

## 目录

1 数据可视化之地图 + Echarts.....	1
1.1 任务目标.....	1
1.2 具体过程及实现.....	1
1.2.1 数据库的安装和配置 .....	1
1.2.2 将数据导入数据库 .....	1
1.2.3 使用 JDBC 连接数据库.....	2
1.2.4 站点的动态加载.....	2
1.2.5 Echarts 图表的显示.....	4
1.3 遇到的问题以及解决方案.....	4
1.3.1 oracle 连接过程中的错误:ORA-12514.....	4
1.3.2 数据库部分字体乱码 .....	6
1.3.3 Java 对象转换成 json .....	8
1.4 项目的创新部分 .....	10
1.4.1 建立 ORM 映射时采用 DAO 模式.....	10
1.4.2 摒弃 jsp，前后端解耦.....	11
1.4.3 动态展示 Echarts 部分 .....	12
1.5 项目效果展示.....	14
2 数据可视化之弧线绘制 .....	15
2.1 任务目标.....	15
2.2 具体过程及实现.....	15
2.2.1 导入站点并显示海量点 .....	15
2.2.2 弧线的绘制.....	15
2.2.3 箭头的绘制.....	15
2.2.4 自流量的表示 .....	16
2.3 遇到的问题及解决 .....	16
2.3.1 箭头的重新绘制.....	16
2.3.2 清除无关海量点.....	16
2.4 项目的创新部分 .....	17
2.4.1 ES6 特性的使用 .....	17
2.4.2 清除按钮的设置.....	18
2.5 项目成果展示.....	18

# 1 数据可视化之地图 + Echarts

## 1.1 任务目标

1. 安装 Oracle11g 数据库
2. 将数据导入数据库中
3. 使用 JDBC 连接数据库，实现数据库的连接和对表的 sql 操作，使用 Java Web 实现页面的动态加载
4. 综合使用百度地图 API 和 Java Web 实现站点的动态加载和实现站点点击事件的添加
5. 综合使用百度地图 API, Echarts, Java Web 以及 JavaScript 实现用户点击地图上的站点，并选择时间后，显示这个时间段内车辆的借车量的 Echarts 图表

## 1.2 具体过程及实现

### 1.2.1 数据库的安装和配置

#### 1.2.1.1 数据库的下载和安装

关于 Oracle11g 的下载安装，我参照的是下面的博客，有非常详细的说明：  
<https://blog.csdn.net/u011171506/article/details/79198013>

#### 1.2.1.2 数据库的配置

数据库的配置主要指数据库可视化工具的安装，我没有使用 PL/SQL 作为可视化工具，我使用的是 Navicat 作为数据库的可视化工具，关于 Navicat 的下载安装破解，参照以下网页：<https://www.jianshu.com/p/42a33b0dda9c>

### 1.2.2 将数据导入数据库

由于提供的数据格式为 bmp 格式，而 Navicat 没有原生支持 bmp 格式的导入，因此采用命令行导入的方式导入数据。命令行导入的步骤为：

1. 将 cmd 或 PowerShell 的目录切换至 oracle 安装的根目录下的 BIN 目录下
2. 使用如下命令：`imp username/pwd@orcl file=filepath`  
(注：@后根数据库实例名，实例名在安装步骤中有提及)  
例如 `imp scott/tiger@orcl_db file=D:\date.dmp`
3. 随后按提示操作即可

## 1.2.3 使用 JDBC 连接数据库

### 1.2.3.1 前期准备工作

需要准备好 IDE 和 Jar 包，IDE 推荐使用 IDEA（我本学期使用 Eclipse 开发），关于 IDE 的安装和使用，比较简单，不多赘述

Jar 包推荐使用 ojdbc6.jar，该 Jar 包可去 oracle 官网下载

### 1.2.3.2 核心代码

核心代码的主要包括：

1. 加载驱动
2. 连接数据库
3. 创建执行语句
4. 返回结果
5. 关闭连接

其中可将步骤 1，2，5 的代码剥离出来，封装成小工具使用，每次使用只需引用，能提高代码的可读性和可维护性。具体的代码比较简单，网上教程很多，不过多赘述。

## 1.2.4 站点的动态加载

### 1.2.4.1 百度地图的静态加载

对于地图的显示，首先需要在百度地图开放平台申请 key，然后可以参照官网给出的 DEMO，引入百度地图 API 的 JS 依赖，创建一个地图实例并且初始化，核心代码如下：

```
//引入百度地图依赖
```

```

<script
type="text/javascript"    src="http://api.map.baidu.com/api?v=2.0&ak= 您的 密 钥
"></script>
// 创建 Map 实例
let map = new BMap.Map("allmap");
// 初始化地图, 设置中心点坐标和地图缩放级别
map.centerAndZoom(new BMap.Point(120.28, 30.28), 12.5);
// 添加地图类型控件初始化地图, 设置中心点坐标和地图缩放级别
map.addControl(new BMap.MapTypeControl({
    mapTypes : [ BMAP_NORMAL_MAP, BMAP_HYBRID_MAP ]
}));
// 设置地图显示的城市 此项是必须设置的
map.setCurrentCity("杭州");
// 开启鼠标滚轮缩放
map.enableScrollWheelZoom(true);

