**[HttpClient + ASP.NET Web API, WCF之外的另一个选择](http://www.cnblogs.com/dudu/archive/2012/05/11/asp_net_webapi_httpclient.html)**

WCF的野心造成了它的庞大复杂，HTTP的单纯造就了它的简单优美。为了实现分布式Web应用，我们不得不将两者凑合在一起 —— WCF服务以HTTP绑定宿主于IIS。

于是有了让人晕头转向的配置、让人郁闷不已的调试，还有那ServiceContract, DataContract, EnumMember...还有还有，[不要在using语句中调用WCF服务](http://www.cnblogs.com/dudu/archive/2011/01/18/1938144.html)...

于是经常自问：拿着牛刀削苹果有必要吗？废话，当然没有必要，水果刀在哪里？

微软看着这么多人拿着牛刀削苹果，自己也看不下去了，于是，一种水果刀横空出世 —— ASP.NET Web API。

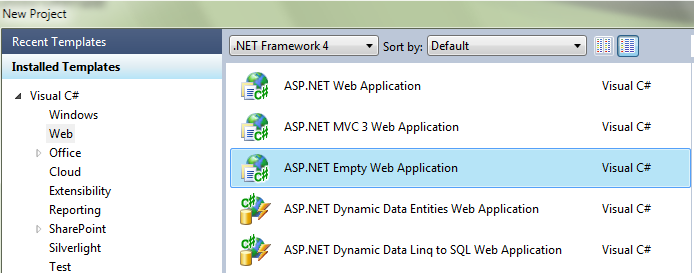
最近我们在实际开发中有个地方用WCF太麻烦，就小试了一下水果刀，感觉还不错。

下面用一个简单的示例分享一下ASP.NET Web API水果刀的用法。

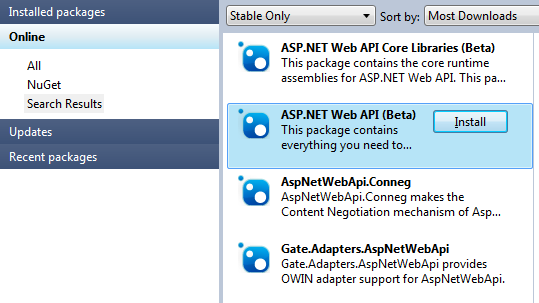
**服务端ASP.NET Web API的实现**

需要准备的工具：Visual Studio 2010, NuGet

1. 新建一个空的ASP.NET Web Application项目。



2. 通过NuGet添加ASP.NET Web API的引用，在NuGet中搜索时要用“AspNetWebApi”（用“ASP.NET Web API”是搜索不到的），然后选择ASP.NET Web API(Beta)进行安装。



3. 添加Global.asax，在Application\_Start中注册Web API的路由，在Global.asax.cs中添加如下代码：

protected void Application\_Start(object sender, EventArgs e)

{

RouteTable.Routes.MapHttpRoute("WebApi", "api/{controller}/{action}/{id}",

new { id = RouteParameter.Optional });

}

4. 添加Controllers文件夹，在其中添加类文件DemoController.cs，并让DemoController继承自ApiController。代码如下：

[复制代码](javascript:void(0);)

namespace CNBlogsWebApiDemo.Controllers

{

public class DemoController : ApiController

{

}

}

[复制代码](javascript:void(0);)

5. 添加ViewModels文件夹，在其中添加Site.cs，并定义Site。

[复制代码](javascript:void(0);)

namespace CNBlogsWebApiDemo.ViewModels

{

public class Site

{

public int SiteId { get; set; }

public string Title { get; set; }

public string Uri { get; set; }

}

}

[复制代码](javascript:void(0);)

6. 给DemoController添加一个方法SiteList，并写上我们的示例代码。代码如下：

[复制代码](javascript:void(0);)

public class DemoController : ApiController

{

public IList<Site> SiteList(int startId, int itemcount)

{

var sites = new List<Site>();

sites.Add(new Site { SiteId = 1, Title = "test", Uri = "www.cnblogs.cc" });

sites.Add(new Site { SiteId = 2, Title = "博客园首页", Uri = "www.cnblogs.com" });

sites.Add(new Site { SiteId = 3, Title = "博问", Uri = "q.cnblogs.com" });

sites.Add(new Site { SiteId = 4, Title = "新闻", Uri = "news.cnblogs.com" });

sites.Add(new Site { SiteId = 5, Title = "招聘", Uri = "job.cnblogs.com" });

var result = (from Site site in sites

where site.SiteId > startId

select site)

.Take(itemcount)

.ToList();

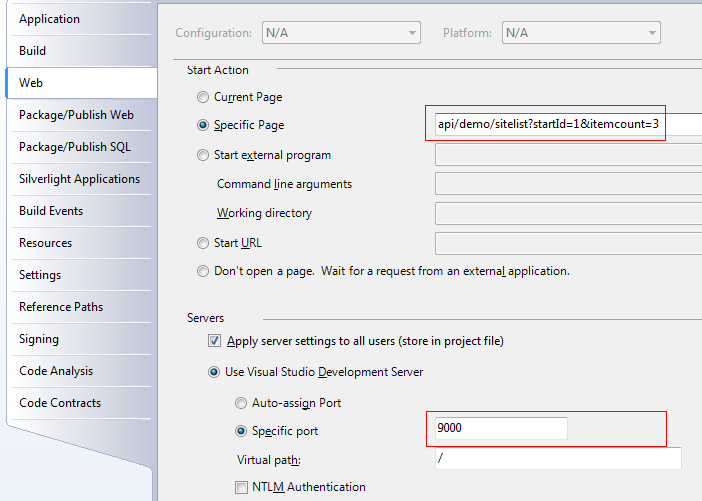
return result;

