AJAX中的JSON与JSONP的区别与用法详解

由于Sencha Touch 2这种开发模式的特性，基本决定了它原生的数据交互行为几乎只能通过AJAX来实现。

当然了，通过调用强大的PhoneGap插件然后打包，你可以实现100%的Socket通讯和本地数据库功能，又或者通过HTML5的WebSocket也可以实现与服务器的通讯和服务端推功能，但这两种方式都有其局限性，前者需要PhoneGap支持，后者要求用户设备必须支持WebSocket，因此都不能算是ST2的原生解决方案，原生的只有AJAX。

说到AJAX就会不可避免的面临两个问题，第一个是AJAX以何种格式来交换数据？第二个是跨域的需求如何解决？这两个问题目前都有不同的解决方案，比如数据可以用自定义字符串或者用XML来描述，跨域可以通过服务器端代理来解决。

但到目前为止最被推崇或者说首选的方案还是用JSON来传数据，靠JSONP来跨域。而这就是本文将要讲述的内容。

JSON和JSONP虽然只有一个字母的差别，但其实他们根本不是一回事儿：JSON是一种数据交换格式，而JSONP是一种依靠开发人员的聪明才智创造出的一种非官方跨域数据交互协议。我们拿最近比较火的谍战片来打个比方，JSON是地下党们用来书写和交换情报的“暗号”，而JSONP则是把用暗号书写的情报传递给自己同志时使用的接头方式。看到没？一个是描述信息的格式，一个是信息传递双方约定的方法。

既然随便聊聊，那我们就不再采用教条的方式来讲述，而是把关注重心放在帮助开发人员理解是否应当选择使用以及如何使用上。

小小的广告一下，该篇文章是在自己群里与Sencha Touch 2的开发者们一起探讨ST2数据交互模型时有感而发写出来的 。

什么是JSON？

前面简单说了一下，JSON是一种基于文本的数据交换方式，或者叫做数据描述格式，你是否该选用他首先肯定要关注它所拥有的优点。

JSON的优点：

1、基于纯文本，跨平台传递极其简单；

2、Javascript原生支持，后台语言几乎全部支持；

3、轻量级数据格式，占用字符数量极少，特别适合互联网传递；

4、可读性较强，虽然比不上XML那么一目了然，但在合理的依次缩进之后还是很容易识别的；

5、容易编写和解析，当然前提是你要知道数据结构；

JSON的缺点当然也有，但在作者看来实在是无关紧要的东西，所以不再单独说明。

JSON的格式或者叫规则：

JSON能够以非常简单的方式来描述数据结构，XML能做的它都能做，因此在跨平台方面两者完全不分伯仲。

1、JSON只有两种数据类型描述符，大括号{}和方括号[]，其余英文冒号:是映射符，英文逗号,是分隔符，英文双引号""是定义符。

2、大括号{}用来描述一组“不同类型的无序键值对集合”（每个键值对可以理解为OOP的属性描述），方括号[]用来描述一组“相同类型的有序数据集合”（可对应OOP的数组）。

3、上述两种集合中若有多个子项，则通过英文逗号,进行分隔。

4、键值对以英文冒号:进行分隔，并且建议键名都加上英文双引号""，以便于不同语言的解析。

5、JSON内部常用数据类型无非就是字符串、数字、布尔、日期、null 这么几个，字符串必须用双引号引起来，其余的都不用，日期类型比较特殊，这里就不展开讲述了，只是建议如果客户端没有按日期排序功能需求的话，那么把日期时间直接作为字符串传递就好，可以省去很多麻烦。

JSON实例：

```
// 描述一个人

var person = {
    "Name": "Bob",
    "Age": 32,
    "Company": "IBM",
    "Engineer": true
}

// 获取这个人的信息
```

```
var personAge = person.Age;

// 描述几个人

var members = [
    {
        "Name": "Bob",
        "Age": 32,
        "Company": "IBM",
        "Engineer": true
    },
    {
        "Name": "John",
        "Age": 20,
        "Company": "Oracle",
        "Engineer": false
    },
    {
        "Name": "Henry",
        "Age": 45,
        "Company": "Microsoft",
        "Engineer": false
    }
]

// 读取其中John的公司名称

var johnsCompany = members[1].Company;

// 描述一次会议

var conference = {
    "Conference": "Future Marketing",
    "Date": "2012-6-1",
    "Address": "Beijing",
    "Members":
    [
        {
            "Name": "Bob",
            "Age": 32,
            "Company": "IBM",
            "Engineer": true
        },
        {
            "Name": "John",
            "Age": 20,
            "Company": "Oracle",
            "Engineer": false
        },
        {
            "Name": "Henry",
            "Age": 45,
            "Company": "Microsoft",
            "Engineer": false
        }
    ]
}

// 读取参会者Henry是否工程师

var henryIsAnEngineer = conference.Members[2].Engineer;
```

