高级 Web 期末 PJ 文档

小组成员：宋浩 张亚中 赵宏博

2016 年 6 月 30 日

目录

1 项目概述

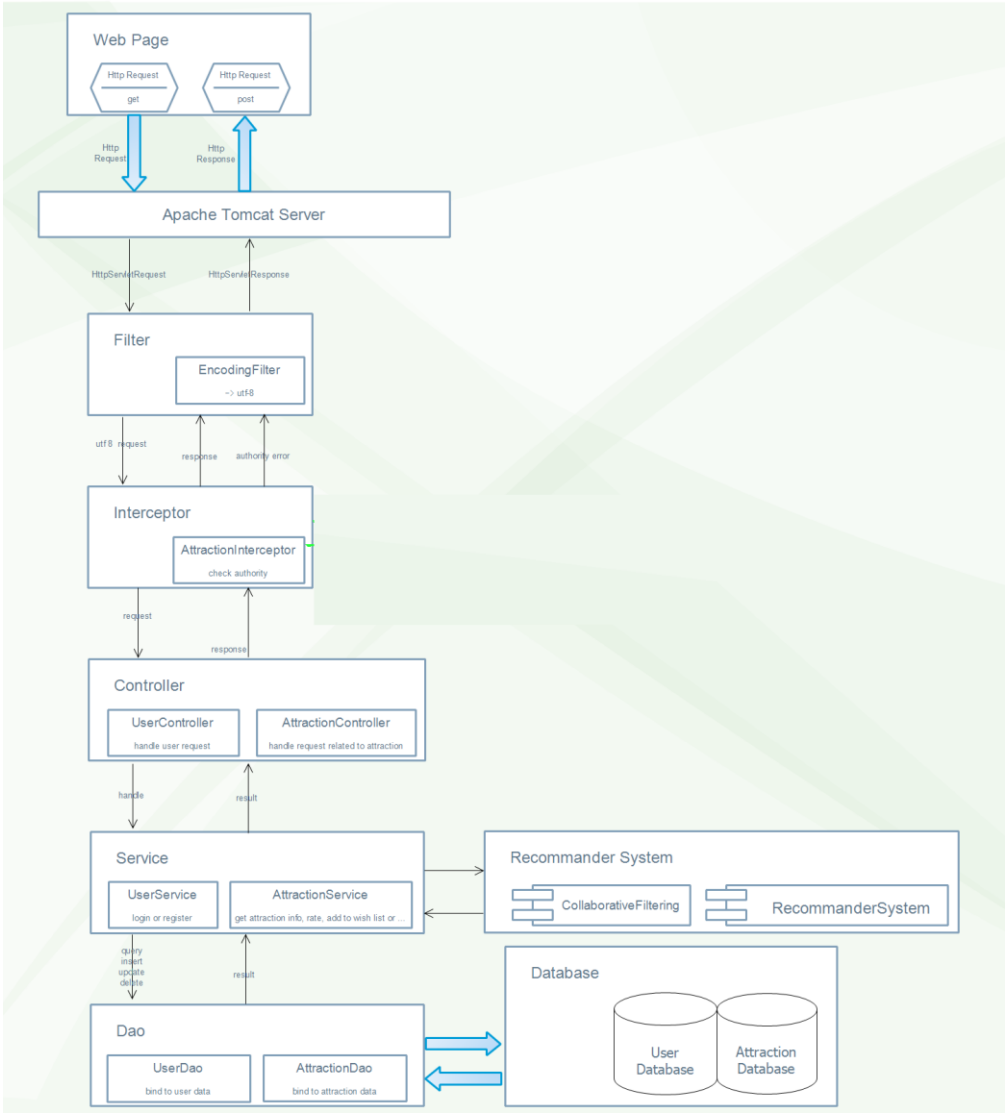
本项目为一个基于 Ionic，Spring 等技术开发的景观点评 Hybrid APP。本 APP 旨在帮助用户根据自己的实时位置，获取周边的景点信息，并查看其他用户的打分及评论。用户也可以自己对一个景点进行评论、打分及上传相应的照片、视频文件。以此构建一个用于景观点评和信息分享的系统。

2 团队分工

宋浩： 前台逻辑、交互设计；页面编写；3D 模型显示；推荐系统
张亚中： 后端服务器编写；数据库设计
赵宏博： 百度地图 API 调研；路径规划、评论页面编写

3 项目整体架构

由于 Struts 框架太重，所以后台使用了 Spring + SpringMVC + Hibernate 框架，整体框架如下（附件包含原图）：



前端由 Web 页面组成，显示地图、景观、导航、用户评价等。通过 form、ajax 等方式，前端页面与后台服务器通信。在这个项目中，请求处理分为了两类，分别是用户相关（登录、注册）和景观相关（查询、添加信息）。架构图中非常清晰地描述了前后端交互时数据的流向：