```

### 1.2.4.2 站点的显示和点击事件的添加

对于点的动态加载, 首先在 jsp 文件的 java 部分使用先前准备好的函数连接数据库并得到需要的结果, 再将结果集包装成 json 格式的数组并赋值给 points, 然后, 将这个值传给前端, 前端的 JS 遍历这个数组实现点的加载和点击事件的添加, 核心代码如下:

```

<%
    IStationInfoDao dao = new StationInfoDaoImpl();
    List<StationInfo> all = dao.getAll();
    ArrayList<JSONObject>points= new ArrayList<JSONObject>();
    for (StationInfo k : all) {
        JSONObject js = JSONObject.fromObject(k);
        points.add(js);
    }
%>
    var opts = {
        width : 200,
        height : 130,
        title : "站点信息"
    };
    var info = <%= points %>;
    info.forEach(function(item) {
        var point = new BMap.Point(item.x, item.y);
        var marker = new BMap.Marker(point); // 创建标注

        var infoWindow = new BMap.InfoWindow("id 是" + item.stationID + "<br>"

```

```

        + "name 是" + item.stationName + "<br>" + "address 是"
        + item.address, opts);

marker.addListener("click", function() {
    map.openInfoWindow(infoWindow, point);
    document.getElementById('stationName').innerHTML = item.stationID;
});
map.addOverlay(marker); // 将标注添加到地图中
})

```

## 1.2.5 Echarts 图表的显示

Echarts 图表的显示较为简单，官方 demo 中 (<http://www.echartsjs.com/examples/editor.html?c=area-stack>) 已经给出了详细的代码，我们只需将其中的参数进行替换，把需要展现的数据放在数组中再传入 option 即可，在此不过多赘述。

## 1.3 遇到的问题以及解决方案

### 1.3.1 oracle 连接过程中的错误:ORA-12514

我们在初次连接 oracle 时，可能会遇到“ORA-12514, TNS:listener does not currently know of service requested in connect descriptor”，接下来我来详细介绍一下我的解决方案（win10 环境下）：

根据报错原因，判断出客户端未监听到实例名，解决思路：

1. 通过重启服务来解决。（一般都是没用的）

2. 考虑监听配置文件 listener.ora，步骤如下：

1. 找到文件位置，（我的在“E:\app\XCQ\product\11.2.0\dbhome\_1\NETWORK”，每个人的不一样，根据自己情况查找）我的文件内容如下：

```

# listener.ora Network Configuration File:
      E:\app\XCQ\product\11.2.0\dbhome_1\NETWORK\ADMIN\listener.ora
# Generated by Oracle configuration tools.

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = CLRExtProc)

```

```
(ORACLE_HOME = E:\app\XCQ\product\11.2.0\dbhome_1)
(PROGRAM = extproc)
(ENVS
"EXTPROC_DLLS=ONLY:E:\app\XCQ\product\11.2.0\dbhome_1\bin\oraclr11.dll")
)
(SID_DESC =
(GLOBAL_DBNAME = orcl_db)
(ORACLE_HOME = E:\app\XCQ\product\11.2.0\dbhome_1)
(SID_NAME = ORCLDB)
)
)

LISTENER =
(DESCRIPTION =
(AADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1521))
)

ADR_BASE_LISTENER = E:\app\XCQ
```

标红的部分是需要添加的，其中 GLOBAL\_DBNAME 和 SID\_NAME（不区分大小写）需要的值可以在配置数据库时产生的文件中查到（如图 1）：

数据库配置概要

全局数据库名: orcl\_db  
数据库配置类型: 单实例  
SID: orcldb  
管理选项类型: 无  
存储类型: 文件系统  
内存配置类型: 自动内存管理

图 1

如果是按照博客中的步骤安装，配置概要会是一样的，否则请参照自己的配置概要，由于我当时将配置概要保存了，所以能方便地查看，假如你没保存，请自行搜索如何查看自己数据库配置。

