPropertyRowMapper<RouteImg>(RouteImg.class),rid);
}

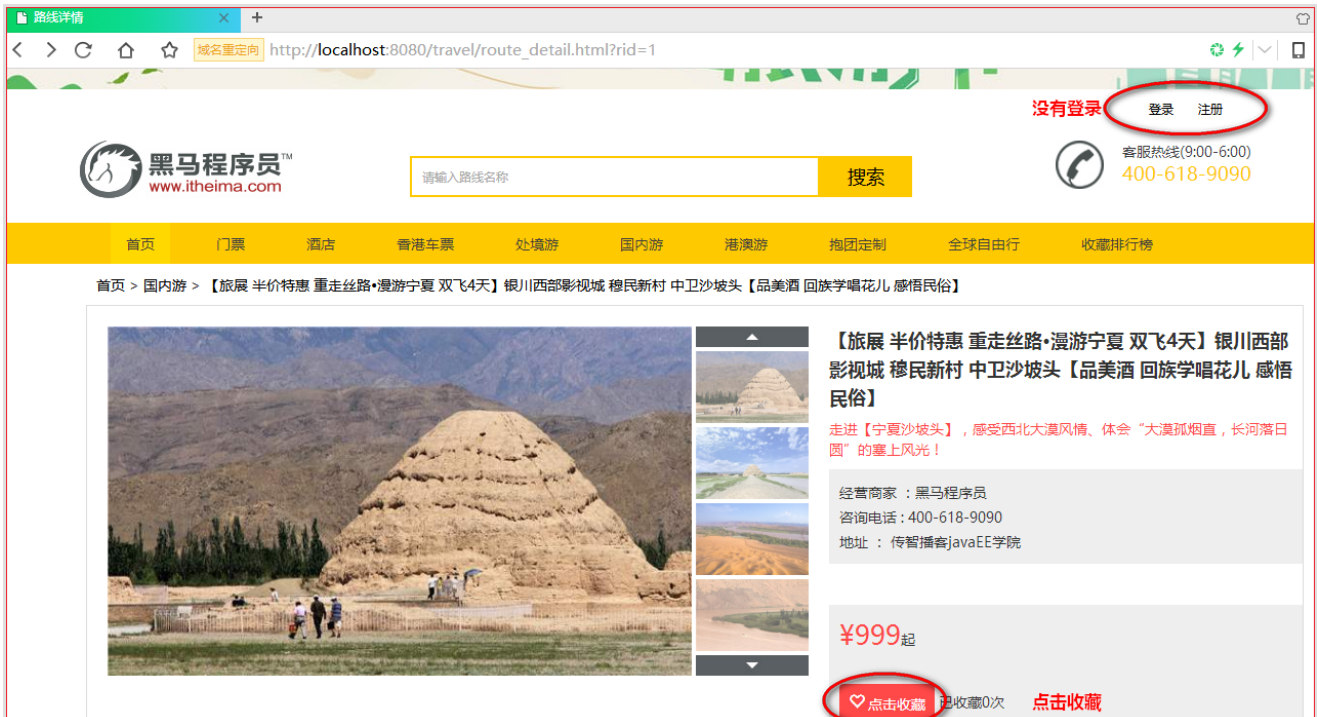
```

案例三-添加收藏

一、案例需求

用户在旅游线路详情页面点击“添加收藏”可以进行收藏该旅游线路，注意只有已经登录的用户才可以收藏当前旅游线路。此功能需要实现2个效果，第1个效果在旅游详情页面加载时根据是否已被收藏显