[**优化网站设计（十四）：使AJAX调用尽可能利用缓存特性**](http://www.cnblogs.com/chenxizhang/archive/2013/05/12/3073690.html)

**前言**

网站设计的优化是一个很大的话题,有一些通用的原则,也有针对不同开发平台的一些建议。这方面的研究一直没有停止过，我在不同的场合也分享过这样的话题。

作为通用的原则，雅虎的工程师团队曾经给出过35个最佳实践。这个列表请参考**Best Practices for Speeding Up Your Web Site** <http://developer.yahoo.com/performance/rules.html>，同时，他们还发布了一个相应的测试工具Yslow <http://developer.yahoo.com/yslow/>

我强烈推荐所有的网站开发人员都应该学习这些最佳实践，并结合自己的实际项目情况进行应用。 接下来的一段时间，我将结合ASP.NET这个开发平台，针对这些原则，通过一个系列文章的形式，做些讲解和演绎，以帮助大家更好地理解这些原则，并且更好地使用他们。

**准备工作**

为了跟随我进行后续的学习，你需要准备如下的开发环境和工具

1. Google Chrome 或者firefox ，并且安装 Yslow这个扩展组件.请注意，这个组件是雅虎提供的，但目前没有针对IE的版本。
   1. <https://chrome.google.com/webstore/detail/yslow/ninejjcohidippngpapiilnmkgllmakh>
   2. <https://addons.mozilla.org/en-US/firefox/addon/yslow/>
   3. 你应该对这些浏览器的开发人员工具有所了解，你可以通过按下F12键调出这个工具。
2. Visaul Studio 2010 SP1 或更高版本，推荐使用Visual Studio 2012
   1. <http://www.microsoft.com/visualstudio/eng/downloads>
3. 你需要对ASP.NET的开发基本流程和核心技术有相当的了解，本系列文章很难对基础知识做普及。

**本文要讨论的话题**

这一篇我和大家讨论的是第十四条原则：[Make Ajax Cacheable](http://developer.yahoo.com/performance/rules.html#cacheajax) （使AJAX调用尽可能利用缓存特性）。

**AJAX的基本概念**

1. AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。
2. AJAX 不是新的编程语言，而是一种使用现有标准的新方法。
3. AJAX 是与服务器交换数据并更新部分网页的艺术，在不重新加载整个页面的情况下。

**AJAX的典型应用场景**

AJAX在目前的应用程序中使用非常广泛，为网站提供了更加丰富的效果（虽然技术很早就有，但最早引起大家注意是在2004年左右的Gmail中）。其典型的应用场景包括

1. 异步加载，使得页面的内容可以分批加载。
2. 局部更新，使得页面的局部更新不会导致页面的刷新。

由于AJAX其实也是需要发起请求，然后服务器执行，并将结果（通常是[JSON](http://json.org/)格式的）发送给浏览器进行最后的呈现或者处理，所以对于网站设计优化的角度而言，我们同样需要考虑对这些请求，是否可以尽可能地利用到缓存的功能来提高性能。

【备注】关于AJAX，以及它与目前的一些技术（主要是服务器端的技术）如何结合的文档，我之前写过很多，有兴趣的朋友可以先参考一下 <http://www.google.ee/search?q=site%3Awww.cnblogs.com%2Fchenxizhang%2F%20ajax>

**什么样的AJAX请求可以被缓存？**

对服务器请求进行优化的方法有很多，我之前已经写过几篇，这些原则也可以应用在AJAX的场景中

1. [优化网站设计（三）：对资源添加缓存控制](http://www.cnblogs.com/chenxizhang/archive/2013/04/30/3052440.html)
2. [优化网站设计（四）：对资源启用压缩](http://www.cnblogs.com/chenxizhang/archive/2013/05/01/3052881.html)
3. [优化网站设计（九）：减少DNS查找的次数](http://www.cnblogs.com/chenxizhang/archive/2013/05/02/3053996.html)
4. [优化网站设计（十）：最小化JAVASCRIPT和CSS](http://www.cnblogs.com/chenxizhang/archive/2013/05/02/3054141.html)
5. [优化网站设计（十一）：避免重定向](http://www.cnblogs.com/chenxizhang/archive/2013/05/05/3060804.html)
6. [优化网站设计（十三）：配置ETags](http://www.cnblogs.com/chenxizhang/archive/2013/05/11/3072898.html)

