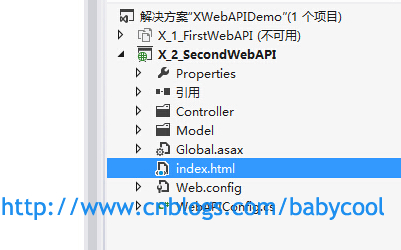
**[ASP.NET Web API 学习【转】](https://www.cnblogs.com/allenhua/p/5366272.html)**

 来看看对于一般前台页面发起的get和post请求，我们在Web API中要如何来处理。

这里我使用Jquery 来发起异步请求实现数据调用。

　　继续使用上一文章中的示例，添加一个index.html页面，添加对jquery的引用。

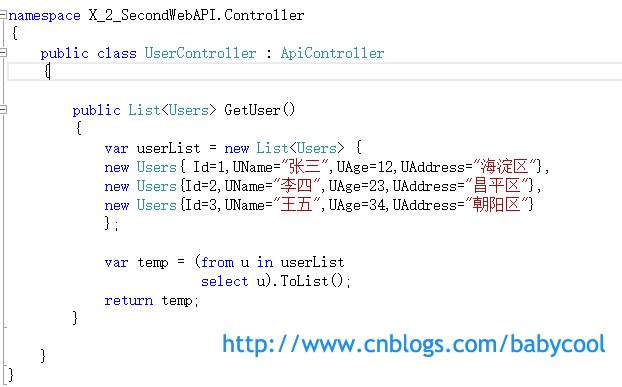


**一、无参数Get请求**

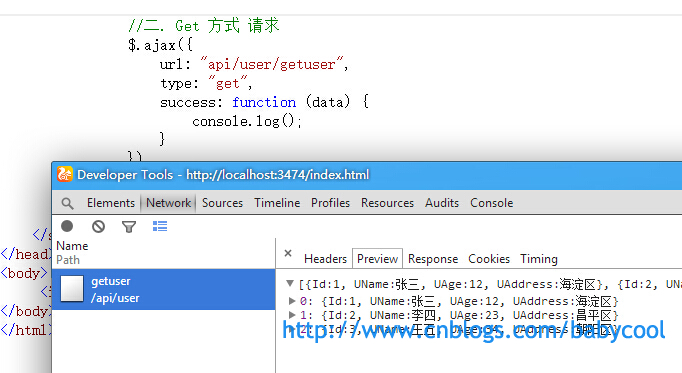
　　一般的get请求我们可以使用jquery提供的$.get() 或者$.ajax({type:"get"}) 来实现：



　　请求的后台Action方法仍为上篇文章中的GetUser() :