}

}

[复制代码](javascript:void(0);)

7. 配置一下Web项目的启动设置Specific Page与Specific port



8. Ctrl+F5运行项目，结果如下：



结果是我们期望的，用浏览器直接可以查看Web API的运行结果，测试时会很方便。

好了，服务端Web API就这么轻松搞定了！

**客户端通过HttpClient调用服务端Web API**

1. 新建一个WebApiTest的类库项目。

2. 在NuGet中添加System.Net.Http(HttpClient就在这里), Json.NET, xUnit.net。

3. 添加类文件WebApiClientTest.cs，添加测试方法WebApi\_SiteList\_Test：

[复制代码](javascript:void(0);)

namespace WebApiClientTest

{

public class WebApiClientTest

{

[Fact]

public void WebApi\_SiteList\_Test()

{

}

}

}

[复制代码](javascript:void(0);)

4. WebApi\_SiteList\_Test() 的代码实现

4.1 首先，要确定三个东西：

　　a) 客户端调用WebAPI的方式是Http Get，还Http Post，我们这里选用Http Post；

　　b) 客户端调用WebAPI时传递的参数格式，我们这里选用的是Json。

　　c) WebAPI返回的数据格式，我们这里选用的也是Json（这也是之前添加Json.NET引用的原因）。

4.2 用到的类

* System.Net.Http.HttpClient
* System.Net.Http.httpContent
* System.Net.Http.StringContent
* System.Net.Http.Headers.MediaTypeHeaderValue
* Newtonsoft.Json.JsonConvert

4.3 准备需要传递给WebAPI的参数

　　需要传递的两个参数是startId ,itemcount，传递的格式是Json。这里可没有Javascript中的JSON.stringify()，但我们有Json.NET，再加上匿名类型，有点用js的感觉，代码如下：

var requestJson = JsonConvert.SerializeObject(new { startId = 1, itemcount = 3 });

　　代码的运行结果：{"startId":1,"itemcount":3}

　　然后用System.Net.Http.StringContent把它打个包：

HttpContent httpContent = new StringContent(requestJson);

　　然后设置一下ContentType：

httpContent.Headers.ContentType = new MediaTypeHeaderValue("application/json");

4.4 通过Http Post调用WebAPI得到返回结果

　　HttpClient闪亮登场，调用它的PostAsync()方法轻松搞定：

var httpClient = new HttpClient();

var responseJson = httpClient.PostAsync("http://localhost:9000/api/demo/sitelist", httpContent)

.Result.Content.ReadAsStringAsync().Result;

　　看一下responseJson的结果：

[{"SiteId":2,"Title":"博客园首页","Uri":"www.cnblogs.com"},{"SiteId":3,"Title":"博问","Uri":"q.cnblogs.com"},{"SiteId":4,"Title":"新闻","Uri":"news.cnblogs.com"}]

　　正宗的Json！你注意到没有，服务端WebAPI的代码未作任何修改，我们只是在Http Headers中将ContentType设置为了application/json，返回的就是Json格式的数据。而我们通过浏览器访问，得到的还是标准的XML。这里就是ASP.NET Web API的魅力之一 —— 一次实现，按需服务。

4.5 将Json格式返回的结果反序列化为强类型

　　Json.NET又登场：

var sites = JsonConvert.DeserializeObject<IList<Site>>(responseJson);

　　展示一下返回结果：

　　代码

sites.ToList().ForEach(x => Console.WriteLine(x.Title + "：" + x.Uri));

　　结果

博客园首页：www.cnblogs.com

博问：q.cnblogs.com

新闻：news.cnblogs.com

4.6 WebApi\_SiteList\_Test() 完整实现代码

[复制代码](javascript:void(0);)

public class WebApiClientTest

{

[Fact]

public void WebApi\_SiteList\_Test()

{

var requestJson = JsonConvert.SerializeObject(new { startId = 1, itemcount = 3 });

HttpContent httpContent = new StringContent(requestJson);

httpContent.Headers.ContentType = new MediaTypeHeaderValue("application/json");

var httpClient = new HttpClient();

var responseJson = httpClient.PostAsync("http://localhost:9000/api/demo/sitelist", httpContent)

.Result.Content.ReadAsStringAsync().Result;

var sites = JsonConvert.DeserializeObject<IList<Site>>(responseJson);

sites.ToList().ForEach(x => Console.WriteLine(x.Title + "：" + x.Uri));

}

}

[复制代码](javascript:void(0);)

注：运行这里的代码之前，要先运行WebAPI项目，先把服务跑起来，客户端才能享受到服务。

与jQuery ajax调用代码比较一下：

[复制代码](javascript:void(0);)

var requestJson = JSON.stringify({ startId: 1, itemcount: 3 });

$.ajax({

url: '/api/demo/sitelist',

data: requestJson,

type: "post",

dataType: "json",

contentType: "application/json; charset=utf8",

success: function (data) {

jQuery.each(data, function (i, val) {

$("#result").append(val.Title + '： ' + val.Uri +'<br/>');

});

}

});

[复制代码](javascript:void(0);)

注：上面的代码是可真实运行的哦，代码在示例代码WebApiDemo项目的AjaxWebApi.htm文件中。这也是ASP.NET Web API “一次实现，按需服务”的体现。

**小结**

水果刀(ASP.NET Web API)用下来感觉还不错，不仅可以削苹果，还可以削梨子，切西瓜也不在话下。用不用牛刀(WCF)，还得多考虑考虑。

**示例代码下载**

<http://files.cnblogs.com/dudu/CNBlogsWebApiDemo.rar>