修改完配置文件后，保存，然后关闭数据库服务和监听服务。然后先启动数据库服务，再打开监听服务，注意顺序！注意顺序！注意顺序！

如图 2，倒数第三个和倒数第二个分别是监听服务和数据库服务。






 OracleJobSchedulerORCLDB		OracleJobSchedulerORCLDB	已停止
 OracleMTSRecoveryService	4060	OracleMTSRecoveryService	正在运行
 OracleOraDb11g_home1ClrAgent		OracleOraDb11g_home1ClrAgent	已停止
 OracleOraDb11g_home1TNSListe	9072	OracleOraDb11g_home1TNSListener	正在运行
 OracleServiceORCLDB	10600	OracleServiceORCLDB	正在运行
 OracleVssWriterORCLDB		Oracle ORCLDB VSS Writer Service	已停止

图 2

一般到这一步便可以解决问题了，但某些时候我们想用 scott 账户连接 oracle 时，会提示账户被锁定，这时我们只需解锁 scott 账户即可：

1. 打开 cmd，输入：sqlplus /nolog
2. 输入 conn system
3. 密码是 as sysdba
4. 然后使用命令解锁：SQL>alter user scott account unlock;
5. 然后赋予一个新口令：SQL>alter user scott identified by tiger;
6. 然后可以使用 scott/tiger 连接数据库了。（如图 3）



图 3

## 1.3.2 数据库部分字体乱码

字体乱码明显是编码问题，是由于提供的 dmp 文件和本地安装的数据库编码不一致导致的，接下来我详细说明一下如何解决数据库的编码问题：

### 1.3.2.1 查看字符集

1. Oracle server 端的字符集

2. Oracle client 端的字符集

3. dmp 文件的字符集

在做数据导入的时候，需要这三个字符集都一致才能正确导入。

- 查询 Oracle server 端的字符集：

有很多种方法可以查出 oracle server 端的字符集，比较直观的查询方法是以下这种：SQL> SELECT USERENV('LANGUAGE') FROM dual;

- 查询 oracle client 端的字符集

在 windows 平台下，就是注册表里 HKEY\_LOCAL\_MACHINE ---》SOFTWARE ---》 ORACLE-HOME 的 NLS\_LANG。环境变量最好也加上相同的键值。

如果检查的结果发现 server 端与 client 端字符集不一致，请统一修改为同 server 端相同的字符集。

- 如何查询 dmp 文件的字符集：

用 oracle 的 exp 工具导出的 dmp 文件也包含了字符集信息，dmp 文件的第 2 和第 3 个字节记录了 dmp 文件的字符集。如 0354，然后用以下 SQL 语句查出它对应的字符集：

```
SQL> SELECT NLS_CHARSET_NAME(TO_NUMBER('0345','xxxx')) FROM dual;
-----
ZHS16GBK
```

### 1.3.2.2 修改字符集

- 修改 server 端字符集

```
SQL> SHUTDOWN IMMEDIATE;    //关闭数据库
SQL> STARTUP MOUNT;          //启动到 Mount

SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
SQL> ALTER SYSTEM SET AQ_TM_PROCESSES=0;

SQL> ALTER DATABASE OPEN;

SQL> ALTER DATABASE CHARACTER SET ZHS16GBK;          //这里可以从父集到子集
SQL> ALTER DATABASE CHARACTER SET INTERNAL_USE AL32UTF8;          //如果是从子集到父集，需要使用 INTERNAL_USE 参数，跳过超子集检测

SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;
```

- 修改 client 端的字符集



在查看时便可修改，参照 1.3.2.1

### • 修改 dmp 文件字符集

上文说过，dmp 文件的第 2 第 3 字节记录了字符集信息，因此直接修改 dmp 文件的第 2 第 3 字节的内容就可以‘骗’过 oracle 的检查。这样做理论上也仅是从子集到超集可以修改，但很多情况下在没有子集和超集关系的情况下也可以修改，我们常用的一些字符集，如 US7ASCII，WE8ISO8859P1，ZHS16CGB231280，ZHS16GBK 基本都可以改。因为改的只是 dmp 文件，所以影响不大。

具体的修改方法比较多，最简单的就是直接用 UltraEdit 修改 dmp 文件的第 2 和第 3 个字节。

比如想将 dmp 文件的字符集改为 ZHS16GBK，可以用以下 SQL 查出该种字符集对应的 16 进制代码：

```
SQL> SELECT TO_CHAR(NLS_CHARSET_ID('ZHS16GBK'),'xxx') FROM dual;  
-----  
0354
```

然后将 dmp 文件的 2、3 字节修改为 0354 即可。如果 dmp 文件很大，用 UltraEdit 无法打开，就需要用程序的方法了。

### 1.3.2.3 关于本次遇到的乱码

本次提供的数据中，[part]stationOrient.dmp 这个文件会出现部分字段乱码，虽然该乱码对结果无影响，但本着刨根问底的精神，我做了些许尝试，发现只需在数据库服务器编码为 ZHS16GBK 的情况下导入，然后修改编码为 AL32UTF8 后即可正常显示。