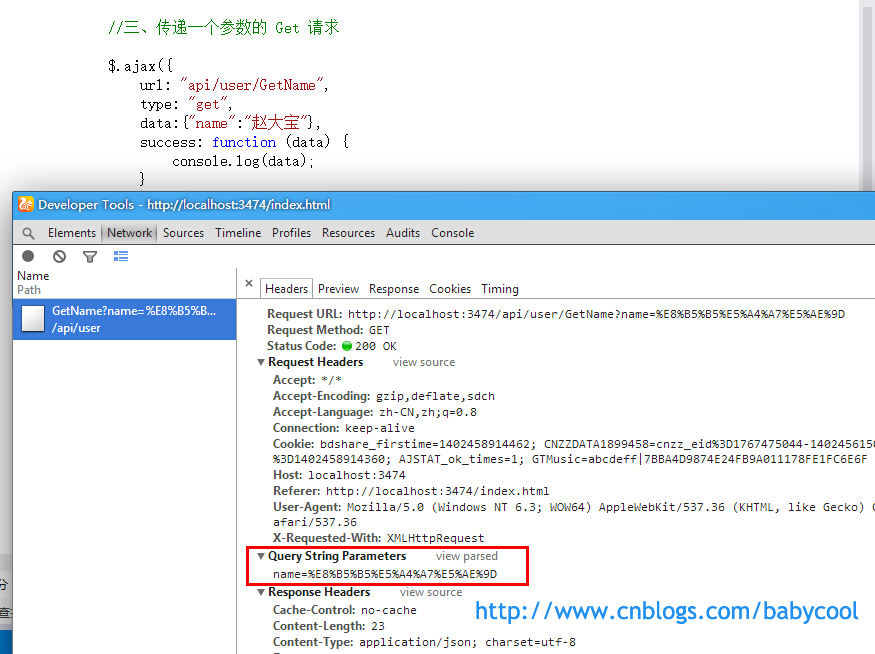


　　也可以用$.ajax({type:"get"}) 方式，正确的获得了返回数据：



**二、传递一个参数的Get请求**

　　通常我们需要传递参数只需要指定ajax方法的data属性即可： data:{"name":"赵大宝"}



　　后台正确的返回数据：



**三、传递两个或多个参数的Get请求**

　　按照上面的方法，对于多个参数我们可以很容易就写出来： data:{"name":"赵大宝","age":12}

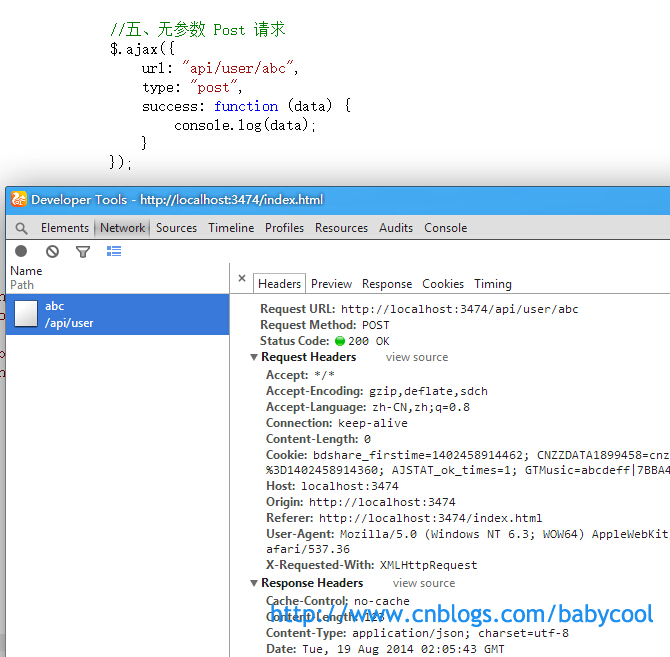


 　　后台正确的返回数据：



**四、无参数的Post请求**

　　我们可以使用$.post() 或$.ajax({type:"post"}) 来发起post请求：



　　后台正确的返回了数据：

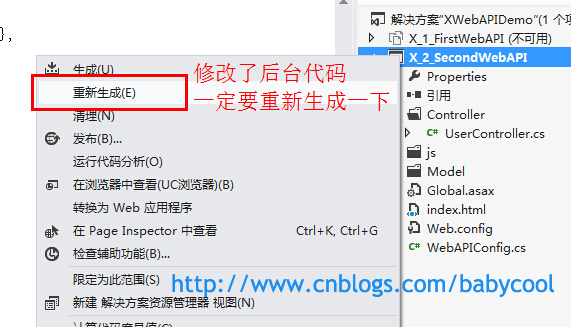


**五、传递一个参数的Post请求：**

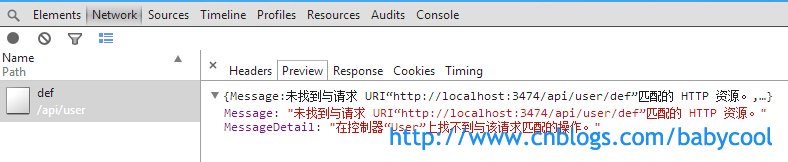
　　首先这里需要提醒大家一下，我们在修改完后台代码后，如果没有重新生成项目，那么在请求时就会报错：“未找到与请求\*\*\*匹配的HTTP资源” ：



