**[C#解析json文件的方法](http://www.cnblogs.com/txw1958/archive/2012/08/01/csharp-json.html)**

**C#** **解析 json**

　　JSON(全称为JavaScript Object Notation) 是一种轻量级的数据交换格式。它是基于JavaScript语法标准的一个子集。 JSON采用完全独立于语言的文本格式，可以很容易在各种网络、平台和程序之间传输。JSON的语法很简单，易于人阅读和编写，同时也易于机器解析和生成。

**JSON与XML的比较**◆可读性  
　　JSON和XML的可读性相比较而言，由于XML提供辅助的标签，更加适合人阅读和理解。  
◆文件大小与传输  
　　XML允许使用方便的标签，所以文件尺寸是要比JSON大的。而且JSON源于Javascript，所以天生的主战场是Javascript与网络，在这里，JSON有着XML无法赶超的优势。

**JSON语法**1. JSON 语法是 JavaScript 对象表示法语法的子集。

* 数据在名称/值对中：名称是字符串，使用双引号表示。值可以是：数字（整数或浮点数），字符串（在双引号中）,数组（在方括号中）,对象（在花括号中），true/false/null。
* 数据由逗号分隔：
* 花括号保存对象：对象可以包含各种数据，包括数组。
* 方括号保存数组：数字可以包含对象。

例如：

[复制代码](javascript:void(0);)

{  
    "employees": [  
        {  
            "firstName": "Bill",  
            "lastName": "Gates"  
        },  
        {  
            "firstName": "George",  
            "lastName": "Bush"  
        }  
    ]  
}

[复制代码](javascript:void(0);)

2. 如果JSON中含有转义字符，则需要转义。例如文件路径中需要使用"\\"而不是"\"。例如：{ "file":"C:\\a.txt"}。

**.NET操作JSON**  
　　JSON文件读入到内存中就是字符串，.NET操作JSON就是生成与解析JSON字符串。操作JSON通常有以下几种方式：  
**1. 原始方式**：自己按照JSON的语法格式，写代码直接操作JSON字符串。如非必要，应该很少人会走这条路，从头再来的。

**2. 通用方式【★★★★★】**：

这种方式是使用开源的类库Newtonsoft.Json(下载地址<http://json.codeplex.com/>)。下载后加入工程就能用。通常可以使用JObject, JsonReader, JsonWriter处理。这种方式最通用，也最灵活，可以随时修改不爽的地方。  
(1)使用JsonReader读Json字符串：

[复制代码](javascript:void(0);)

string jsonText = @"{""input"" : ""value"", ""output"" : ""result""}";  
JsonReader reader = new JsonTextReader(new StringReader(jsonText));  
  
while (reader.Read())  
{  
    Console.WriteLine(reader.TokenType + "\t\t" + reader.ValueType + "\t\t" + reader.Value);  
}

[复制代码](javascript:void(0);)

(2)使用JsonWriter写字符串：

[复制代码](javascript:void(0);)

StringWriter sw = new StringWriter();  
JsonWriter writer = new JsonTextWriter(sw);  
  
writer.WriteStartObject();  
writer.WritePropertyName("input");  
writer.WriteValue("value");  
writer.WritePropertyName("output");  
writer.WriteValue("result");  
writer.WriteEndObject();  
writer.Flush();  
  
string jsonText = sw.GetStringBuilder().ToString();  
Console.WriteLine(jsonText);

[复制代码](javascript:void(0);)

 (3)使用JObject读写字符串：

JObject jo = JObject.Parse(jsonText);  
string[] values = jo.Properties().Select(item => item.Value.ToString()).ToArray();

 (4)使用JsonSerializer读写对象(基于JsonWriter与JsonReader):

数组型数据

[复制代码](javascript:void(0);)

string jsonArrayText1 = "[{'a':'a1','b':'b1'},{'a':'a2','b':'b2'}]";