示“点击收藏”是否可编辑；第2个效果是登录用户收藏当前旅游线路成功后的效果。

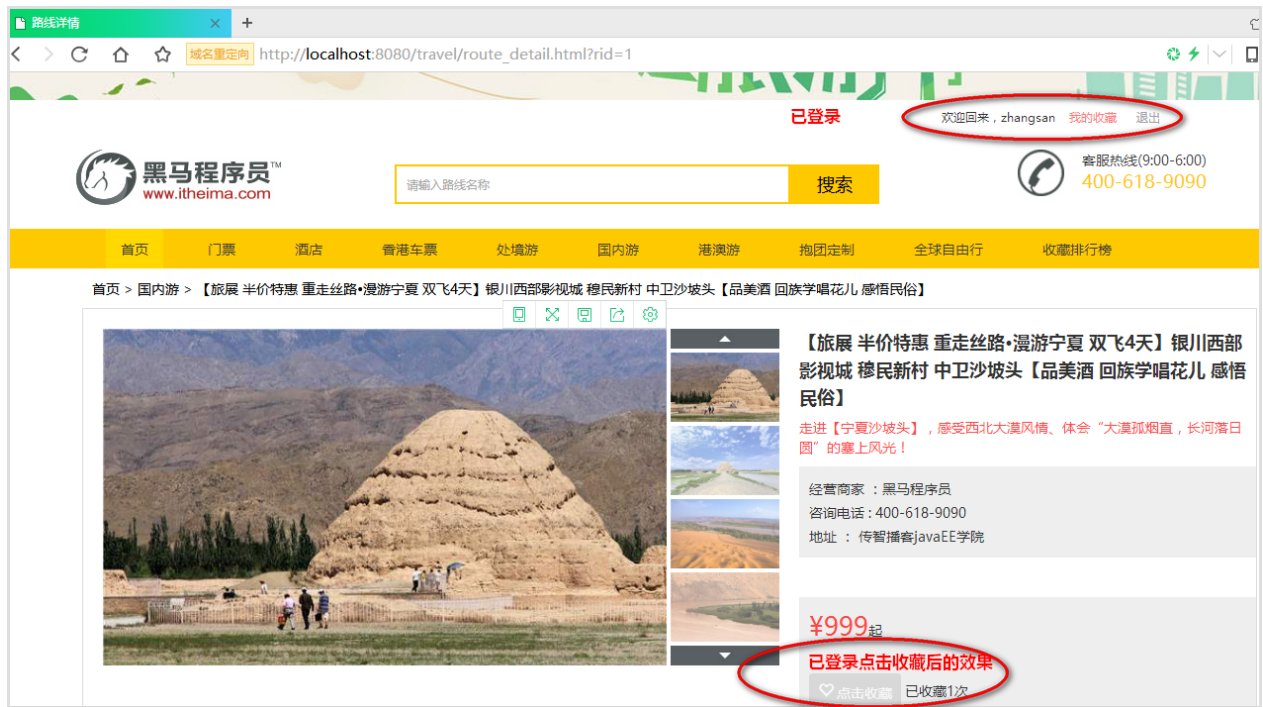
- 没有登录,显示可编辑状态, 点击“点击收藏”跳转到登录页面



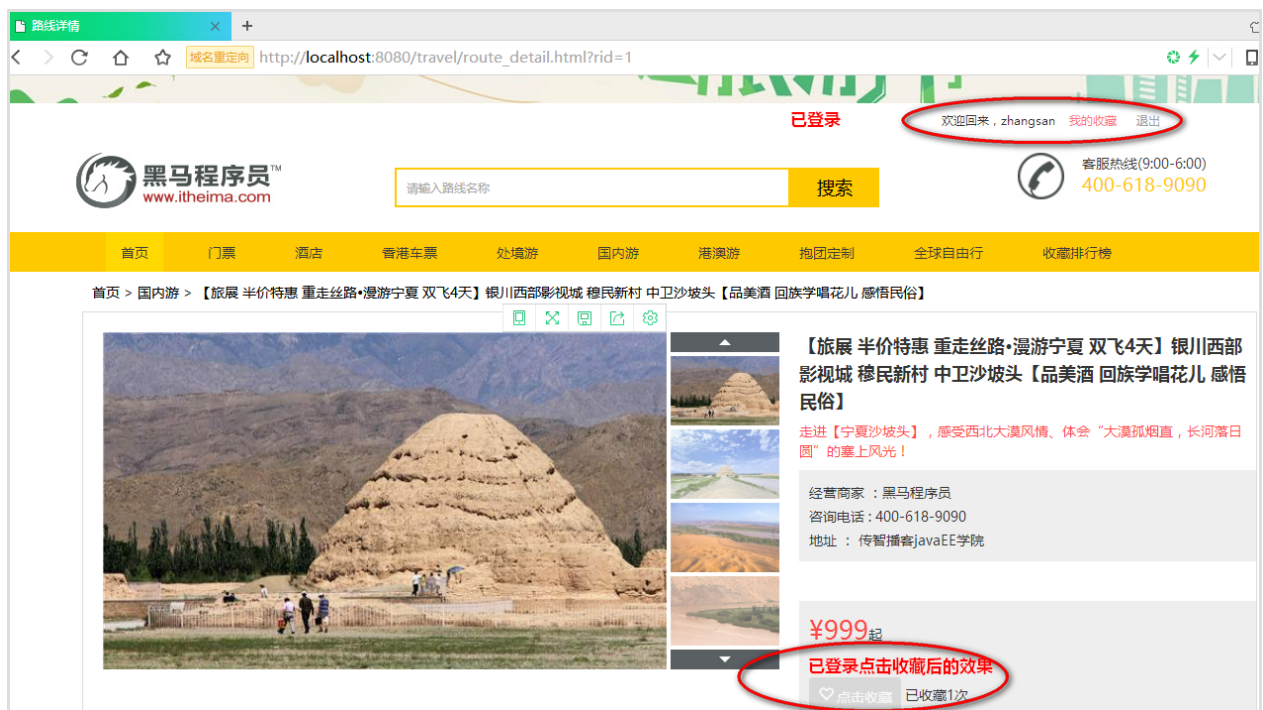
没有登录点击“点击收藏”跳转到登录页面



- 已经登录
 - 已经收藏, 显示不可编辑状态,显示收藏次数