　　所以，我们只要我们修改了后台代码，就一定要重新生成一下：



　　不过，当我们重新生成项目，再次发送请求，看到的仍然是404错误，再次检查一番代码，也没有发现是哪里的问题。

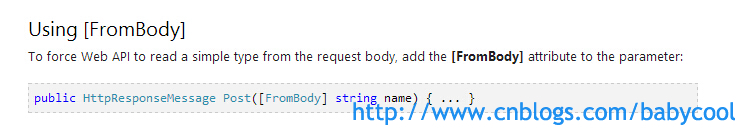


**事实上，ASP.NET Web API能够正确的识别我们的UserController控制器处理Post /api/user ，但却不能找到一个可以接受的方法来处理请求。**

也就是说，Web API接收到的请求能够找到User控制器，但在该控制器中找不到名称为Def 的这个Action。

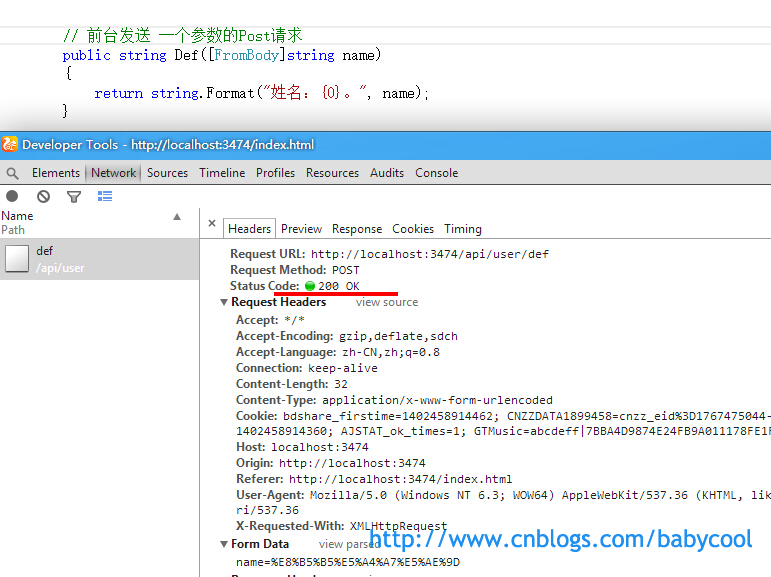
那我们要怎么来解决呢？

通过搜索MSDN上Web API官网中的说明，我们可以找到下面的一段介绍：



即**在Action 方法中我们需要使用 [FromBody] 属性标签来标明属性**。

　　修改后，再次发送请求，我们可以看到，Status Code 为200，请求发送成功。



　　可以看到，在post请求中，方法的参数必须要用 [FromBody] 属性来修饰才可以， [FromBody] 就告诉Web API 要从post请求体重去获取参数的值。