首先，数据经过 Filter 过滤器，这一层包含一个 EncodingFilter，由 Spring 框架提供，负责将所有 Request 的编码转为 UTF-8，防止中文乱码等问题。

然后数据会经过 Interceptor 拦截器，这里的 AttractionInterceptor 负责检查用户的登录状态，若要发起有关景观信息的请求，用户必须是登录状态，否则将返回错误信息。但用户相关请求不会受到影响。

```
@Override
public boolean preHandle(HttpServletRequest httpServletRequest,
                         HttpServletResponse httpServletResponse,
                         Object o) throws Exception {
    Cookie[] cookies = httpServletRequest.getCookies();
    if (cookies != null) {
        for (Cookie cookie : cookies)
            if (cookie.getName().equals("userId")
                    && Integer.parseInt(cookie.getValue()) > 0)
                return true;
    }
    return false;
}
```

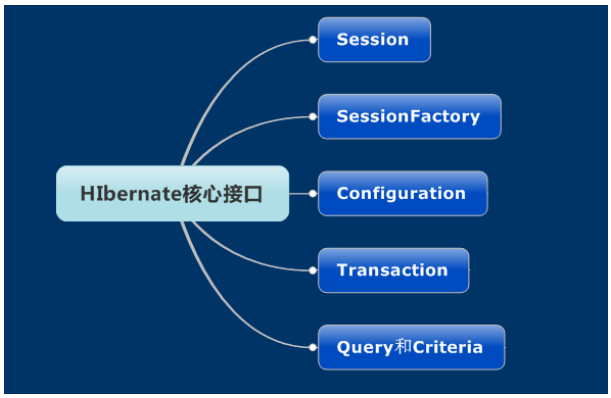
通过拦截器后，SpringMVC 负责 url 分发，将请求交付对应的 Controller。比如用户把景观添加到心愿单这个请求，将会交付 AtrractionController 的 addWish(userId, attractionId) 方法，返回值为是否添加成功，代码如下：

```
@ResponseBody
@RequestMapping(params = "action=wish", produces = "application/json;
charset=utf-8")
public String addWish(@CookieValue("userId") int userId,
                      @RequestParam("attractionId")  int  attractionId)
throws JSONException {
    if (attractionService.findWish(userId, attractionId) != null) {
        return ResponseMaker.error(-1);
    } else {
        Wish wish = new Wish(userId, attractionId,
                new Timestamp(System.currentTimeMillis()));
        attractionService.insert(wish);
        return ResponseMaker.success();
    }
}
```

Controller 负责前后端交互的基本逻辑，而 Service 用于完成所有功能，比如这里的 findWish(userId, attractionId)，即在心愿单数据表中寻找用户和景观的关联关系，若已经添加心愿，则返回错误，否则将景观加入心愿单。

上例中包含了数据库的两个操作：查询，添加。由于使用了 Hibernate 框架，所以数据

库的操作变得非常简单。Hibernate 的核心接口主要有：



如果对象和数据表已经建立了持久化关系，当要添加某个实例时，只需要 getSession().save(object);

另一方面，我们还实现了基于协同过滤技术的推荐系统模块，用来向用户推荐合适的景观。这一部分详见 4.8。

当完成所有处理后，Controller 会将结果信息包装成特定格式的 JSON 字符串，返回给 Interceptor 拦截器，拦截器可以修改某些字段，然后交给 Filter 过滤器，相似地，过滤器会检查编码类型，并转为 UTF-8。最后通过 Tomcat 服务器，交付前端页面。

4 主要功能实现

4.1 景观列表

景观列表使用嵌套循环(ng-repeat)实现复杂的层次结构。外层循环为 Tab 的生成，内层循环为 Tab 内 List 的生成。效果如下:

对评分、收藏、足迹等排序的实现原理为 Angularjs 的 OrderBy Filter，将下面的 Button 绑定到一个更改 Filter 的事件上，然后让 List 中的内容根据 Filter 排序。以此实现点

击不同 Button 进行不同排序。

4.2 附近景观

用户进入附近页面之后，APP 会调用百度 API，加载地图并自动定位到上海市。而且，APP 会加载景点信息，包括景点的评分、足迹、收藏度、推荐度等信息，并保存在内部。

用户可以通过点击左上角悬浮框中的选择按钮，来选择是显示不同类型的景点。之后，APP 会根据用户的选择，调用百度地图中关于 Marker 的 API，来添加 Marker。例如用户点击“上海工业地址 1”的时候，类型为“上海工业类型 1”的景点，APP 调用 Marker 的 API，新建多个 Marker，并将其在地图上标注出来。