### 1.3.3 Java 对象转换成 json

在前端处理数据时，json 绝对是最好的数据类型选择，因此我们需要将数据库获得的结果包装成 json 格式，然后返回给前端，但 Java 语言没有原生支持 json，因此我们需要引用第三方工具，我推荐 net.sf.json

使用该工具需要引用如下的 jar 包：

1. commons-beanutils-1.8.3.jar
2. commons-collections-3.2.jar
3. commons-lang-2.3.jar
4. commons-logging-1.0.4.jar
5. ezmorph-1.0.6.jar

## 6. json-lib-2.4-jdk15.jar

jar 包可在网上进行下载，将其导入进工程后，可使用 `JSONObject.fromObject(obj)` 函数将 Java 对象转换为 json 格式对象，但需要注意的是，被转换的 `obj` 对象，必须是一个 JavaBean 对象，或者至少具有 `getxxx()` 的方法，否则会得到空数据。

例如，有以下的示例代码：

```
import oracle.dao.IStationInfoDao;
import oracle.dao.impl.StationInfoDaoImpl;
import oracle.domain.StationInfo;

IStationInfoDao dao = new StationInfoDaoImpl ();
List<StationInfo> allStationInfo = dao.getAll();
ArrayList<JSONObject> points = new ArrayList<JSONObject>();
for (StationInfo k : allStationInfo) {
    JSONObject jsobj = JSONObject.fromObject(k);
    points.add(jsobj);
}
```

## 1.4 项目的创新部分

### 1.4.1 建立 ORM 映射时采用 DAO 模式

DAO 模式作为与数据库打交道的东西,他只关注怎么将数据写入数据库,和怎么取出来.使用 DAO 模式,虽然使程序架构的复杂性增加,但达到了解耦的目的,使程序更加健壮,便于后期功能的拓展以及维护。如图 4,是我程序中的 Java 部分的架构:

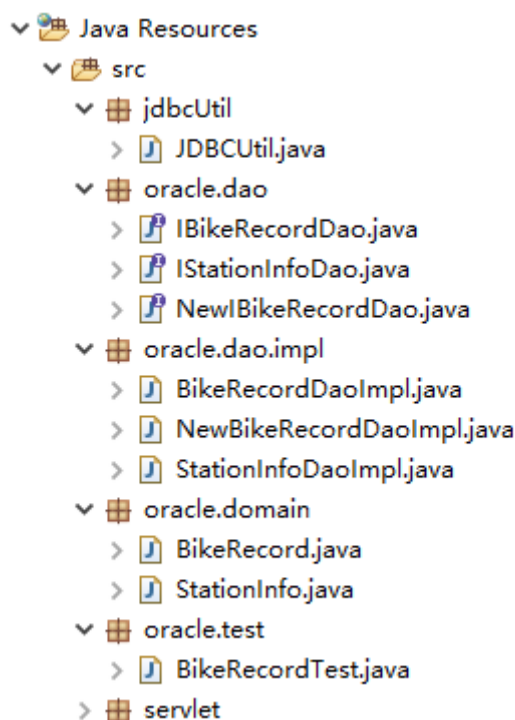


图 4

因为是第一次写 Java Web 项目,在 package 的命名上没有按照规范,但是这不影响架构的体现:

- jdbcUtil 中是我剥离出来的 JDBC 部分操作,在 1.2.3.2 中已说明
- oracle.domain 是和数据库建立的 ORM 映射,其中的类按照 JavaBean 的规范编写,类名就是数据库中的表名,类的属性是数据库中表的属性
- oracle.dao 中包含的都是接口类(interface),就是说这个类只声明具体函数,比如只声明具体的增删改查函数,却不在其中实现所声明的函数
- oracle.dao.impl 则包含函数的具体实现,每个类都继承(implements)对应的接口,在其中重载接口中声明的函数
- oracle.test 则是测试类,用于测试编写的 Java 代码能否达到预期效果具体的使用可以参照该例子:

```
NewIBikeRecordDao dao = new NewBikeRecordDaoImpl();
JSONArray Records = dao.getRelation();
System.out.println(Records);
```

## 1.4.2 摒弃 jsp，前后端解耦

### 1.4.2.1 为什么要这样做

我承认 jsp 是一个好的发明，在互联网早期，他非常的行之有效，方便快捷的展示一个页面，可是随着时代发展，在如今时代，网页越来越复杂的情况下，jsp 这种前后端不解耦的模式，会带来许多问题：

1. 动态资源和静态资源全部耦合在一起，服务器压力大，因为服务器会收到各种 http 请求，例如 CSS 的 http 请求，JS 的，图片的等等。一旦服务器出现状况，前后台一起玩完，用户体验极差。

