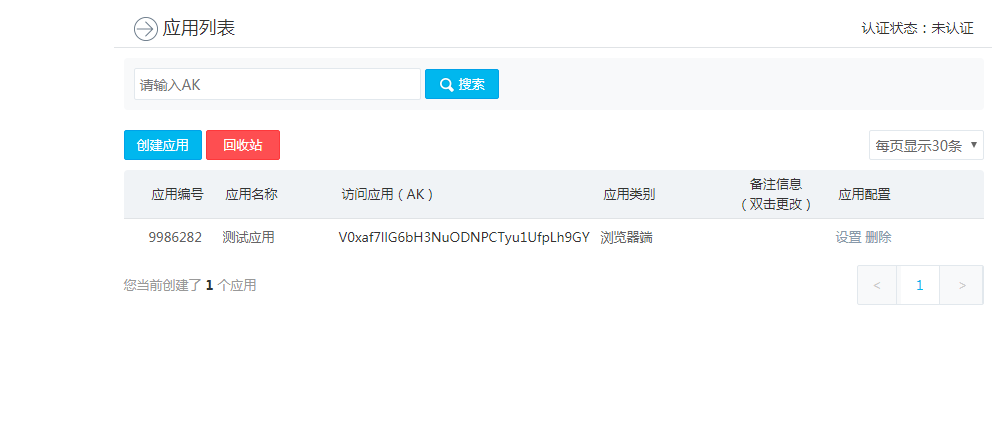
## 百度地图展示

### 百度账号注册以及简单demo

百度地图需要去百度地图开发者网站申请<http://lbsyun.baidu.com/apiconsole/key>，进入这个网站后只要注册百度账号申请过程比较简单



点击创建应用按钮。



该选的都选上：



然后就Ok了就可以进行地图的加载。

### 4.1.2百度地图加载这个比较简单：

新建Html 在代码添加一个div

<div id="mapDiv">

</div>然后使用百度地图的jsapi

引入<script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=V0xaf7llG6bH3NuODNPCTyu1UfpLh9GY"></script>

其中ak=V0xaf7llG6bH3NuODNPCTyu1UfpLh9GY就是我们申请的序列号 如果没有这个号可能在浏览地图的时候就会出现问题了

地图加载的js代码如下：

var map = new BMap.Map("mapDiv");

//map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);

map.centerAndZoom('成都',12);

var top\_left\_control = new BMap.ScaleControl({ anchor: BMAP\_ANCHOR\_BOTTOM\_LEFT }); // 左上角，添加比例尺

//var top\_left\_navigation = new BMap.NavigationControl(); //左上角，添加默认缩放平移控件

var top\_right\_navigation = new BMap.NavigationControl({ anchor: BMAP\_ANCHOR\_TOP\_RIGHT }); //右上角，仅包含平移和缩放按钮 //, type: BMAP\_NAVIGATION\_CONTROL\_SMALL

/\*缩放控件type有四种类型:

BMAP\_NAVIGATION\_CONTROL\_SMALL：仅包含平移和缩放按钮；BMAP\_NAVIGATION\_CONTROL\_PAN:仅包含平移按钮；BMAP\_NAVIGATION\_CONTROL\_ZOOM：仅包含缩放按钮\*/

//添加控件和比例尺

map.addControl(top\_left\_control);

//map.addControl(top\_left\_navigation);

map.addControl(top\_right\_navigation);

map.enableScrollWheelZoom(); //启动鼠标滚轮缩放地图

map.enableKeyboard(); //启动键盘操作地图

为了让地图全屏显示 我们还要加上相应的css代码：

html, body

{

padding: 0;

margin: 0;

height: 100%;

overflow:hidden;

}

#mapDiv

{

z-index:1;

width:100%;