但是，对于AJAX而言，有一些特殊性，并不是所有的AJAX请求都是可以缓存的。这是由于AJAX的请求通常有两种不同的方法：POST和GET。他们在进行请求的时候，就会略有不同。

1. **POST的请求，是不可以在客户端缓存的**，每次请求都需要发送给服务器进行处理，每次都会返回状态码200。（这里可以优化的是，服务器端对数据进行缓存，以便提高处理速度）
2. **GET的请求，是可以（而且默认）在客户端进行缓存的**，除非指定了不同的地址，否则同一个地址的AJAX请求，不会重复在服务器执行，而是返回304。

**针对POST的情况如何优化**

POST的请求，浏览器通常会假定用户是想要提交（或者发送）数据给服务器，既然如此，那么浏览器自然就不会对该请求进行缓存，因为你是提交数据，所以它认为服务器自然每次都是需要处理的。我们可以来看一个例子。

using System;

using System.Web.Services;

namespace WebApplication4

{

/// <summary>

/// Summary description for HelloWebService

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

**[System.Web.Script.Services.ScriptService]**

public class HelloWebService : System.Web.Services.WebService

{

[WebMethod]

public string HelloWorld()

{

return string.Format("Hello,world -- {0}", DateTime.Now);

}

}

}

上面是一个简单的XML Web Service的定义。需要注意的是，如果希望支持AJAX访问的话，必须要添加ScriptService这个Attribute。

我们的调用代码如下：

//XML Web Service只支持POST，这种方式无法在浏览器中缓存，但可以结合服务器端的缓存，减少后台代码执行的次数来提高性能

$("#btCallXMLWebService").click(function () {

$.ajax({

type: "**POST**",

contentType: "application/json;utf-8",

url: "HelloWebService.asmx/HelloWorld",

data: null,

dataType: "json",

success: function (result) {

alert(result.d);

}

});

});

运行起来之后，我们多次点击这个按钮，会截获到如下的请求：

[](http://images.cnitblog.com/blog/9072/201305/12092618-4b8cedb73e05414994296742d26ba57f.png)

根据上面的截图不难看出，其实每次都请求都是重新被处理过的，它们都是返回状态码为200。

这就是POST AJAX请求的处理情况，它不会被客户端缓存。那你可能会说，能不能将type改为GET呢？例如下面这样

$("#btCallXMLWebService").click(function () {

$.ajax({

type: "**GET**",

contentType: "application/json;utf-8",

url: "HelloWebService.asmx/HelloWorld",

data: null,

dataType: "json",

success: function (result) {

alert(result.d);

}

});

});

答案是，针对XML Web Service或者标准的WCF服务，它们不支持通过GET进行请求，只支持POST请求。

[image](http://images.cnitblog.com/blog/9072/201305/12092619-198b3f6ba4b345658bc6940626de84f1.png)

服务器返回了状态码为500的错误，并且在正文里面描述了这个错误的信息，如下图所示

[](http://images.cnitblog.com/blog/9072/201305/12092619-b2c946c46ec04d44a5167d7f2ebb3803.png)

那么，针对这种场景，我们是否有什么方法进行优化呢？

using System;

using System.Web.Services;

namespace WebApplication4

{

/// <summary>

/// Summary description for HelloWebService

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

[System.Web.Script.Services.ScriptService]

public class HelloWebService : System.Web.Services.WebService

{

/// <summary>

/// 该方法被缓存了10秒钟，是将结果缓存在服务器内存中。

/// </summary>

/// <returns></returns>

**[WebMethod(CacheDuration=10)]**

public string HelloWorld()

{

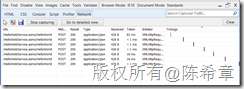
return string.Format("Hello,world -- {0}", DateTime.Now);

}

}

}

这样修改之后，对于客户端而言，其实没有什么改变的，多次调用的时候，服务器都需要处理，然后返回状态码为200。但是区别是什么呢？区别在于服务器并不需要每次都运行真正的代码，它将结果缓存在内存中，在10秒之内重复调用，就直接返回该内存中的数据即可。（这样可以提高服务器的性能，提高并发性）

[](http://images.cnitblog.com/blog/9072/201305/12092620-0e0c8c6841c44f04adf765a3f62c8cbd.png)

