

Processing JSON in .NET

Databases

Telerik Software Academy

<http://academy.telerik.com>

Table of Contents

- ◆ The JSON data format
 - ◆ Rules and features
 - ◆ Usage
- ◆ JSON.NET Overview
 - ◆ Installation and usage
 - ◆ LINQ-to-JSON
 - ◆ JSON to XML and XML to JSON

The JSON Data Format

What is JSON?

The JSON Data Format

- ◆ **JSON (JavaScript Object Notation) is a lightweight data format**
 - ◆ **Human and machine-readable**
 - ◆ **Based on the way to create objects in JS**
 - ◆ **Platform independent – can be used with any programming language**

- ◆ The JSON data format follows the rules of object creation in JavaScript
 - ◆ Strings, numbers and Booleans are valid JSON

```
"this is string and is valid JSON"
```

- ◆ Arrays are valid JSON

```
[5, 'string', true]
```

- ◆ Objects are valid JSON

```
{  
  "firstname": "Doncho",  
  "lastname": "Minkov",  
  "occupation": "Technical trainer"  
}
```



Processing JSON in .NET

How to parse JSON in .NET?

Built-in JSON Serializers

- ◆ .NET has built-in JSON serializer:
 - ◆ The `JavaScriptSerializer` class, contained in `System.Web.Extensions` assembly
- ◆ `JavaScriptSerializer` has parsing from object to JSON string and vice versa:

```
var place = new Place(...);
var serializer = new JavaScriptSerializer();

var jsonPlace =
    serializer.Serialize(place);
var objPlace =
    serializer.Deserialize<place>(jsonPlace);
```

JavaScript Serializer

Live Demo

JavaScriptSerializer Features

- ◆ JavaScript serializer has nice features:
 - ◆ Serializing objects to JSON and vice versa
 - ◆ Correct parsing of dictionaries:

```
digits =
    new Dictionary<string, int>
{
    { "one", 1},
    { "two", 2},
    ...
};
```

JavaScriptSerializer Features

- ◆ JavaScript serializer has nice features:
 - ◆ Serializing objects to JSON and vice versa
 - ◆ Correct parsing of dictionaries:

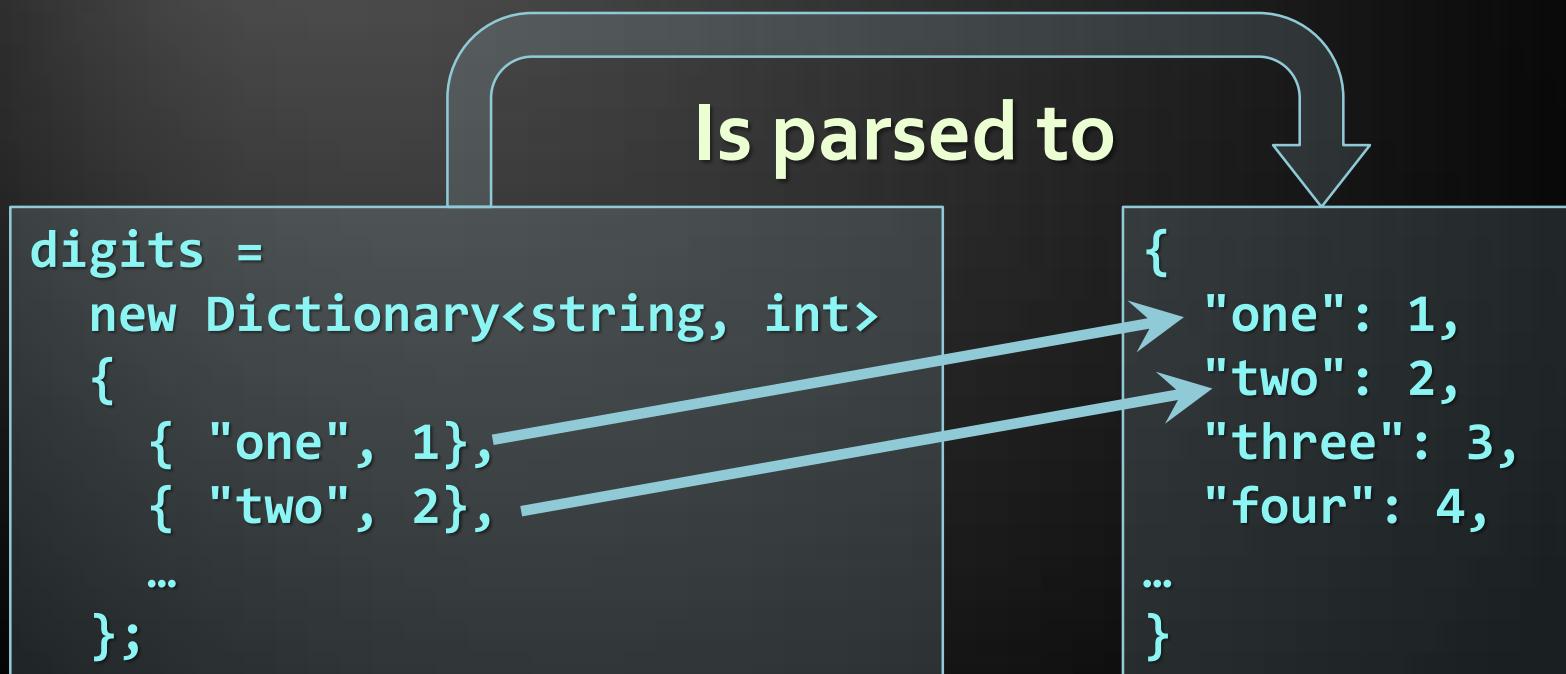
Is parsed to

```
digits =  
new Dictionary<string, int>  
{  
    { "one", 1},  
    { "two", 2},  
    ...  
};
```

```
{  
    "one": 1,  
    "two": 2,  
    "three": 3,  
    "four": 4,  
    ...  
}
```