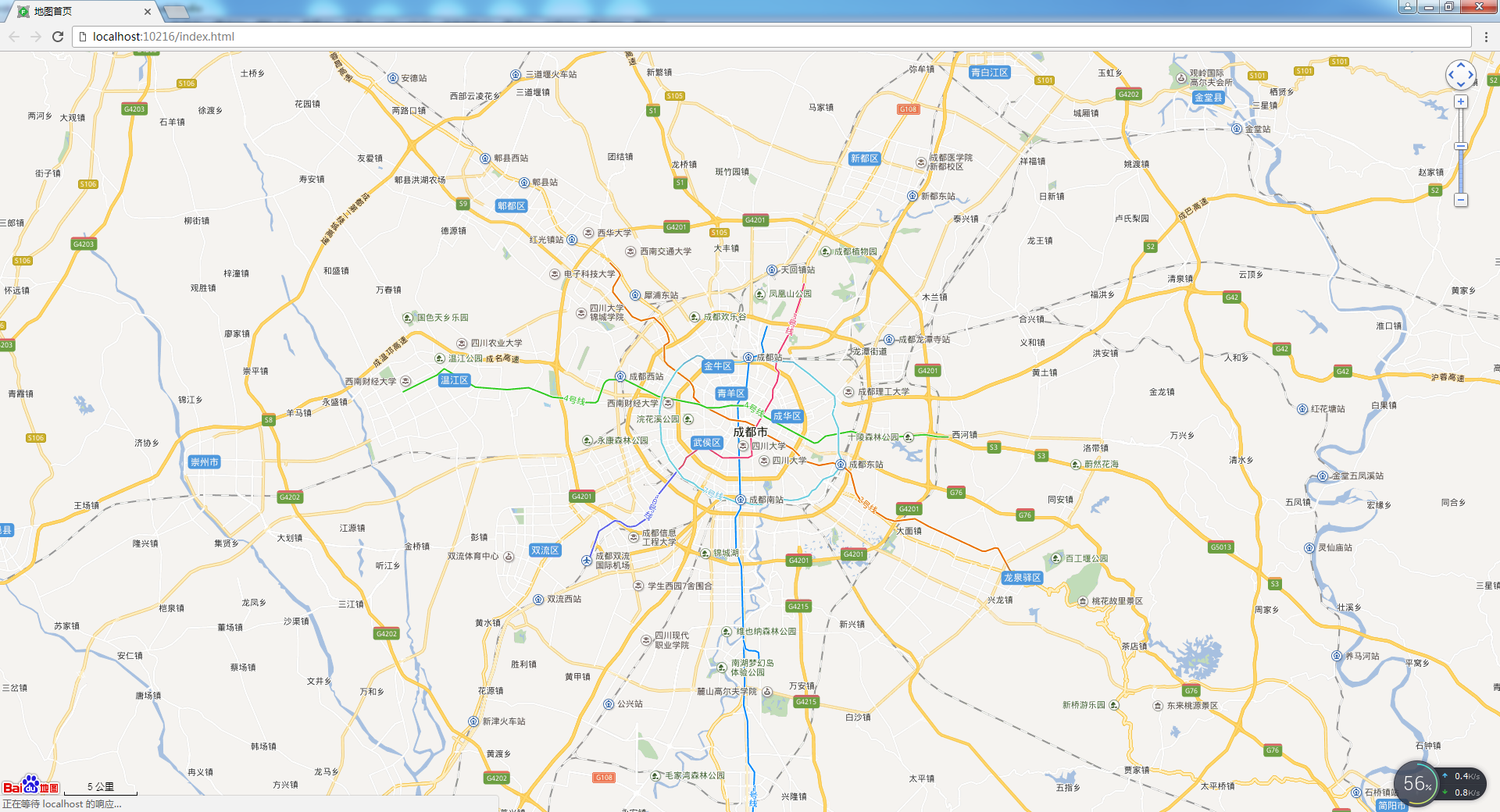
height:100%;

overflow:hidden;

position:absolute;