- 没有收藏, 显示可编辑状态, 点击“点击收藏”实现成功收藏, 并更新收藏次数效果



二,思路分析

1.详情页面“点击收藏”显示是否可编辑分析

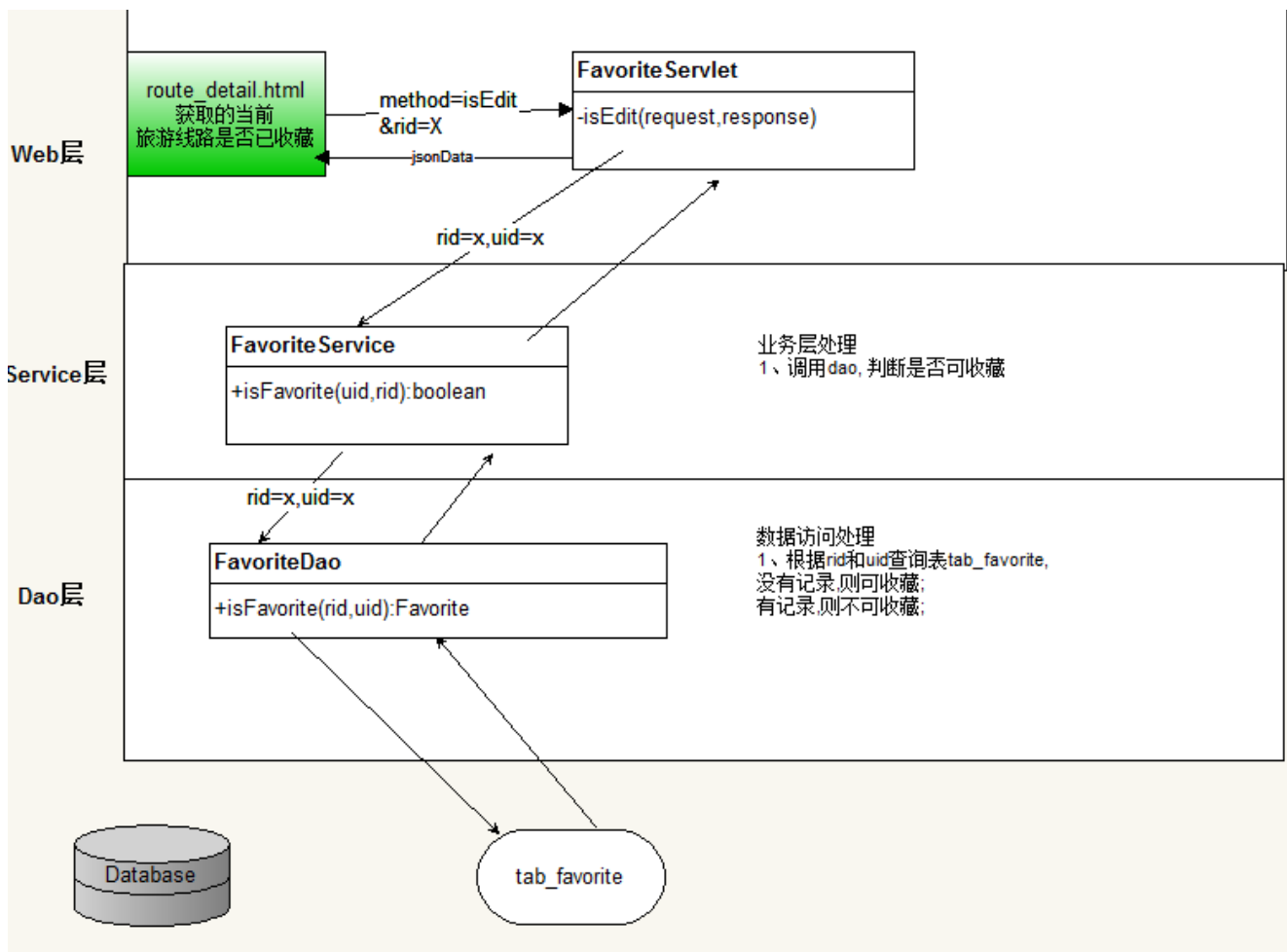
没有登录



登录了,但是没有收藏



登录了,收藏了



1. route_detail.html, 发送Ajax请求, 获得当前的旅游线路是否可收藏