但让我们诧异的却是，后台返回的数据中name的值为空。

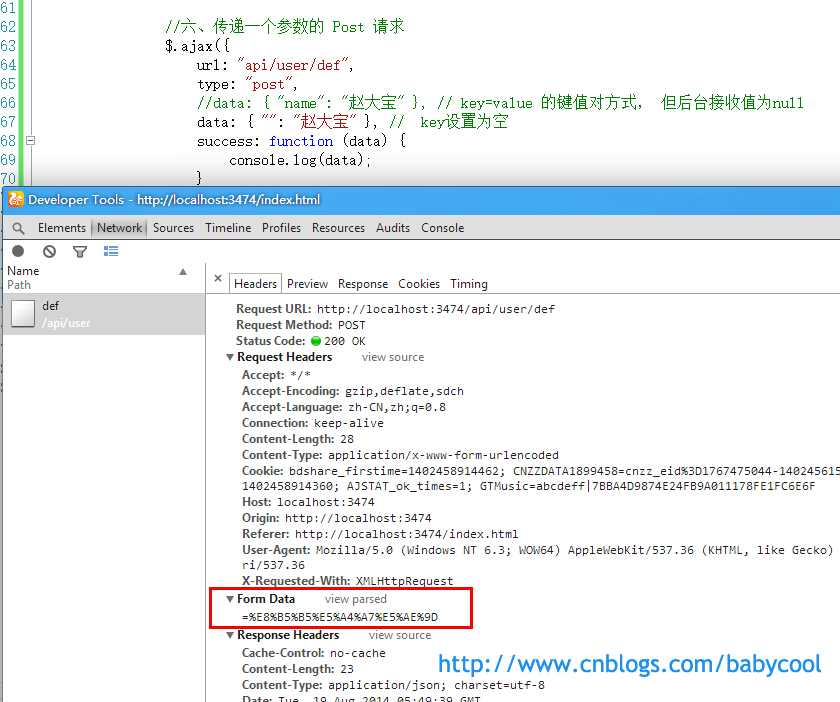


　　通过调试，我们可以看到，后台Action 中接收到的name值为null。

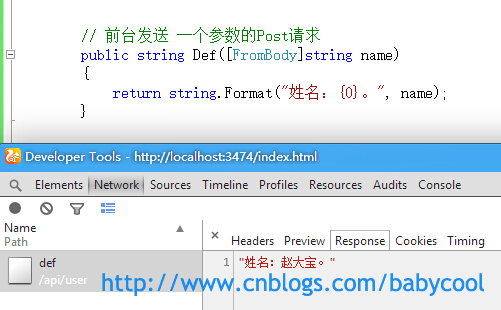


　　通过上面的测试我就也能够猜测到，Web API 要求请求传递的 [FromBody] 参数，肯定是有一个特定的格式，才能被正确的获取到。而这种特定的格式并不是我们常见的 key=value 的键值对形式。Web API 的模型绑定器希望找到 [FromBody] 里没有键名的值，也就是说， 不是 key=value ，而是 =value 。

现在，咱们把data中的key设置为空，然后再次发送请求：



　　测试可见，后台正确的接收到了数据:

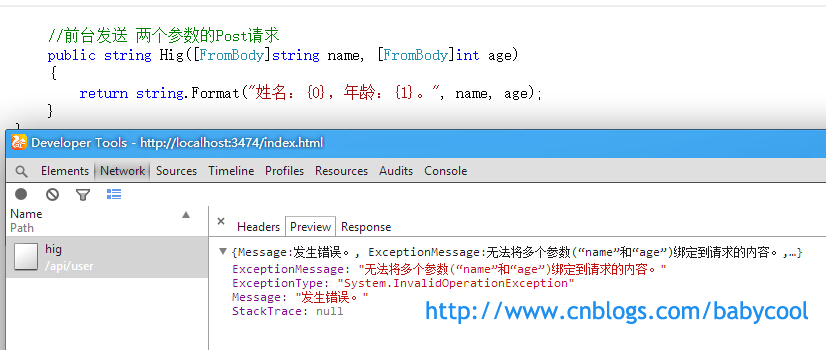


**六、传递两个参数的Post请求**

　　按理说，一个参数的请求实现了，那么传递两个或者多个参数也就很顺利了，对于两个参数的后台接收方法，我们可能会这样来写:

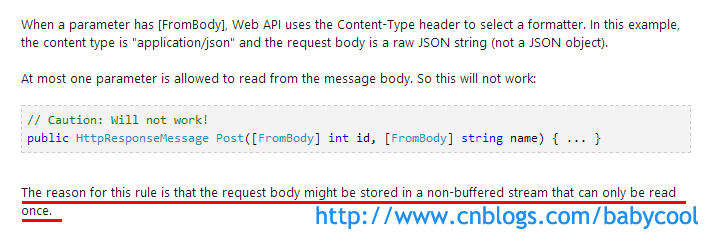


但事实证明，这样是错误的。



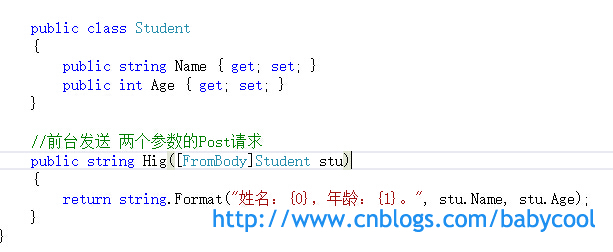
那到底两个或者多个参数我们要怎样来定义呢？

再次查看官网中的介绍，我们了解到：



　　也就是说，**[FromBody] 修饰的参数只能有一个**。我们需要对传递的多个参数进行封装才行。

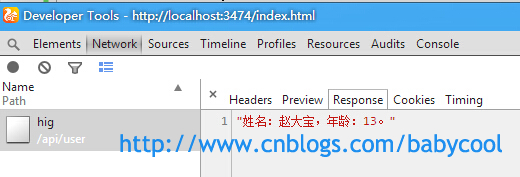
这里，我们可以将 Name和 Age 封装成一个Student 类：



　　前台页面再次发送请求：



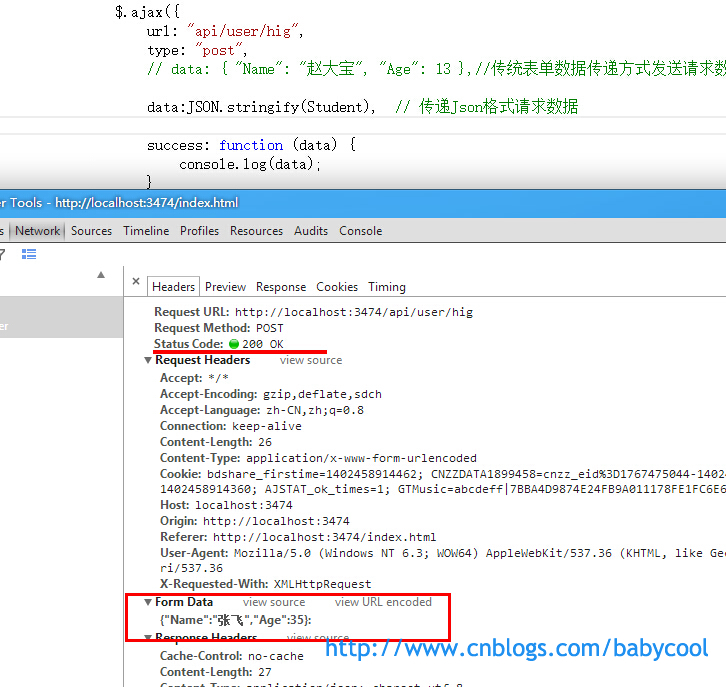
　　Status Code为200，请求成功，并正确获取到后台返回的数据：



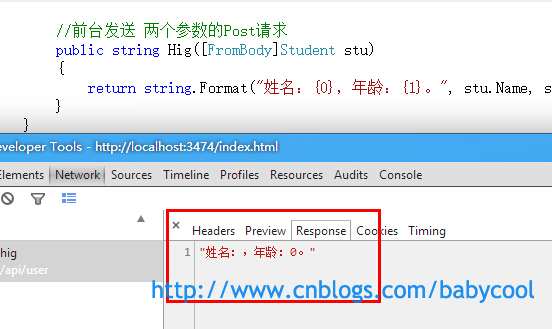
　　这里，我们通过上面图片中的请求头数据可以看到，Form Data 的格式是 key=value&key=value 这种表单数据格式 Name=%E8%B5%B5%E5%A4%A7%E5%AE%9D&Age=13 ，我们平时使用的比较多的一般为json格式。 这里，我们使用 JSON.stringify() 来序列化数据。