}

然后我们直接运行程序(VS2010中直接按f5) 在chrome中或者其他浏览器中浏览我们创建的网页就可以打开地图

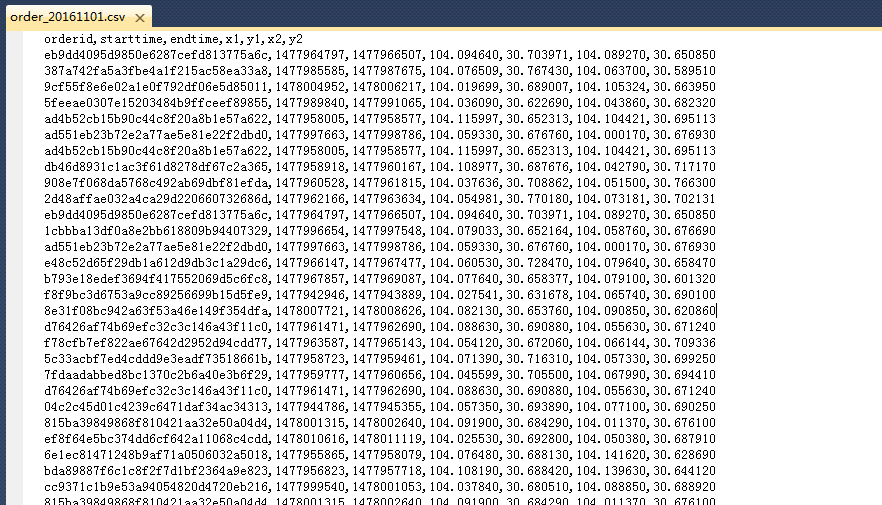


然后我们就可以在这个基础上进行开发了。

### 4.1.3

## 订单数据的读取与加载

## 订单我们使用滴滴官方提供的csv格式，数据结构如下：



### 4.2.2我们使用c#程序可以对csv文件进行解析 生成C#中方便处理的DataTable格式

然后我们就好操作了。C#解析CSV的代码如下：

/// <summary>

/// Stream读取.csv文件

/// </summary>

/// <param name="filePath">文件路径</param>

/// <returns></returns>

public static DataTable OpenCSV(string filePath)

{

DataTable dt = new DataTable();

FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read);

StreamReader sr = new StreamReader(fs, System.Text.Encoding.Default);

//记录每次读取的一行记录

string strLine = "";

//记录每行记录中的各字段内容

string[] aryLine;

//标示列数

int columnCount = 0;

//标示是否是读取的第一行

bool IsFirst = true;

//逐行读取CSV中的数据

while ((strLine=sr.ReadLine())!=null)

{

aryLine = strLine.Split(',');

if (IsFirst==true)

{

IsFirst = false;

columnCount = aryLine.Length;

for (int i = 0; i < columnCount; i++)

{

DataColumn dc = new DataColumn(aryLine[i]);

dt.Columns.Add(dc);

}

}

else

{

DataRow dr = dt.NewRow();

for (int j = 0; j < columnCount; j++)

{

dr[j] = aryLine[j];

}

dt.Rows.Add(dr);

}

}

sr.Close();

fs.Close();

return dt;

}

### 4.2.3

## 对于表格数据的展示前端我们采用jsGrid控件这个控件是开源的相关开发资料都可以从官方网站http://js-grid.com/看到。他的使用也是比较简单只需要指定div就可以进行加载。

$("#mapDiv").jsGrid({

width: "100%",

height: "100%",

paging: true,

pagerFormat: "页码: {first} {prev} {pages} {next} {last} {pageIndex} of {pageCount}",

pagePrevText: "下一页",

pageNextText: "前一页",

noDataContent: "加载中，请稍等。。。",

pageFirstText: "首页",

pageLastText: "尾页",

itemTemplate: function (value, item) { console.log(value); console.log(item); },

fields: [

{ name: "orderid", title: "订单号", type: "text", validate: "required", width: 160 },

{ name: "stime", title: "开始时间", type: "date" },

{ name: "etime", title: "结束时间", type: "date" },

{ name: "x1", title: "开始经度", type: "number", width: 35 },

{ name: "y1", title: "开始纬度", type: "number", width: 35 },

{ name: "x2", title: "结束经度", type: "number", width: 35 },

{ name: "y2", title: "结束纬度", type: "number", width: 35 }

]

});

其中fields就是我们需要显示的列信息。

使用以上代码就可以创建一个空的表格。

这边因为数据需要从后端获取，前端以html+js方式展示。后端程序我们使用asp.net提供的.ashx一般请求处理程序相应。数据请求我们采用post方式 比如我们请求所有数据数据我们使用后端使用C#语言相应前端请求代码如下：

string t = context.Request.QueryString["t"]; //因为其他界面也要调用这个处理程序 所以我们根据t值判断处理那个请求返回相应的值

string csvFile = context.Server.MapPath("order\_20161101.csv");

DataTable dtorders = CSVHelper.OpenCSV(csvFile);

if ("data".Equals(t.ToLower()))

{

//添加新列用于把unix时间戳列转为时间格式

dtorders.Columns.Add(new DataColumn(){ ColumnName="stime",DataType=typeof(System.String)});

dtorders.Columns.Add(new DataColumn() { ColumnName = "etime", DataType = typeof(System.String) });

foreach (DataRow row in dtorders.Rows)

{

row["stime"] = Long2Datetime(Convert.ToInt64(row["starttime"])).ToString("yyyy-MM-dd HH:mm:ss");

row["etime"] = Long2Datetime(Convert.ToInt64(row["endtime"])).ToString("yyyy-MM-dd HH:mm:ss");