```
$.post("favorite",{method:"isEdit",rid:rid}.....)
```

2. 创建FavoriteServlet继承BaseServlet, 创建isEdit(), 在isEdit()

//判断用户是否登录过了

//登录过了, 获得请求参数rid

//获得uid, 调用业务, 判断用户是否收藏过

//封装ResultInfo, 响应数据

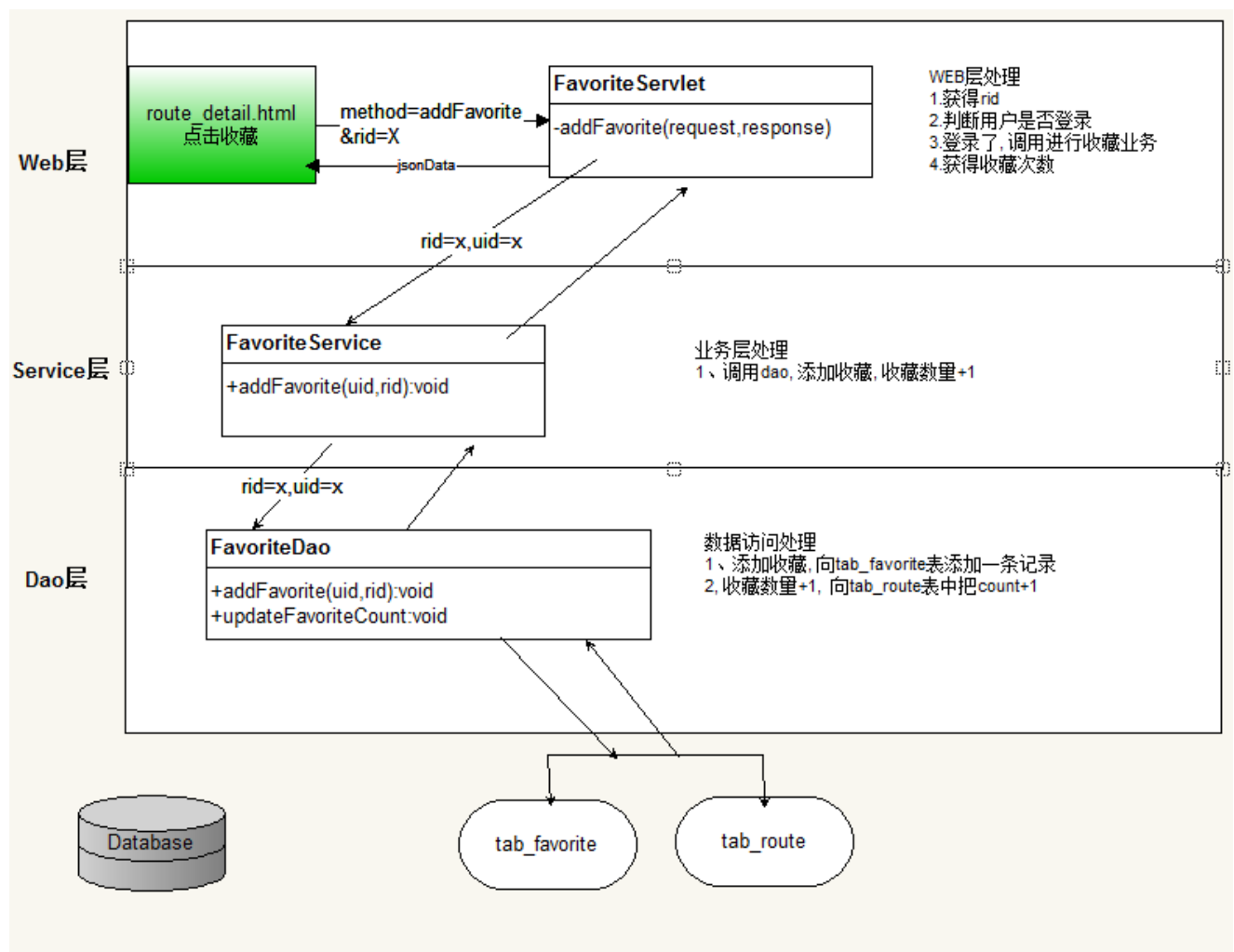
3. 创建FavoriteService

调用Dao, 判断当前用户是否收藏了

4. 创建FavoriteDao

根据uid和rid查询tab_favorite

2. 点击“点击收藏”实现增加收藏数据分析



三, 代码实现

1. 详情页面“点击收藏”显示是否可编辑实现

- route_detail.html