2. jsp 必须要在支持 java 的 web 服务器里运行（例如 tomcat, jetty, resin 等），无法使用 nginx 等，性能提不上来。

3. 如果 jsp 中的内容很多，页面响应会很慢，因为是同步加载。

4. 第一次请求 jsp，必须要在 web 服务器中编译成 servlet，第一次运行会较慢。

5. 每次请求 jsp 都是访问 servlet 再用输出流输出的 html 页面，效率没有直接使用 html 高。

6. html, css, js, java 糅合在一起，不利于团队进行分工。

因此，我打算进行前后端解耦，前端使用 html+css+JavaScript，后台则用 servlet 封装接口，让前端异步地向后台请求数据，将项目的重心移至前端。

### 1.4.2.2 具体的操作

Web 部分的结构如图 5:

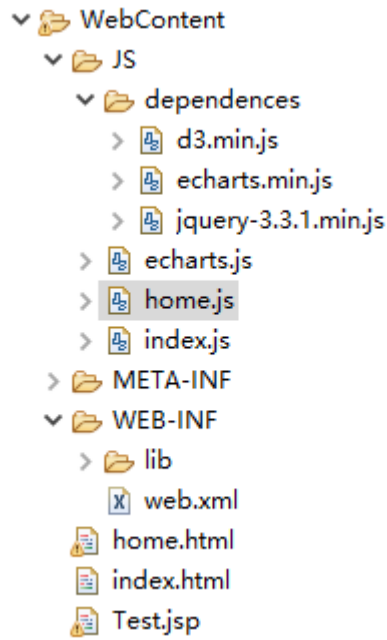


图 5

Test.jsp 是最初的页面，重构后，Test.jsp 被分为 home.html 和 home.js。在 home.html 中，我写了 html 和 CSS，主要是为百度地图和 Echarts 建立了一个 div。而在 home.js 中，我则做了百度地图的初始化工作和具体 Echarts 的绘制函数，Echarts 绘制所需的数据也在这个文件中通过 jQuery 异步地向后台请求。web.xml 中是 servlet 的具体配置，servlet 中则是调用之前写好的函数，使用 JDBC 连接数据库并取得数据。

### 1.4.3 动态展示 Echarts 部分

简单来说，就是 Echarts 的 div 最开始不显示，点击站点后才予以显示，并可通过隐藏按钮隐藏 Echarts 的 div

实现方案就是添加事件绑定，在 html 里，Echarts 的 CSS 中的 hidden 属性预先设置为 true。在 js 的站点点击事件绑定时，将 hidden 的属性改为 false，并修改地图和表格 div 的宽高比。

需要注意的是，Echarts 自身存在一个 bug，就是在事先隐藏，后来展示的 div 中画图时，会错误地读取 div 宽高，导致表格的大小达不到预期。因此，表

格的 CSS 需要设置高度为 0px（不可设置为百分制）还需要在画图时对表格进行 resize。相关核心代码如下：

```
//home.html
//地图的 CSS
#allmap {
    width: 100%;
    height: 100%;
    overflow: hidden;
    margin: 0;
    font-family: "微软雅黑";
}
//表格的 CSS
#echart {
    width: 100%;
    height: 0px;
    hidden:true;
}

//home.js
//给站点添加点击事件
marker.addEventListener("click", function() {
    let echart = document.getElementById('echart');
    map.openInfoWindow(infoWindow, point);
    echart.hidden = false;
    document.getElementById('allmap').style.height = "70%";
    echart.style.height = "30%";
    document.getElementById('stationID').innerHTML = item.stationID;
});

//echart.js
$(document).ready(function() {
    $('#draw').click(function() {
        let myChart = echarts.init(document.getElementById('echarts'));
        myChart.resize({
            width:document.getElementById('allmap').clientwidth,
            height:document.getElementById('allmap').clientHeight * 3 / 7,
//宽高比为 3 比 7
        })
    })
})
```