JArray ja = (JArray)JsonConvert.DeserializeObject(jsonArrayText1);

string ja1a = ja[1]["a"].ToString();

//或者

JObject o = (JObject)ja[1];

string oa = o["a"].ToString();

[复制代码](javascript:void(0);)

嵌套格式

string jsonText = "{\"beijing\":{\"zone\":\"海淀\",\"zone\_en\":\"haidian\"}}";

JObject jo = (JObject)JsonConvert.DeserializeObject(jsonText);

string zone = jo["beijing"]["zone"].ToString();

string zone\_en = jo["beijing"]["zone\_en"].ToString();

自定义类Project

[复制代码](javascript:void(0);)

Project p = new Project() { Input = "stone", Output = "gold" };  
JsonSerializer serializer = new JsonSerializer();  
StringWriter sw = new StringWriter();  
serializer.Serialize(new JsonTextWriter(sw), p);  
Console.WriteLine(sw.GetStringBuilder().ToString());  
  
StringReader sr = new StringReader(@"{""Input"":""stone"", ""Output"":""gold""}");  
Project p1 = (Project)serializer.Deserialize(new JsonTextReader(sr), typeof(Project));  
Console.WriteLine(p1.Input + "=>" + p1.Output);

[复制代码](javascript:void(0);)

　　上面的代码都是基于下面这个Project类定义：

class Project  
{  
    public string Input { get; set; }  
    public string Output { get; set; }  
}

　　此外，如果上面的JsonTextReader等类编译不过的话，说明是我们自己修改过的类，换成你们自己的相关类就可以了，不影响使用。

**3. 内置方式**：使用.NET Framework 3.5/4.0中提供的System.Web.Script.Serialization命名空间下的JavaScriptSerializer类进行对象的序列化与反序列化，很直接。

[复制代码](javascript:void(0);)

 Project p = new Project() { Input = "stone", Output = "gold" };  
 JavaScriptSerializer serializer = new JavaScriptSerializer();  
 var json = serializer.Serialize(p);  
 Console.WriteLine(json);  
  
 var p1 = serializer.Deserialize<Project>(json);  
 Console.WriteLine(p1.Input + "=>" + p1.Output);  
 Console.WriteLine(ReferenceEquals(p,p1));

[复制代码](javascript:void(0);)

**注意：**如果使用的是VS2010，则要求当前的工程的Target Framework要改成.Net Framework 4，不能使用Client Profile。当然这个System.Web.Extensions.dll主要是Web使用的，直接在Console工程中用感觉有点浪费资源。  
　　此外，从最后一句也可以看到，**序列化与反序列化是深拷贝的一种典型的实现方式**。

**4. 契约方式**：使用System.Runtime.Serialization.dll提供的DataContractJsonSerializer或者 JsonReaderWriterFactory实现。

[复制代码](javascript:void(0);)

Project p = new Project() { Input = "stone", Output = "gold" };  
DataContractJsonSerializer serializer = new DataContractJsonSerializer(p.GetType());  
string jsonText;  
  
using (MemoryStream stream = new MemoryStream())  
{  
    serializer.WriteObject(stream, p);  
    jsonText = Encoding.UTF8.GetString(stream.ToArray());  
    Console.WriteLine(jsonText);  
}  
  
using (MemoryStream ms = new MemoryStream(Encoding.UTF8.GetBytes(jsonText)))  
{  
    DataContractJsonSerializer serializer1 = new DataContractJsonSerializer(typeof(Project));  
    Project p1 = (Project)serializer1.ReadObject(ms);  
    Console.WriteLine(p1.Input + "=>" + p1.Output);  
}

[复制代码](javascript:void(0);)

　　这里要注意，这里的Project类和成员要加相关的Attribute：

[复制代码](javascript:void(0);)

[DataContract]  
class Project  
{  
    [DataMember]  
    public string Input { get; set; }  
    [DataMember]  
    public string Output { get; set; }  
}

[复制代码](javascript:void(0);)