```

//*****判断是否可以收藏*****
$.post("favorite",{method:"isEdit",rid:rid},function (result) {
    if(result.flag){
        //没有异常的，判断是否可以编辑
        if (result.data.isEdit){
            //可以编辑
            //a 移除class, already
            $(".collect a").removeClass("already");
            //b.移除 disabled
            $(".collect a").removeAttr("disabled");
        }
    }else{
        alert(result.msg);
    }
}, "json");

```

- FavoriteServlet.java

```

@WebServlet("/favorite")
public class FavoriteServlet extends BaseServlet{

    private FavoriteService favoriteService = new FavoriteService();
    private RouteService routeService = new RouteService();

    /**
     * 判断当前UI是否可以编辑(是否可以显示 点击收藏)
     * @param request
     * @param response
     */
    public void isEdit(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String data = null;
        ResultInfo resultInfo = null;
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, Object> map = new HashMap<String, Object>();
        try {

            //判断用户是否登录过了
            User user = (User) request.getSession().getAttribute("user");
            if(user == null){
                map.put("isEdit", true);
                resultInfo = new ResultInfo(true, map, "没有登录!");
            }else{
                //登录过了, 获得请求参数rid
                String rid = request.getParameter("rid");
                //获得uid, 调用业务, 判断用户是否收藏过
                boolean isFavorite = favoriteService.isFavorite(user.getUid(), rid); //true就是收
                藏过, false没有收藏过
                if(isFavorite){//收藏过
                    map.put("isEdit", false);
                    resultInfo = new ResultInfo(true, map, "登录过但是已经收藏了!");
                }else{//没有收藏过
                    map.put("isEdit", true);
                    resultInfo = new ResultInfo(true, map, "登录过但是没有收藏了!");
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
            resultInfo = new ResultInfo(false, null, "服务器异常!");
        } finally {
            data = objectMapper.writeValueAsString(resultInfo);
            System.out.println("是否可以编辑data=" + data);
            response.getWriter().print(data);
        }
    }
}

```

```
}
```

- FavoriteServie.java

```
public class FavoriteService {

    private FavoriteDao favoriteDao = new FavoriteDao();

    /**
     * 判断当前用户是否收藏了
     * @param uid
     * @param rid
     * @return true就是收藏过了，false没有收藏过
     */
    public boolean isFavorite(int uid, String rid) throws Exception{
        try {
            //可以查询到的，收藏过了
            Favorite favorite = favoriteDao.isFavorite(uid,rid);
            return true;
        } catch (Exception e) {
            //查询不到的，没有收藏过
            e.printStackTrace();
            return false;
        }
    }
}
```

- FavoriteDao.java

```
public class FavoriteDao {
    JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Util.getDataSource());
    /**
     * 根据uid和rid查询tab_favorite表，判断是否收藏过
     * @param uid
     * @param rid
     * @return
     * @throws Exception
     */
    public Favorite isFavorite(int uid, String rid) throws Exception {
        String sql = "SELECT * FROM tab_favorite WHERE rid = ? AND uid = ?";
        return jdbcTemplate.queryForObject(sql,new BeanPropertyRowMapper<>
(Favorite.class),rid,uid);
    }
}
```

2.点击“点击收藏”实现增加收藏数据实现

- 在route_detail.html页面给点击收藏设置点击事件

```
<a onclick="addFavorite()" class="btn already disabled="disabled">  
    <i class="glyphicon glyphicon-heart-empty"></i>点击收藏  
</a>
```

- route_detail.html中创建函数响应点击事件

```
//4. 添加收藏  
function addFavorite() {  
    alert("哈哈");  
  
    //添加收藏  
    $.post("favorite", {method: "addFavorite", rid: rid}, function (result) {  
        if (result.flag) {  
            //判断是否登录  
            var data = result.data;  
            if (data.isLogin) {  
                //登录过, 别完成了收藏  
                //显示不可以收藏(添加类already, 添加disabled属性)  
                $(".collect a").addClass("already");  
                $(".collect a").attr("disabled", "disabled");  
                //清除点击事件  
                $(".collect a").removeAttr("onclick");  
                //显示收藏次数  
                $("#collectSpanId").html("已收藏"+data.count+"次");  
  
            } else {  
                location.href = "login.html";  
            }  
  
        } else {  
            alert(result.msg);  
        }  
    }, "json");  
}
```

- FavoriteServlet.java

```

@WebServlet("/favorite")
public class FavoriteServlet extends BaseServlet {

    private FavoriteService favoriteService = new FavoriteService();
    private RouteService routeService = new RouteService();