隐藏按钮的实现和上述类似，只是将 hidden 的属性改成 true 即可。

# 1.5 项目效果展示

图 6 为不点击站点时的页面，图 7 为点击站点并绘制后的效果

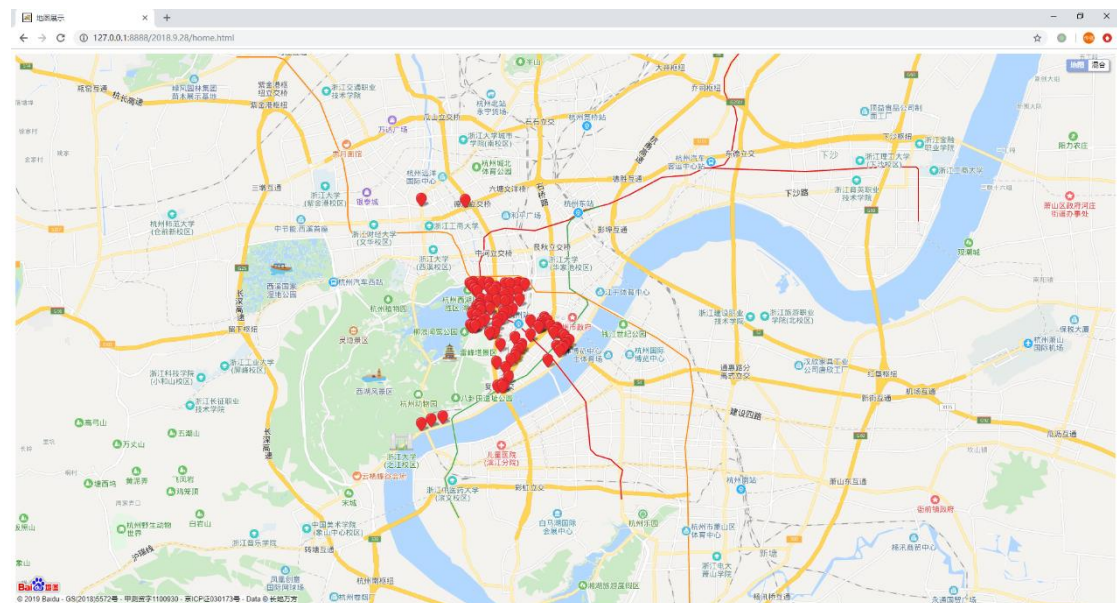


图 6

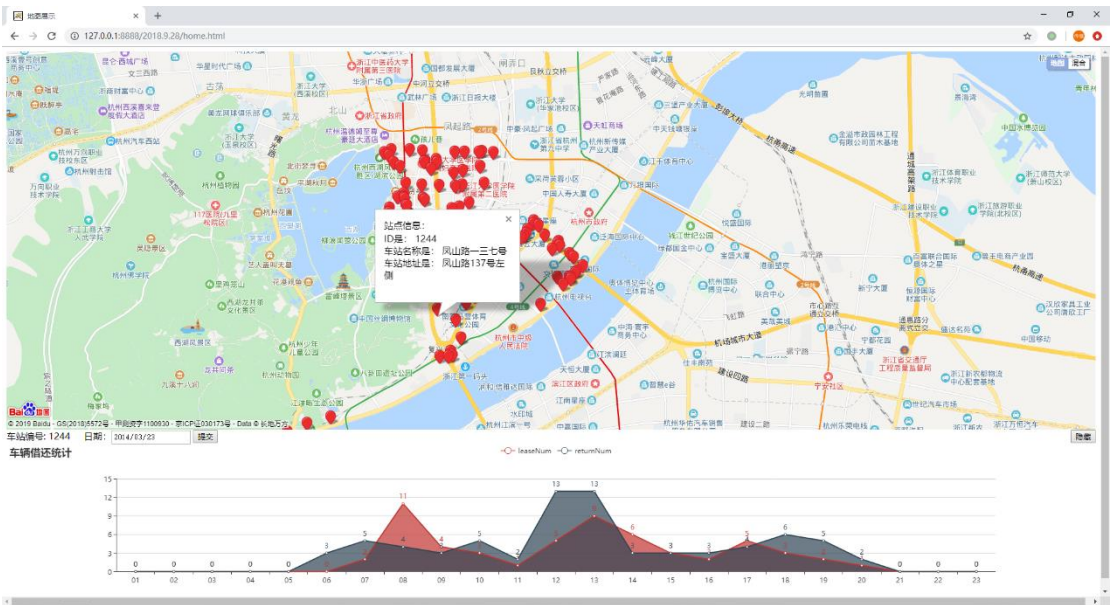


图 7



## 2 数据可视化之弧线绘制

### 2.1 任务目标

1. 导入新 dmp 文件并综合使用百度地图 API 实现海量点的动态加载
2. 实现站点间弧线的绘制来表示站点间的车流量
3. 给站点间的弧线绘制箭头
4. 显示自流量

### 2.2 具体过程及实现

#### 2.2.1 导入站点并显示海量点

在任务 1 的完成后，该任务并不具有难度。数据库导入数据，建立 ORM 映射，前端请求站点数据并展示即可。需要注意的是，新数据的数据量很大，站点应以海量点的形式展现，海量点的相关 API 可以在百度地图官方 DEMO 中学习，不过多赘述