地图下方的列表中，也将有这些景点的信息，用户可以点击这些列表中的景点选项，来跳转到景点详细介绍页面。列表中还包含 5 个不同的按钮：“评分”、“收藏”、“足迹”、“收藏数”、“推荐”。用户可以点击这 5 个不同的按钮，将景点进行以不同方式进行排序。

显示在地图上的标注，都注册了点击事件，以相应用户的点击。通过调用百度地图 API 中的 *addEventListener(“click”, function)*方法，可以为标注添加点击事件。用户点击标注之后，地图会平移到标注附近，并弹出相应的景点简介的窗口，显示出景点的名称、评分，及其简要介绍。用户可以通过点击窗口上的“详情”按钮，进入相应景点详细介绍页面，或者点击“取消”按钮，关闭窗口。

4.3 路线规划

在路线页面中，用户可以通过输入“出发地”和“目的地”，然后点击“搜索”按钮，APP 会调用百度地图中搜索 API，现将两点之间的行车路线标注出来，之后，调用百度地图中检索数据的接口，以获取驾驶路线途径的各个中间点。然后将计算其他景点到这些中间点的距离，如果该距离比较小，则将符合条件的景点从地图中标注出来。

在地图中标注出来的每个景点都对应一个 Marker，而且每个 Marker 都注册了点击事件，也是调用百度地图 API 中 *addEventListener()*方法。用户点击地图上关于景点的标注，屏幕会自动平移到景点附近，而且屏幕左上方也会显示出相关景点的详细信息，包括景点的名称、评分，以及景点的图片，方便用户查看。用户可以通过点击“进入景点”，进入景点详细介绍的页面，或者点击“取消”，关闭弹窗。

4.4 搜索补全

搜索补全功能监听用户的键盘事件，然后在页面加载时获取后台的景点名称。实时的将用户的输入，与景点名称进行匹配，将匹配的结果显示在页面上。

4.5 素材库

4.5.1 文件类型匹配

前端获取相应景点其他用户上传的文件，通过分析文件的后缀名，选择相对应的 html tag，例如 jpg/png 选择<img>，mp4 选择<video>

4.5.2 3D 模型显示

3D 模型为 json 格式的文件通过 fabric.js/Three.js 的 loadFromJSON()和 JSONLoader()来实现。

4.6 区域识别凸显

区域识别凸显利用一个多边形环绕景点一周而成。多边形的顶点为人工标注，并存入相

对应景点的数据库中。

前端获得相应的多边形坐标后利用，将这些坐标创建一个个 BMap.Point，然后将这些点组成一个数组，利用百度 API 将这个数组传入 BMap.Polygon 中，完成对这个多边形的绘制。以此实现区域识别凸显。

4.7 标识和文字建议

在评论页面中，页面首先会获取相关景点的详细信息，包括景点的边框，及其之前的标注信息。之后，APP 会调用百度地图中的关于 Marker 的 API，按照不同类型，新建不同的自定义图片的标注，然后将这些标注添加到地图界面中。另外，获取到的景点边框是多个坐标点，页面加载时，会将这些坐标点放到一个数组中，以多边形的形式在地图中描绘出来，以表示景点的边框。

页面加载完成之后，用户可以查看相关景点的边框范围，以及之前的标注，如果标注有文字描述，相应的文字描述会以标签的形式显示在标注旁边。

用户也可以添加不同类型的标注，包括添加带有文字描述的标注。用户添加标注时，首先点击相关标注的图标，之后，地图中调用百度地图中 Marker（标注）的 API，根据选中的标注类型新建一个带图标的标注，并在地图中间显示出来，然后用户就可以拖动该标注，放到合适的位置即可。当用户完成拖动之后，该标注的坐标就会保存下来。如果用户需要添加带有文字说明的图标，需要首先在输入框中输入相关的文字说明，然后就可以像添加普通图标一样添加。

当用户添加完成之后，点击“保存”按钮，就可以将自己的修改提交保存。这时，APP会将之前临时描绘的标注提取出其坐标、标注类型和标注内容（如果有的话），一起提交至后台。

4.8 推荐系统

本项目使用协同过滤作为推荐系统的算法。协同过滤是利用集体智慧的一个典型方法。举例来说如果用户现在想去某个景点，但不知道具体去哪个，系统通过用户曾经的操作，例如评分，加入足迹等行为，去寻找与之类似的用户，然后将类似用户去过的景点，推荐给当前用户。

我们使用皮尔逊相关系数计算用户与景点间的相似度。在这里我们只考虑用户的评分，我们认为评分是用户对于一个景点最准确和肯定的评价。经过计算皮尔逊相关度后，我们可以获得与这个用户最相似的其他用户。

然后我们使用简单的加权混合来模拟计算该用户对于其他景点的打分，并输出评分最高的几项，作为我们给该用户的推荐。