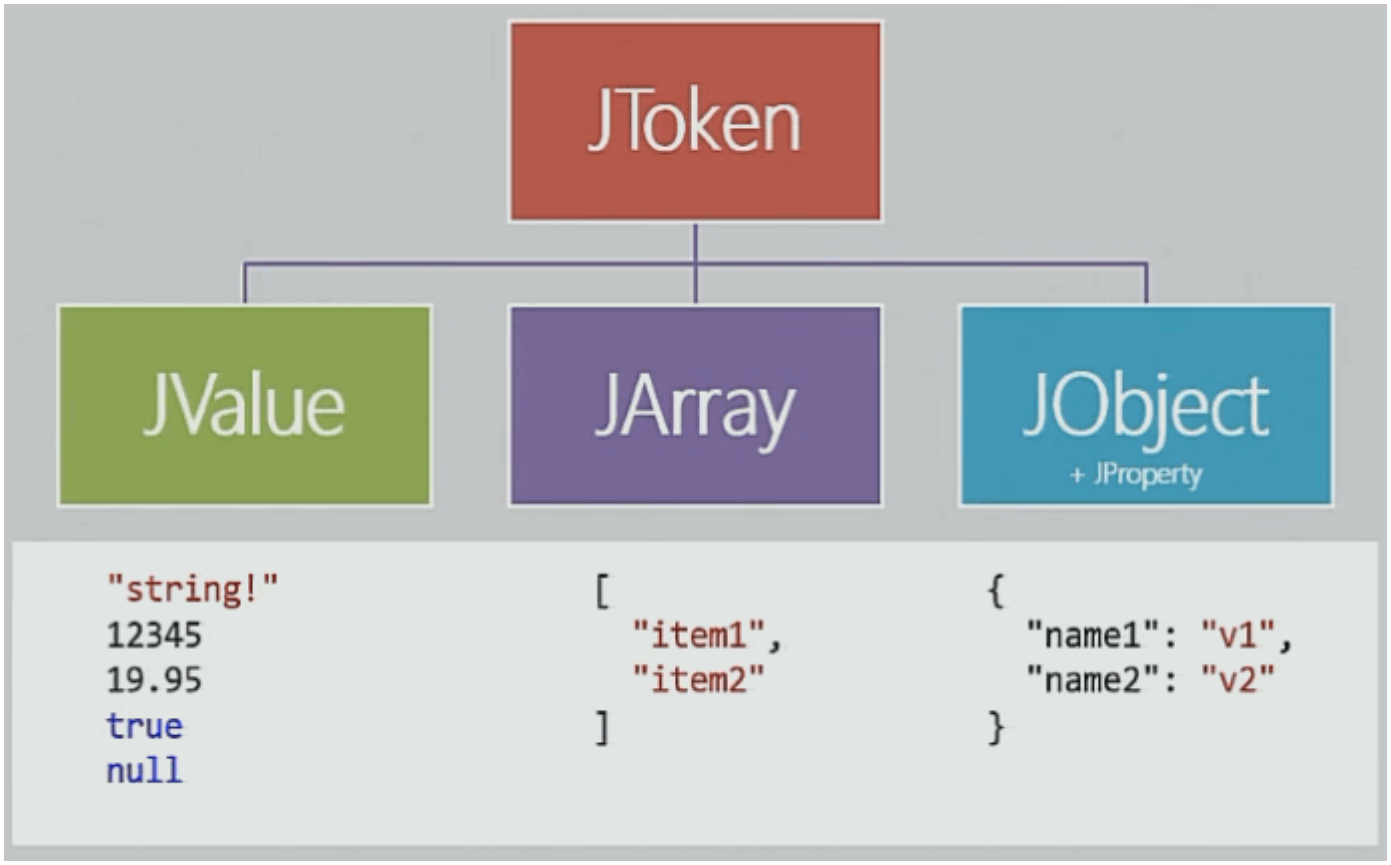
عنوان: **LINQ to JSON به کمک JSON.NET**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۹ ۲۳:۵۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, JSON, ASP.NET Web API, json.net

عموما از امکانات LINQ to JSON کتابخانه‌ی JSON.NET زمانی استفاده می‌شود که ورودی JSON تو در توی حجیمی را دریافت کرده‌اید اما قصد ندارید به ازای تمام موجودیت‌های آن یک کلاس معادل را جهت نگاشت به آن‌ها تهیه کنید و صرفا یک یا چند مقدار تو در توی آن جهت عملیات استخراج نهایی مدنظر است. به علاوه در اینجا LINQ to JSON واژه‌ی کلیدی dynamic را نیز پشتیبانی می‌کند.