再次发送请求：



　　可以看到，From Data 中数据的格式为json字符串，Status Code为200，请求正确，但结果又出错了，后台又没有接收到数据参数：



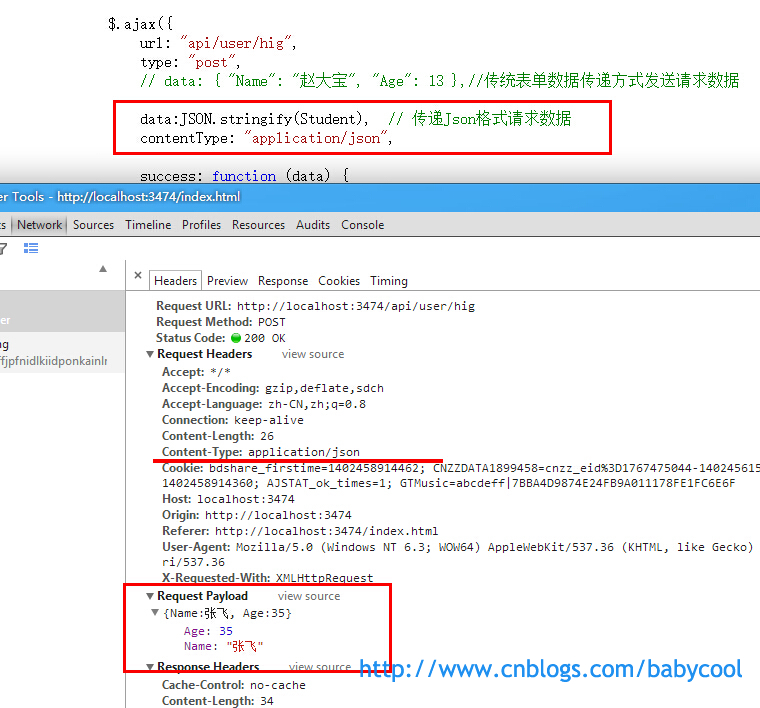
那么问题到底出在哪呢？

　　我们再次查看上图中请求头内容，注意到，虽然我们传递的数据格式是json字符串，但请求头中的Content-type 却为

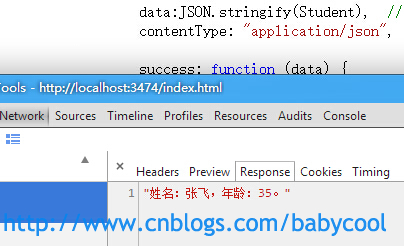
application/x-www-form-urlencoded ，并不是json格式对于的 application/json 。 而编码格式 application/x-www-form-urlencoded 表示的是：窗体数据被编码为名称/值对。

　　这里，也就是我想要说的一个注意点。**平时我们在使用json数据的时候，很容易就忘记了 要指定 Content-type 为 "application/json" ，所以就容易导致很多“想不到”的错误。**

所以，我们在 $.ajax() 中指定 Content-type :