}

dtorders.Columns.Remove("starttime");

dtorders.Columns.Remove("endtime");

context.Response.Write(JsonConvert.SerializeObject(dtorders));

return;

}

其中Unix时间戳格式转字符串时间格式代码如下：

public DateTime Long2Datetime(long l)

{

System.DateTime startTime = TimeZone.CurrentTimeZone.ToLocalTime(new System.DateTime(1970, 1, 1)); // 当地时区

DateTime dt = startTime.AddSeconds(l);

return dt;

}

后端代码写好后我们前端就可以使用jquery的ajax方式进行调用调用的连接为data.ashx?t=data

.data.ashx data.ashx?t=data

然后jquery ajax的调用代码如下：

//从后台获取gps数据

var dataurl="./data.ashx?t=data";

$.ajax({

type: "get",

url: dataurl,

dataType: 'json', //WebService 会返回Json类型

success: function (result) { //回调函数，result，返回值

if (result.length > 0) {

var jsgrid = $("#mapDiv").jsGrid({

width: "100%",

height: "100%",

noDataContent: "无数据。。。",

paging: true,

pagerFormat: "页码: {first} {prev} {pages} {next} {last} {pageIndex} of {pageCount}",

pagePrevText: "下一页",

pageNextText: "前一页",

pageFirstText: "首页",

pageLastText: "尾页",

itemTemplate: function (value, item) { console.log(value); console.log(item); },

data: result,

fields: [

{ name: "orderid", title: "订单号", type: "text", validate: "required", width: 160 },

{ name: "stime", title: "开始时间", type: "date" },

{ name: "etime", title: "结束时间", type: "date" },

{ name: "x1", title: "开始经度", type: "number", width: 35 },

{ name: "y1", title: "开始纬度", type: "number", width: 35 },

{ name: "x2", title: "结束经度", type: "number", width: 35 },

{ name: "y2", title: "结束纬度", type: "number", width: 35 }

]

});

}

else {

$("#mapDiv").jsGrid({

width: "100%",

height: "100%",

paging: true,

pagerFormat: "页码: {first} {prev} {pages} {next} {last} {pageIndex} of {pageCount}",

pagePrevText: "下一页",

pageNextText: "前一页",

noDataContent: "加载中，请稍等。。。",

pageFirstText: "首页",

pageLastText: "尾页",

itemTemplate: function (value, item) { console.log(value); console.log(item); },

fields: [

{ name: "orderid", title: "订单号", type: "text", validate: "required", width: 160 },

{ name: "stime", title: "开始时间", type: "date" },

{ name: "etime", title: "结束时间", type: "date" },

{ name: "x1", title: "开始经度", type: "number", width: 35 },

{ name: "y1", title: "开始纬度", type: "number", width: 35 },

{ name: "x2", title: "结束经度", type: "number", width: 35 },

{ name: "y2", title: "结束纬度", type: "number", width: 35 }

]

});