【备注】如果是WCF来做服务的话，默认是不支持直接对操作进行缓存的。

**如何设计支持GET的服务**

我们了解到默认情况下，XML Web Service和WCF服务，都只支持使用POST方法的AJAX调用。那么是否有办法设计出来一个支持GET的服务呢？

1. XML Web Service不支持GET
2. WCF服务，可以通过一个特殊的webHttpBinding支持GET。本文将讨论这一种做法。

【备注】WCF有多种适用场景，我之前写过两篇文章，有兴趣的朋友可以参考

[***WCF技术的不同应用场景及其实现分析***](http://www.google.ee/url?sa=t&rct=j&q=site%3Awww.cnblogs.com%2Fchenxizhang%2F%20wcf&source=web&cd=2&ved=0CDgQFjAB&url=%68%74%74%70%3a%2f%2f%77%77%77%2e%63%6e%62%6c%6f%67%73%2e%63%6f%6d%2f%63%68%65%6e%78%69%7a%68%61%6e%67%2f%61%72%63%68%69%76%65%2f%32%30%31%31%2f%30%36%2f%32%31%2f%32%30%38%35%39%37%39%2e%68%74%6d%6c&ei=CeqOUbr2ENG6lQX7-YGIDA&usg=AFQjCNGpyzIfRxZwoJ1gckpWFXFGHLXMig)

[***WCF技术的不同应用场景及其实现分析(续）***](http://www.cnblogs.com/chenxizhang/archive/2011/06/21/2086314.html)

3. ASP.NET MVC中可以支持Web API这个功能，可以通过GET的方式进行调用。这个的做法，本文不做探讨，有兴趣的朋友可以参考 <http://www.asp.net/web-api>

我们来看一个例子。在WCF中支持一种特殊的Operation，就是WebGet

using System;

using System.ServiceModel;

using System.ServiceModel.Activation;

using System.ServiceModel.Web;

namespace WebApplication4

{

[ServiceContract(Namespace = "")]

**[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]**

public class HelloWCFService

{

**[WebGet(ResponseFormat = WebMessageFormat.Json)]**

public string RestfulHelloWorldWithParameter(string name)

{

return string.Format("Hello,{0} -- {1}", DateTime.Now, name);

}

}

}

AJAX调用代码如下

$("#btCallRestfulWCFServicebwithParameter").click(function () {

var name = $("#txtName").val();

//GET请求默认就是会被缓存（在同一个浏览器中，默认是临时缓存，浏览器一关闭就删除掉）

$.getJSON("HelloWCFService.svc/RestfulHelloWorldWithParameter", { name: name }, function (data) {

alert(data.d);

});

});

运行起来之后，我们分别输入不同的name参数，并且分别调用两次。

[](http://images.cnitblog.com/blog/9072/201305/12092621-bab4f48a7bde47d1b5d81c2d5840888a.png)

我们可以发现，第一次调用会返回状态码（200），而第二次调用会返回状态码（304），但如果参数不一样，又会返回状态码（200）。依次类推。

我们也确实可以在浏览器缓存中找到两份缓存的数据

[image](http://images.cnitblog.com/blog/9072/201305/12092623-3153ef73696047b7af49159fe8abda15.png)

所以对于GET请求，默认就会被缓存。但是，如果你想改变这个行为，例如你有时候不想做缓存，应该如何来实现呢？

**避免对GET请求做缓存**

有的时候，我们可能希望GET请求不被缓存，有几种做法来达到这样的目的。

1. 每次调用的时候，请求不同的地址（可以在原始地址后面添加一个随机的号码）例如下面这样：

$("#btCallRestfulWCFServicebwithParameter").click(function () {

var name = $("#txtName").val();

//GET请求默认就是会被缓存（在同一个浏览器中，默认是临时缓存，浏览器一关闭就删除掉）

$.getJSON("HelloWCFService.svc/RestfulHelloWorldWithParameter", { name: name,**version:Math.random()** }, function (data) {

alert(data.d);

});

});

1. 如果你所使用的是jquery的话，则可以考虑禁用AJAX的缓存

$.ajaxSetup({ cache: false });

**使用HTML5的新特性来减少不必要的AJAX调用**

我觉得一个比较彻底的做法是，考虑将一部分数据缓存在客户端中，而且最好不要在临时文件夹中，以便下次启动时还能使用到这些数据。HTML 5中提供了一个新的特性：local storage，可以很好地解决这个问题，如果有兴趣的朋友可以参考下面的文档

1. <http://w3school.com.cn/html5/html_5_webstorage.asp>