    /**
     * 添加收藏
     * @param request
     * @param response
     */
    public void addFavorite(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String data = null;
        ResultInfo resultInfo = null;
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, Object> map = new HashMap<String, Object>();
        try {
            //1. 获得rid
            String rid = request.getParameter("rid");

            //2. 判断用户是否登录
            User user = (User) request.getSession().getAttribute("user");
            if(user == null){
                //没有登录
                map.put("isLogin", false);
                resultInfo = new ResultInfo(true, map, "没有登录的!");
            }else{
                //登录过了, 调用收藏的业务,
                favoriteService.addFavorite(user.getId(), rid);
                //再获得收藏的count 响应
                int count = routeService.findByRid(Integer.parseInt(rid)).getCount();
                map.put("isLogin", true);
                map.put("count", count);
                resultInfo = new ResultInfo(true, map, "登录了, 并且收藏成功了!");
            }
        } catch (Exception e) {
            e.printStackTrace();
            resultInfo = new ResultInfo(false, null, "服务器异常!");
        } finally {
            data = objectMapper.writeValueAsString(resultInfo);
            System.out.println("收藏的数据data=" + data);
            response.getWriter().print(data);
        }
    }
}

```

- FavoriteService.java

```

public void addFavorite(int uid, String rid) throws Exception {
    Connection connection = null;

    try {
        //*****1,开启事务
        //获得数据源
        DataSource dataSource = C3P0Util.getDataSource();
        //启动事务管理器（获取datasource操作数据库连接对象并绑定到当前线程中）
        TransactionSynchronizationManager.initSynchronization();
        //从数据源中获取jdbcTemplate操作的当前连接对象
        connection = DataSourceUtils.getConnection(dataSource);
        //设置连接不自动提交事务
        connection.setAutoCommit(false);
        //调用Dao 向tab_favorite插入一条数据
        favoriteDao.addFavorite(uid,rid);
        //调用Dao 更新tab_route count+1
        favoriteDao.updateCount(rid);
        //提交事务
        connection.commit();
    } catch (Exception e) {
        e.printStackTrace();
        //回滚事务
        connection.rollback();
    } finally {
        try {
            //*****5,还原成自动事务
            //释放当前线程与连接对象的绑定
            TransactionSynchronizationManager.clearSynchronization();
            //重置当前连接为自动提交事务
            connection.setAutoCommit(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

- FavoriteDao.java


```

public class FavoriteDao {
    private JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Util.getDataSource());

    public void addFavorite(int uid, String rid) throws Exception {
        String sql = "insert into tab_favorite values(?,?,?)";
        jdbcTemplate.update(sql, rid, new Date(), uid);
    }

    public void updateCount(String rid) throws Exception {
        String sql = "update tab_route set count = count+1 where rid = ?";
        jdbcTemplate.update(sql, rid);
    }

    /**
     * 根据uid和rid查询tab_favorite表, 判断是否收藏过
     * @param uid
     * @param rid
     * @return
     * @throws Exception
     */
    public Favorite isFavorite(int uid, String rid) throws Exception {
        String sql = "SELECT * FROM tab_favorite WHERE rid = ? AND uid = ?";
        return jdbcTemplate.queryForObject(sql, new BeanPropertyRowMapper<>
(Favorite.class), rid, uid);
    }
}