关于JSON，就说这么多，更多细节请在开发过程中查阅资料深入学习。

什么是JSONP？

先说说JSONP是怎么产生的：

其实网上关于JSONP的讲解有很多，但却千篇一律，而且云里雾里，对于很多刚接触的人来讲理解起来有些困难，小可不才，试着用自己的方式来阐释一下这个问题，看看是否有帮助。

1、一个众所周知的问题，Ajax直接请求普通文件存在跨域无权限访问的问题，甭管你是静态页面、动态网页、web服务、WCF，只要是跨域请求，一律不准；

2、不过我们又发现，Web页面上调用js文件时则不受是否跨域的影响（不仅如此，我们还发现凡是拥有"src"这个属性的标签都拥有跨域的能力，比如<script>、<img>、<iframe>）；

3、于是可以判断，当前阶段如果想通过纯web端（ActiveX控件、服务端代理、属于未来的HTML5之Websocket等方式不算）跨域访问数据就只有一种可能，那就是在远程服务器上设法把数据装进js格式的文件里，供客户端调用和进一步处理；

4、恰巧我们已经知道有一种叫做JSON的纯字符数据格式可以简洁的描述复杂数据，更妙的是JSON还被js原生支持，所以在客户端几乎可以随心所欲的处理这种格式的数据；

5、这样子解决方案就呼之欲出了，web客户端通过与调用脚本一模一样的方式，来调用跨域服务器上动态生成的js格式文件（一般以JSON为后缀），显而易见，服务器之所以要动态生成JSON文件，目的就在于把客户端需要的数据装入进去。

6、客户端在对JSON文件调用成功之后，也就获得了自己所需的数据，剩下的就是按照自己需求进行处理和展现了，这种获取远程数据的方式看起来非常像AJAX，但其实并不一样。

7、为了便于客户端使用数据，逐渐形成了一种非正式传输协议，人们把它称作JSONP，该协议的一个要点就是允许用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为函数名来包裹住JSON数据，这样客户端就可以随意定制自己的函数来自动处理返回数据了。

如果对于callback参数如何使用还有些模糊的话，我们后面会有具体的实例来讲解。

JSONP的客户端具体实现：

不管jQuery也好，extjs也罢，又或者是其他支持jsonp的框架，他们幕后所做的工作都是一样的，下面我来循序渐进的说明一下jsonp在客户端的实现：

1、我们知道，哪怕跨域js文件中的代码（当然指符合web脚本安全策略的），web页面也是可以无条件执行的。

远程服务器remoteserver.com根目录下有个remote.js文件代码如下：

```
alert('我是远程文件');
```

本地服务器localserver.com下有个jsonp.html页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript" src="http://remoteserver.com/remote.js"></script>
</head>
<body>

</body>
</html>
```

毫无疑问，页面将会弹出一个提示窗体，显示跨域调用成功。

2、现在我们在jsonp.html页面定义一个函数，然后在远程remote.js中传入数据进行调用。

jsonp.html页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
    var localHandler = function(data){
        alert('我是本地函数，可以被跨域的remote.js文件调用，远程js带来的数据是: ' + data.result);
    };
    </script>
    <script type="text/javascript" src="http://remoteserver.com/remote.js"></script>
</head>
<body>

</body>
</html>
```

remote.js文件代码如下：

```
localHandler({"result":"我是远程js带来的数据"});
```

运行之后查看结果，页面成功弹出提示窗口，显示本地函数被跨域的远程js调用成功，并且还接收到了远程js带来的数据。很欣喜，跨域远程获取数据的目的基本实现了，但是又一个问题出现了，我怎么让远程js知道它应该调用的本地函数叫什么名字呢？毕竟是jsonp的服务者都要面对很多服务对象，而这些服务对象各自的本地函数都不相同啊？我们接着往下看。

3、聪明的开发者很容易想到，只要服务端提供的js脚本是动态生成的就行了呗，这样调用者可以传一个参数过去告诉服务端"我想要一段调用XXX函数的js代码，请你返回给我"，于是服务器就可以按照客户端的需求来生成js脚本并响应了。

看jsonp.html页面的代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
    // 得到航班信息查询结果后的回调函数
    var flightHandler = function(data){
        alert('你查询的航班结果是：票价 ' + data.price + ' 元，' + '余票 ' + data.tickets + ' 张。');
    };
    // 提供jsonp服务的url地址（不管是什么类型的地址，最终生成的返回值都是一段javascript代码）
    var url = "http://flightQuery.com/jsonp/flightResult.aspx?code=CA1998&callback=flightHandler";
    // 创建script标签，设置其属性
    var script = document.createElement('script');
    script.setAttribute('src', url);
    // 把script标签加入head，此时调用开始
    document.getElementsByTagName('head')[0].appendChild(script);
    </script>
</head>
<body>