#### 2.2.2 弧线的绘制

弧线的绘制使用的是百度地图提供的 `BMapLib.CurveLine()` 接口，具体的实现可参照官方 DEMO ([http://lbsyun.baidu.com/jsdemo.htm#c1\\_13](http://lbsyun.baidu.com/jsdemo.htm#c1_13))。我们要做的是替换参数。比如我们需要弧线的两个端点，这可以通过后台数据处理得到，我们需要弧线的粗细和颜色，这可以通过两个站点间的流量大小获取权值后得到。因此难度并不大。

我认为弧线绘制的难点在于绘制弧线后要清除无关的海量点，关于该难点的解决方案，我会在下文给出。

#### 2.2.3 箭头的绘制

箭头的绘制由于下发了箭头绘制函数，因此大大降低了难度，只需在绘制弧线时添加箭头即可。我认为箭头绘制的难点在于地图缩放后箭头不会自动缩放，影响美观，具体的解决方案会在下文给出。



## 2.2.4 自流量的表示

自流量是指从当前站点借车又还向当前站点的数量。在绘制时，在每个点的外画一个圆，圆的颜色编码了自流量的大小。具体可采用 `BMap.Circle()` 的方法绘制，该 API 的 DEMO 可参照官网，颜色的设置可以参照弧线绘制，不过多说明。

## 2.3 遇到的问题及解决

### 2.3.1 箭头的重新绘制

需要达到的目的很简单，就是当地图缩放的时候，重新绘制箭头。难处在于百度地图没有现成的 API 可以调用，所以需要思考一种解决方案。经过思考，我采取了当地图缩放时，清除原有的箭头，再重新绘制。实现方案就是定义一个数组，存放已绘的箭头，用于清除，再定义一个数组，存储需要画箭头的弧线，用于重绘。核心代码如下：

```
$(document).on("mousewheel DOMMouseScroll", function (e) {
    if(e.originalEvent.wheelDelta) {
        arrow.forEach(function(Arrow) {          //存放已绘的箭头
            map.removeOverlay(Arrow);
        })
        arrow = [];
        redraw.forEach(function(obj) {           //存放弧线
            addArrow(obj.Curveline,obj.Opt);
        })
    }
});
```

### 2.3.2 清除无关海量点

由于海量点的构造函数比较特殊，构造函数的返回值不是一个数组，因此，不能通过模仿箭头重绘一样，用遍历的方式清除多余的海量点，我思考后，决定通过清除所有海量点再重绘有关海量点来达到此目的。具体实现思路为定义一个数组，用于存放弧线的端点，在弧线绘制完毕之后，清除所有的海量点，再用刚才的数组生成新海量点。核心代码如下：

```
drawLines.done(function() {
    let options = {
```

```

        size: BMAP_POINT_SIZE_SMALL,
        shape: BMAP_POINT_SHAPE_STAR,
        color: '#d340c3'
    }
    map.removeOverlay(oceanPoints[0]); //数组的第一个值是所有海量点的
集合
    pointCollection = new BMap.PointCollection(pointsNotNeedToClean,
options); //传入新数组
    pointCollection.addEventListener('click', function (e) {
        alert('单击点的坐标为: ' + e.point.lng + ', ' + e.point.lat);
    });
    oceanPoints.push(pointCollection);
    map.addOverlay(pointCollection); // 将标注添加到地图中
});

```

## 2.4 项目的创新部分

### 2.4.1 ES6 特性的使用

我觉得 ES6 的使用，既是创新点，也是难点。ES6 是 JavaScript 的一个革新比较大的版本特性（ES7 和 ES8 已经发布，但只是修修补补 ES6）。我尝试使用 ES6 的闭包，promise，事件循环等特性来完成我的目的，带来了一定的方便，但因为初学，这也花费了我大量时间去定位和修复一些 bug。这个创新部分不能在效果展示中显示，但能体现在代码中，比如：

```

$.getJSON("newMapInit").done(function(stations) {
    addOceanPoints(stations);
}).done(function(stations) {
    $("#submit").click(function() {
        drawLines(stations);
    });
    $("#heatmap").click(function() {
        drawHeatmap(stations);
    })
});

```

这段代码是 index.js 的主干部分，用 promise 的特性来用同步的思维写异步的代码（JavaScript 是异步执行的）。

ES6 的特性几乎体现在我每一行代码上，我花了大量时间去学习和改写，我认为这不仅能提高代码的可读性（比如上文，在获取数据后添加海量点，添加完

后绑定点击事件)，对自身的能力也是一种锻炼，毕竟前端离不开 JS，JS 也难以逃离异步等理念。

### 2.4.2 清除按钮的设置

这个功能的难度本身不大，但当箭头重绘和海量点部分删除功能加入后，实现起来就没那么方便了。但看了代码后，还是很好理解的：

```
$("#clear").click(function() {
    arrow = [];
    redraw = [];
    oceanPoints = [];
    $('.ruler').css("visibility", "false");
    $('.ruler .ruler_content').css("visibility", "hidden");
    map.clearOverlays();
    $.getJSON("newMapInit").done(function(stations) {
        addOceanPoints(stations);
    })
})
```