```

面向接口编程

一,面向接口编程

1.介绍

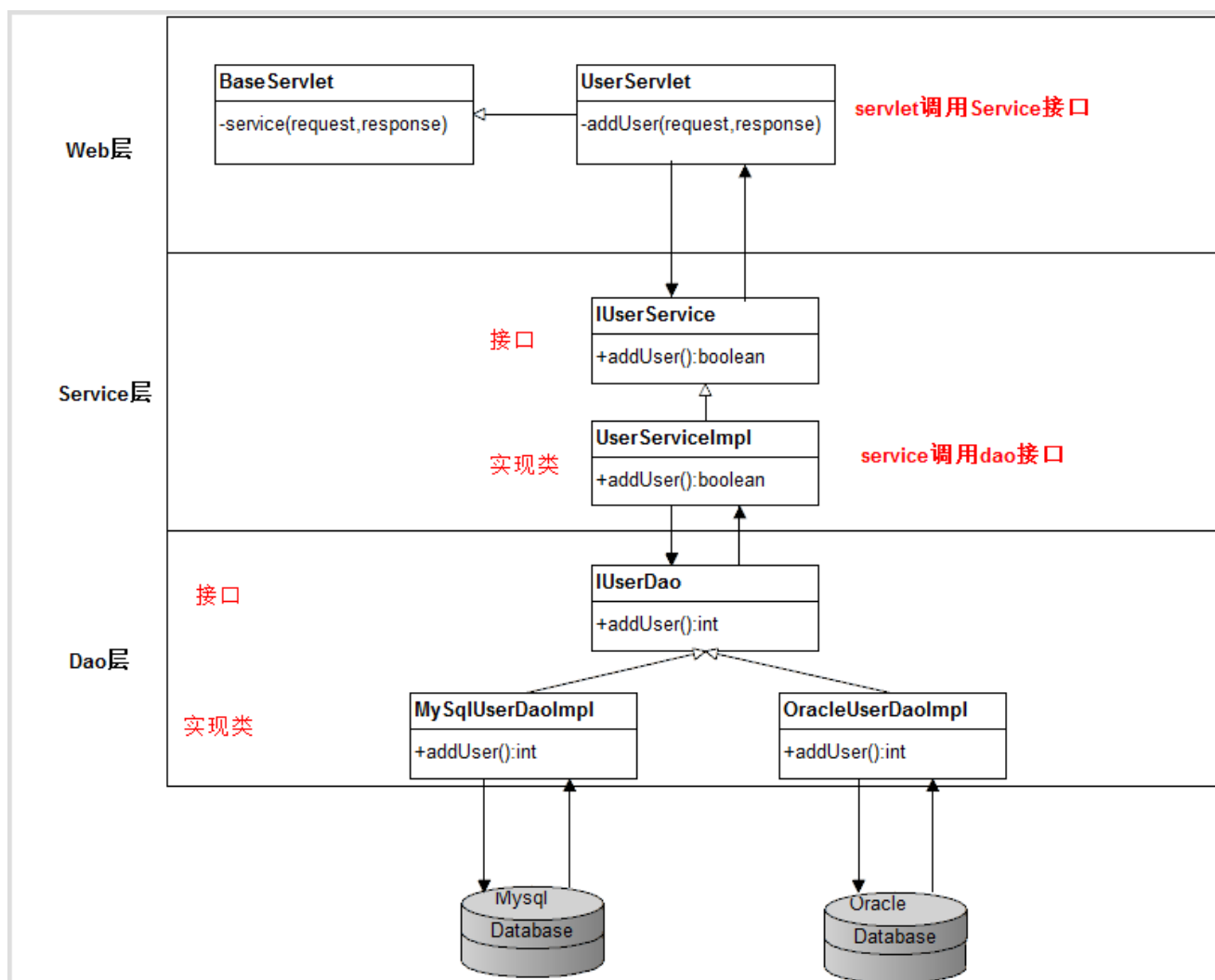
面向接口编程是开发程序的功能先定义接口，接口中定义约定好的功能方法声明，通过实现该接口进行功能的实现，完成软件或项目的要求。软件或项目随着时间的不断变化，软件的功能要进行升级或完善，开发人员可以创建不同的新类重新实现该接口中所有方法的方式进行开发，从而达到系统升级和扩展的目的。

2.面向接口编程案例

2.1 案例需求

在一个软件企业中开发的软件要求即可操作mysql数据库，也可以操作Oracle数据库实现数据的存储与查询。现要求开发一个软件添加用户的功能，可以通过配置文件灵活配置软件是使用mysql数据库还是oracle数据库实现功能。

2.2案例的架构图



3. 面向接口编程的好处

1. 隐藏实现

web层只调用service接口，service层只调用Dao接口，上一层只调用下一层接口，不需要知道具体实现类，从而隐藏了实现。

2. 易扩展

系统功能升级扩展，我们知道程序设计的原则是对修改是关闭，对新增是开放。面向接口编程扩展功能只需要创建新实现类重写接口方法进行升级扩展就可以了，达到了可以不修改源程序代码的基础上达到扩展的目的。

4. 实现代码

4.1 初级版本

- 步骤1：创建IUserService接口，定义addUser()接口方法

```

/**
 * 用户业务类接口
 */
public interface IUserService {

    /**
     * 添加用户方法定义
     * @return boolean, true代表添加成功, false代表失败
     * @throws Exception
     */
    boolean addUser() throws Exception;
}

```

- 步骤2: 创建UserServicelImpl实现类实现IUserService接口, 实现addUser()方法

```

public class UserServiceImpl implements IUserService {
    /**
     * 添加用户方法定义
     *
     * @return boolean, true代表添加成功, false代表失败
     * @throws Exception
     */
    @Override
    public boolean addUser() throws Exception {
        System.out.println("UserServiceImpl执行添加用户业务方法。。。");
        return true;
    }
}