}

},

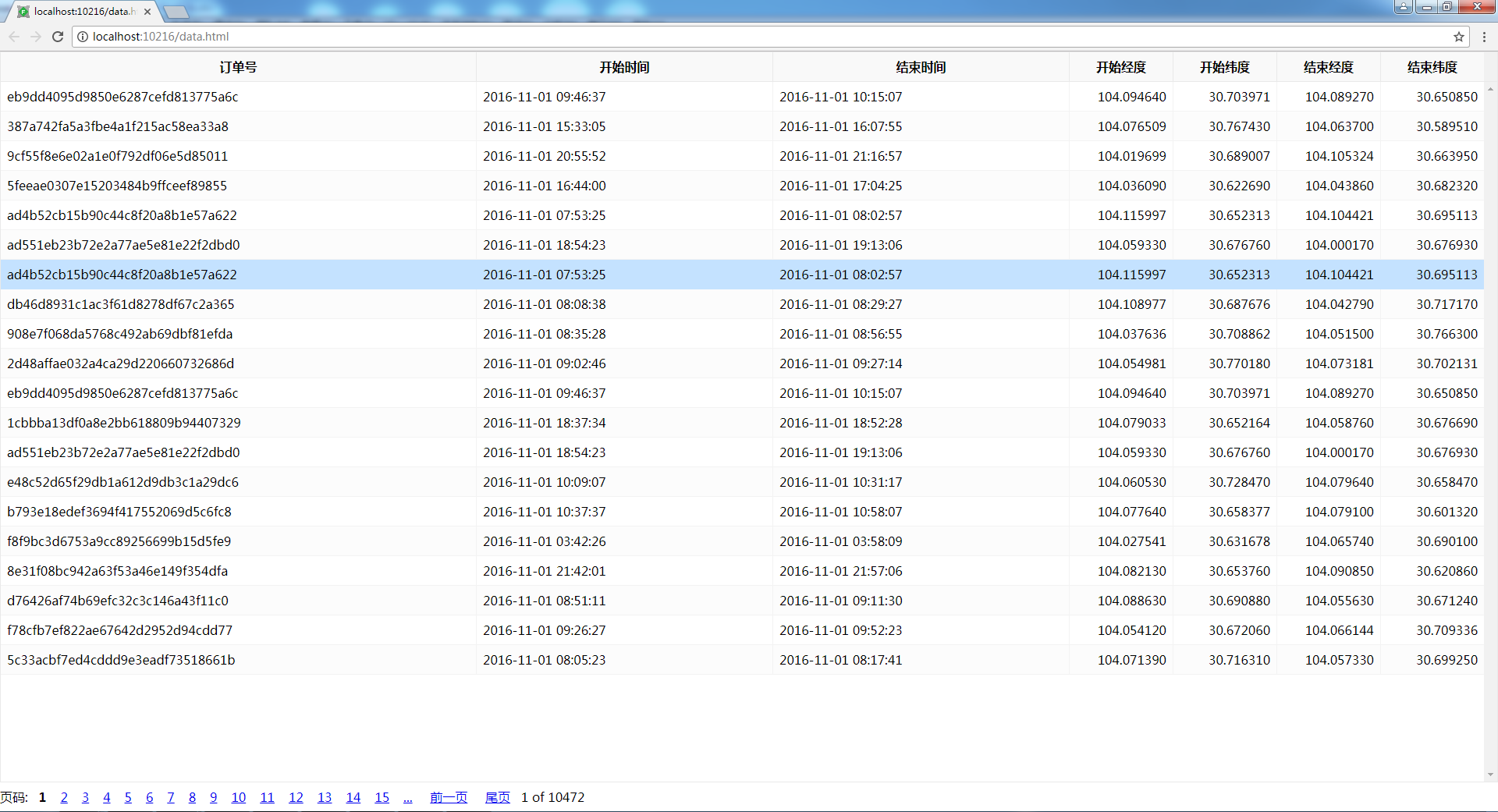
error: function (err) { // 出错s

console.log(err);

}

});

在浏览器中浏览数据效果如下：



## 在地图上展示订单数据

## 获取成都所有公交站点以及公交站点的地图加载

4.4.1前端请求

公交站点数据我们需要调用百度本地化服务查询并输出到控制台：

代码如下：

//查询所有公交站点

var local = new BMap.LocalSearch("成都",

{ renderOptions: { autoViewport: true }, pageCapacity: 100 //map: map,

, onSearchComplete: function (results) {

if (local.getStatus() != BMAP\_STATUS\_SUCCESS) {

// resultStr = "查询异常！";

// console.log

} else {

var totalPages = results.getNumPages();

var currPage = results.getPageIndex(); // 获取当前是第几页数据

// alert("totalResults:" + totalResults);

for (i = 0; i < results.getCurrentNumPois(); i++) {

//console.log(i \* results.getPageIndex());

console.log(results.getPoi(i));

//resultArray[50 \* currPage + i] = results.getPoi(i);//在当前页面下获取页面中的内容

}

if (results.getPageIndex() <= results.getNumPages() - 1) {

//console.log(results.Br);

// for (var i = 0; i < results.Br.length; i++) {

// console.log(results.getPoi(i));

// }

local.gotoPage(results.getPageIndex() + 1); // 遍历到最后一页之后不再进行下一页搜索，此时，已经获取到全部的搜索结果，

}

}

}

});

local.search("公交站");

注意以上代码只需要获取公交站的时候使用就行其他时间不需要执行。

然后执行 我们在浏览器的控制台就能获取到数据



然后对数据进行整理输出到一个json文件

4.4.2后端数据处理返回

4.4.3界面效果

## 上下车公交区间分析

## 按照时间段分析

## 公交站点订单统计

## 公交站点订单统计图

## 公交区间订单分析

## 公交区间订单分析图形化展示