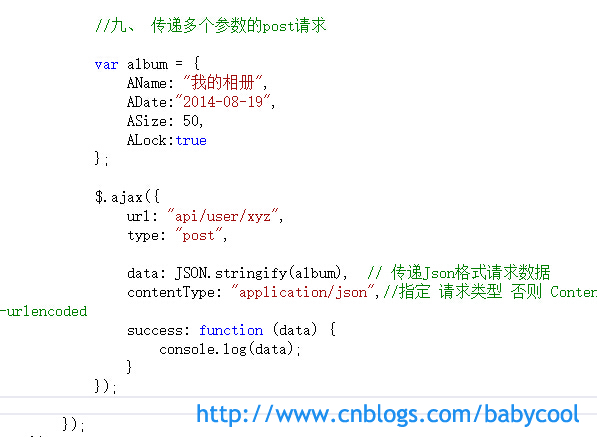


　　这次，后台正确的接收并返回了数据：



**七、传递多个参数的Post请求**

　　有了上面的经验，传递多个参数的post请求我们就很容易能写出来了：



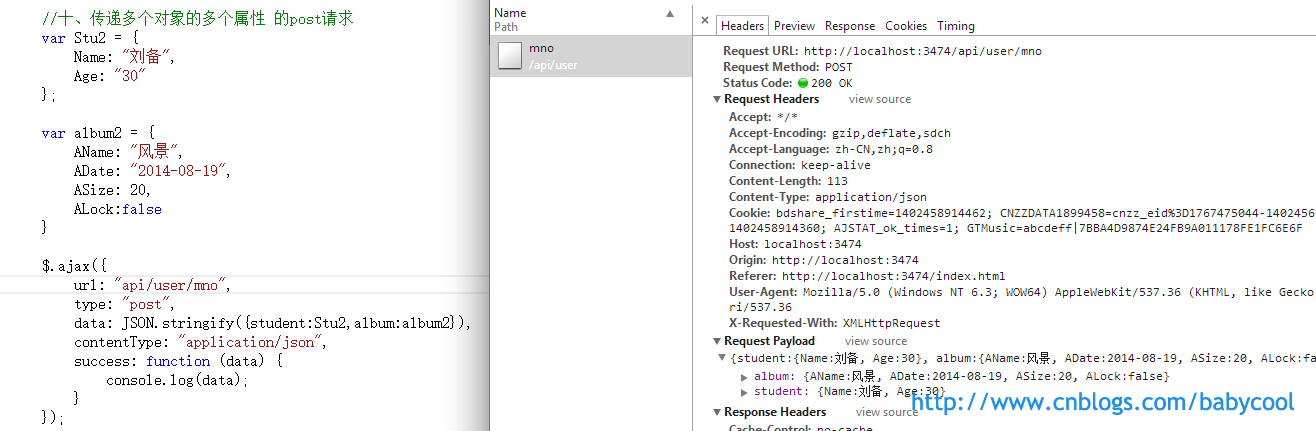
后台接收并返回数据：



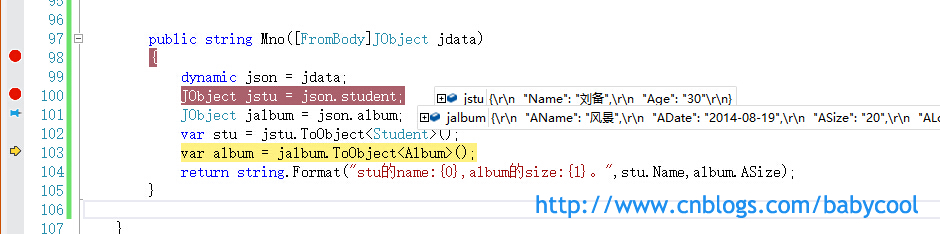
**八、传递多个不同对象的Post请求**

　　有时候我们还会遇到需要传递多个不同对象参数的情况，对于这种特殊的情况在 Json.Net 中为我们提供了一个 名为 JObject 的通用对象容器，我们可以通过 .对象名 的方式来动态的遍历参数里面的属性的值，然后动态的转换和json类型中对应的属性类型。