```

- 步骤3: 在web层的UserServlet里面创建实例IUserService接口对象定义,并调用addUser()业务方法

```

/*
 * 实例IUserService
 */
private IUserService userServiceItf = new UserServiceImpl();

/**
 *
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
private void addUser(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    try {
        userServiceItf.addUser(); //调用业务方法
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

4.2 终极版本

面向接口编程的目的是隐藏实现类，在初级版本里面明确使用了UserServiceImpl实现类对象。那么如何实现灵活配置实现类达到动态创建实例实现类对象呢？

- 首先，定义配置文件impl.properties配置文件，存放在“src/main/resources”目录下impl.properties文件内容：通过配置文件配置实现类优点非常多，可以灵活配置扩展的实现类

```
IUserService=com.itheima.travel.service.impl.UserServiceImpl
```

- 其次，创建工厂类FactoryUtil.java，代码如下

```

/**
 * 工程工具类
 */
public class FactoryUtil {

    //定义读取properties配置文件对象
    private static ResourceBundle resourceBundle;
    static{
        //读取类路径下面的impl.properties文件
        resourceBundle = ResourceBundle.getBundle("impl");
    }

    /**
     *
     * @param itfName, 配置文件中key
     * @return Object, 返回类完整名的实例对象
     */
    public static Object getImplObject(String itfName){

        try {
            //根据itfName接口名字获取这个key的value, 也就是获取类完整名(包名+类名)
            String className = resourceBundle.getString(itfName);
            //根据类完整名实例对象, 调用无参构造函数
            return Class.forName(className).getConstructor().newInstance();
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    }
}

```

- 优化web层的UserServlet里面创建实例UserService接口对象定义, 达到隐藏实现类的目的

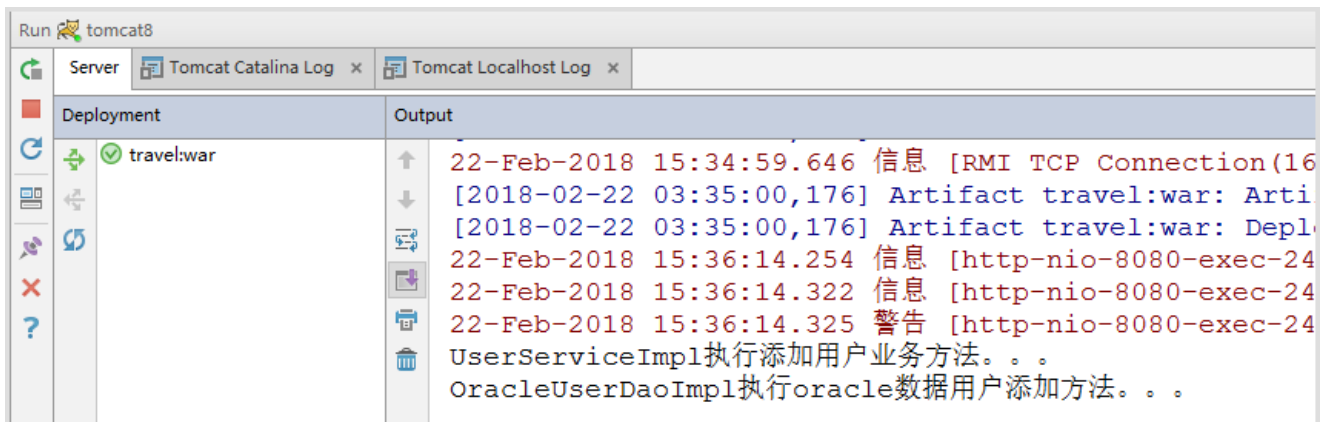
```

/**
 * 实例IUserService
 */
private IUserService userServiceItf =
(IUserService)FactoryUtil.getImplObject("IUserService");

/**
 *
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
private void addUser(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    try {
        userServiceItf.addUser();//调用业务方法
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

运行项目，打开浏览器运行"<http://localhost:8080/travel/user?mmethod=addUser>",控制台输出信息：



The screenshot shows the IDE's Run console with the following content:

Deployment	Output
travel:war	<pre> 22-Feb-2018 15:34:59.646 信息 [RMI TCP Connection(16 [2018-02-22 03:35:00,176] Artifact travel:war: Arti [2018-02-22 03:35:00,176] Artifact travel:war: Depl 22-Feb-2018 15:36:14.254 信息 [http-nio-8080-exec-24 22-Feb-2018 15:36:14.322 信息 [http-nio-8080-exec-24 22-Feb-2018 15:36:14.325 警告 [http-nio-8080-exec-24 UserServiceImpl执行添加用户业务方法。。。 OracleUserDaoImpl执行oracle数据用户添加方法。。。 </pre>