همانطور که در تصویر مشخص است، خروجی‌های JSON عموما ترکیبی هستند از مقادیر، آرایه‌ها و اشیاء. هر کدام از این‌ها در LINQ to JSON به اشیاء JValue، JArray و JObject نگاشت می‌شوند. البته در حالت JObject هر عضو به یک JProperty و JValue تجزیه خواهد شد.
برای مثال آرایه [1,2] تشکیل شده‌است از یک JArray به همراه دو JValue که مقادیر آن‌را تشکیل می‌دهند. اگر مستقیما بخواهیم یک JArray را تشکیل دهیم می‌توان از شیء JArray استفاده کرد:

```
var array = new JArray(1, 2, 3);
var arrayToJson = array.ToString();
```

و اگر یک JSON رشته‌ای دریافتی را داریم می‌توان از متد Parse مربوط به JArray کمک گرفت:

```
var json = "[1,2,3]";
var jArray= JArray.Parse(json);
var val = (int)jArray[0];
```

خروجی JArray یک لیست از JTokenها است و با آن می‌توان مانند لیست‌های معمولی کار کرد.

در حالت کار با اشیاء، شیء JObject امکان تهیه اشیاء JSON ایی را دارا است که می‌تواند مجموعه‌ای از JPropertyها باشد:

```
var jObject = new JObject(
new JProperty("prop1", "value1"),
new JProperty("prop2", "value2")
);
var jObjectToJson = jObject.ToString();
```

با JObject به صورت dynamic نیز می‌توان کار کرد:

```
dynamic jObj = new JObject();
jObj.Prop1 = "value1";
jObj.Prop2 = "value2";
jObj.Roles = new[] {"Admin", "User"};
```

این روش بسیار شبیه است به حالتی که با اشیاء جاوا اسکریپتی در سمت کلاینت می‌توان کار کرد.
و حالت عکس آن توسط متد JObject.Parse قابل انجام است:

```
var json = "{ 'prop1': 'value1', 'prop2': 'value2'}";
var jObj = JObject.Parse(json);
var val1 = (string)jObj["prop1"];
```

اکنون که با اجزای تشکیل دهنده‌ی LINQ to JSON آشنا شدیم، مثال ذیل را درنظر بگیرید:

```
var array = @"[
{
  'prop1': 'value1',
  'prop2': 'value2'
},
{
  'prop1': 'test1',
  'prop2': 'test2'
}
]";
var objects = JArray.Parse(array);
var obj1 = objects.FirstOrDefault(token => (string) token["prop1"] == "value1");
```

خروجی JArray یا JObject از نوع IEnumerable است و بر روی آن‌ها می‌توان کلیه متدهای LINQ را فراخوانی کرد. برای مثال در اینجا اولین شیء‌ایی که مقدار خاصیت prop1 آن مساوی value1 است، یافت می‌شود و یا می‌توان اشیاء را بر اساس مقدار خاصیتی مرتب کرده و سپس آن‌ها را بازگشت داد:

```
var values = objects.OrderBy(token => (string) token["prop1"])
.Select(token => new {Value = (string) token["prop2"]})
.ToList();
```

امکان انجام sub queries نیز در اینجا پیش بینی شده‌است:

```
var array = @"[
{
  'prop1': 'value1',
  'prop2': [1,2]
},
{
  'prop1': 'test1',
  'prop2': [1,2,3]
}
]";
var objects = JArray.Parse(array);
var objectContaining3 = objects.Where(token => token["prop2"].Any(v => (int)v == 3)).ToList();
```

در این مثال، خواص prop2 از نوع آرایه‌ای از اعداد صحیح هستند. با کوئری نوشته شده، اشیایی که خاصیت prop2 آن‌ها دارای عضو 3 است، یافت می‌شوند.