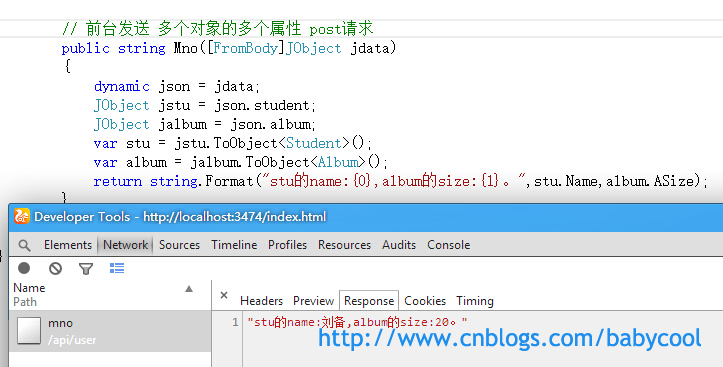
比如：



后台调试，通过dynamic 动态转换获取属性的值：



后台正确的返回了数据：



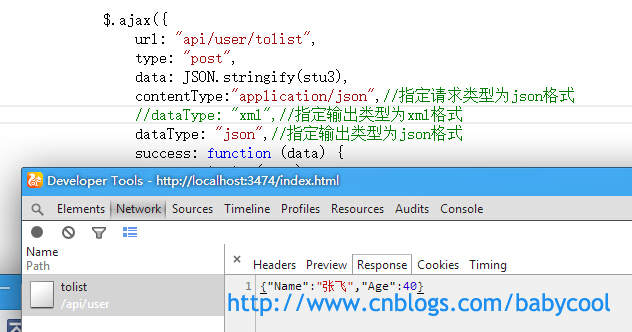
**九、获取不同类型的数据**

　　一般我们获取到后台返回的数据格式类型为 json格式，我们也可以指定 请求头中的输出类型来获得不同的返回类型数据：

指定输出类型为xml格式：



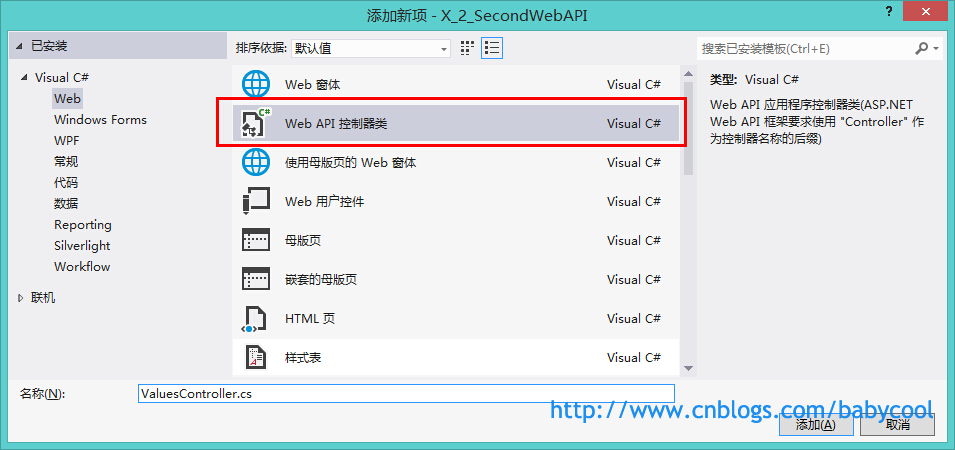
指定输出类型为 json格式：

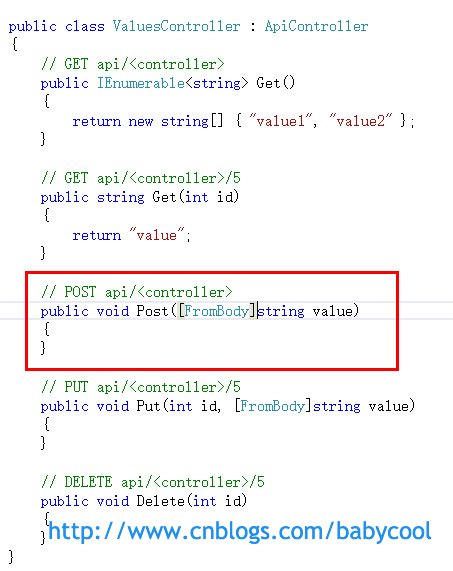


　　基本到这里，这篇文章的主要内容也就算说完了，其中的重点还是 Post请求中对于参数的处理是需要重点注意的。

　　上面我们在测试的过程中，都是通过自己创建的Controller控制器类来对参数进行接收处理，可能有人会问，我们这么写到底是否符合规范呢，Web API中默认是怎么来处理的呢？ 这里，Visual Studio 中也为我们自带了 Web API控制器类：

我们可以添加新建项选择 Web API控制器类 即可：





　　这里我们可以看到，默认创建的Controller控制器类中，对于Post请求的Action方法都自带了 [FromBody] 属性。 现在不用我说，你也已经知道为什么 它会为参数默认带一个 [FromBody]属性吧！