如上，就是将之前提到过的数组都置空，隐藏标尺，清除所有的覆盖物（此时的覆盖物只有自流量点，弧线和部分海量点），然后重新加载所有海量点。

### 2.5 项目成果展示

图 8：初始页面。图 9：约束条件下的弧线和自流量

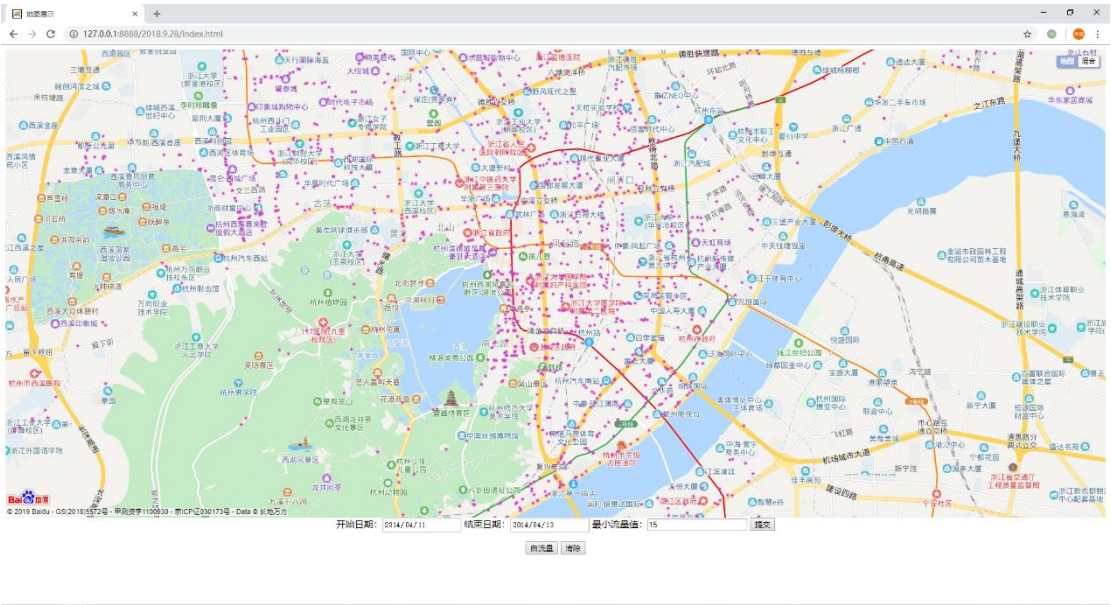


图 8

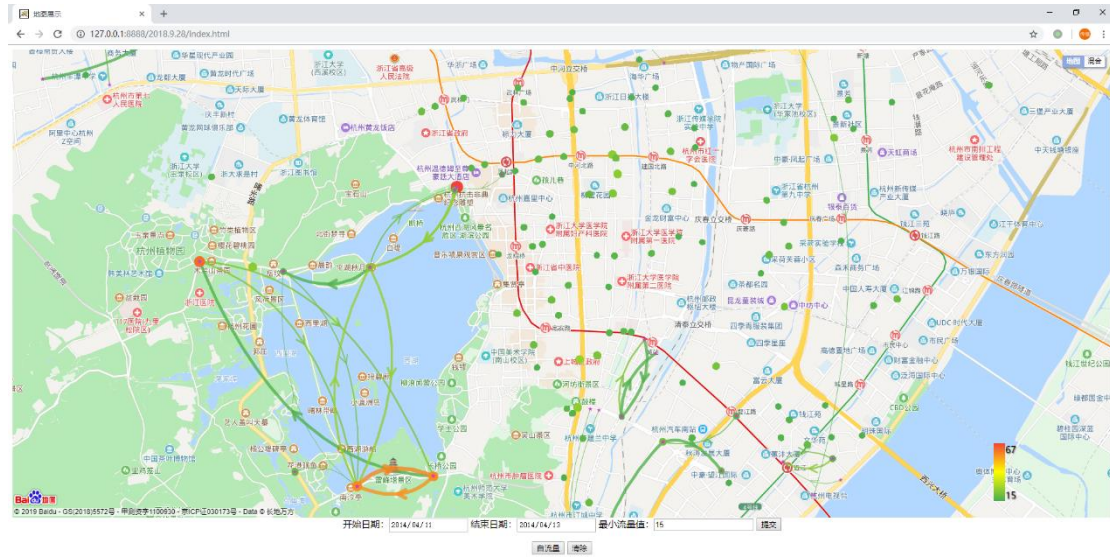


图 9