</body>
</html>
```

这次的代码变化比较大，不再直接把远程js文件写死，而是编码实现动态查询，而这也正是jsonp客户端实现的核心部分，本例中的重点也就在于如何完成jsonp调用的全过程。

我们看到调用的url中传递了一个code参数，告诉服务器我要查的是CA1998次航班的信息，而callback参数则告诉服务器，我的本地回调函数叫做flightHandler，所以请把查询结果传入这个函数中进行调用。

OK，服务器很聪明，这个叫做flightResult.aspx的页面生成了一段这样的代码提供给jsonp.html（服务端的实现这里就不演示了，与你选用的语言无关，说到底就是拼接字符串）：

```
flightHandler({
    "code": "CA1998",
    "price": 1780,
    "tickets": 5
});
```

我们看到，传递给flightHandler函数的是一个json，它描述了航班的基本信息。运行一下页面，成功弹出提示窗口，jsonp的执行全过程顺利完成！

4、到这里为止的话，相信你已经能够理解jsonp的客户端实现原理了吧？剩下的就是如何把代码封装一下，以便于与用户界面交互，从而实现多次和重复调用。

什么？你用的是jQuery，想知道jQuery如何实现jsonp调用？好吧，那我就好人做到底，再给你一段jQuery使用jsonp的代码（我们依然沿用上面那个航班信息查询的例子，假定返回jsonp结果不变）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
>
 <html xmlns="http://www.w3.org/1999/xhtml" >
 <head>
     <title>Untitled Page</title>
      <script type="text/javascript" src=jquery.min.js"></script>
      <script type="text/javascript">
     jQuery(document).ready(function(){
        $.ajax({
             type: "get",
             async: false,
             url: "http://flightQuery.com/jsonp/flightResult.aspx?code=CA1998",
             dataType: "jsonp",
             jsonp: "callback",//传递给请求处理程序或页面的，用以获得jsonp回调函数名的参数名(一般默认为:callback)
             jsonpCallback:"flightHandler",//自定义的jsonp回调函数名称，默认为jQuery自动生成的随机函数名，也可以写"?"，jQuery会自
动为你处理数据
             success: function(json){
                 alert('您查询到航班信息：票价：  ' + json.price + ' 元，余票：  ' + json.tickets + ' 张。');
             },
             error: function(){
                 alert('fail');
             }
         });
     });
     </script>
     </head>
  <body>
  </body>
 </html>
```

是不是有点奇怪？为什么我这次没有写flightHandler这个函数呢？而且竟然也运行成功了！哈哈，这就是jQuery的功劳了，jquery在处理jsonp类型的ajax时（还是忍不住吐槽，虽然jquery也把jsonp归入了ajax，但其实它们真的不是一回事儿），自动帮你生成回调函数并把数据取出来供success属性方法来调用，是不是很爽呀？

好啦，写到这里，我已经无力再写下去，又困又累，得赶紧睡觉。朋友们要是看这不错，觉得有启发，给点个"推荐"呗！由于实在比较简单，所以就不再提供demo源码下载了。

4月20日下午的补充：

没想到上了博客园的头条推荐。看到大家对这篇文章的认可和评论，还是很开心的，这里针对ajax与jsonp的异同再做一些补充说明：

1、ajax和jsonp这两种技术在调用方式上"看起来"很像，目的也一样，都是请求一个url，然后把服务器返回的数据进行处理，因此jquery和ext等框架都把jsonp作为ajax的一种形式进行了封装；

2、但ajax和jsonp其实本质上是不同的东西。ajax的核心是通过XmlHttpRequest获取非本页内容，而jsonp的核心则是动态添加<script>标签来调用服务器提供的js脚本。

3、所以说，其实ajax与jsonp的区别不在于是否跨域，ajax通过服务端代理一样可以实现跨域，jsonp本身也不排斥同域的数据的获取。

4、还有就是，jsonp是一种方式或者说非强制性协议，如同ajax一样，它也不一定非要用json格式来传递数据，如果你愿意，字符串都行，只不过这样不利于用jsonp提供公开服务。

总而言之，jsonp不是ajax的一个特例，哪怕jquery等巨头把jsonp封装进了ajax，也不能改变着一点！

原文链接：http://www.cnblogs.com/dowinning/archive/2012/04/19/json-jsonp-jquery.html#!comments

摘录时间：2015-10-29 17：48 - JosonLiu