JavaScriptSerializer Features

- ◆ JavaScript serializer has nice features:
 - ◆ Serializing objects to JSON and vice versa
 - ◆ Correct parsing of dictionaries:



JavaScript Serializer Features

Live Demo

JSON.NET

Better JSON parsing than with JavaScriptSerializer

- ◆ **JavaScriptSerializer is good**
 - ◆ But JSON.NET is better
- ◆ **JSON.NET:**
 - ◆ Has better performance
 - ◆ Supports LINQ-to-JSON
 - ◆ Has an out-of-the-box support for parsing between JSON and XML

Installing JSON.NET

- ◆ To install JSON.NET run in the Package Manager Console:

```
$ Install-Package Newtonsoft.Json
```

- ◆ Has two primary methods:
 - ◆ Serialize an object:

```
var serializedPlace =  
    JsonConvert.SerializeObject(place);
```

- ◆ Deserialize an object:

```
var deserializedPlace =  
    JsonConvert.DeserializeObject<Place>(serializedPlace);
```

Serializing and Deserializing Objects

Live Demo

- ◆ JSON.NET can be configured to:
 - ◆ Indent the output JSON string
 - ◆ To convert JSON to anonymous types
 - ◆ To control the casing and properties to parse
 - ◆ To skip errors
- ◆ JSON.NET also supports:
 - ◆ LINQ-to-JSON
 - ◆ Direct parse between XML and JSON

Configuring JSON.NET

- ◆ To indent the output string use `Formatting.Indented`:

```
JsonConvert.SerializeObject(place, Formatting.Indented);
```

- ◆ Deserializing to anonymous types:

```
var json = @"{ 'firstName': 'Doncho', "
              'lastName': 'Minkov',
              'occupation': 'Technical Trainer' }";
var template = new
{
    FirstName = "",
    Lastname = "",
    Occupation = ""
};
var person =
    JsonConvert.DeserializeAnonymousType(json, template);
```

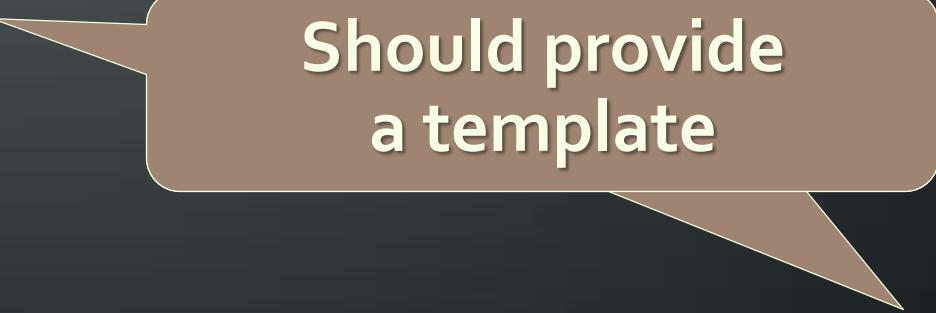
Configuring JSON.NET

- ◆ To indent the output string use `Formatting.Indented`:

```
JsonConvert.SerializeObject(place, Formatting.Indented);
```

- ◆ Deserializing to anonymous types:

```
var json = @"{ 'firstName': 'Doncho',  
                'lastName': 'Minkov',  
                'occupation': 'Technical Trainer' }";  
  
var template = new  
{  
    FirstName = "",  
    Lastname = "",  
    Occupation = ""  
};  
  
var person =  
    JsonConvert.DeserializeObjectAnonymousType(json, template);
```



Should provide
a template

Configuring JSON.NET

Live Demo

JSON.NET Parsing of Objects

- ◆ By default JSON.NET takes each Property/Field from the public interface of a class and parses it
 - ◆ This can be controlled using attributes:

```
public class User
{
    [JsonProperty("user")]
    public string Username{get;set;}
    [JsonIgnore]
    public string Password{get;set;}
}
```

JSON.NET Parsing of Objects

- ◆ By default JSON.NET takes each Property/Field from the public interface of a class and parses it
 - ◆ This can be controlled using attributes:

```
public class User
{
    [JsonProperty("user")]
    public string Username{get;set;}
    [JsonIgnore]
    public string Password{get;set;}
}
```

Tells the converter to parse "Username" to "user"

JSON.NET Parsing of Objects

- ◆ By default JSON.NET takes each Property/Field from the public interface of a class and parses it as is
 - ◆ This can be controlled using attributes:

```
public class User
{
    [JsonProperty("user")]
    public string Username{get;set;}

    [JsonIgnore]
    public string Password{get;set;}
}
```

Tells the converter to parse "Username" to "user"

Tells the converter to skip the property "Password"

JSON.NET Parsing of Objects

Live Demo

- ◆ JSON.NET has a support for LINQ-to-JSON

```
var jsonObj = JObject.Parse(json);
Console.WriteLine("Places in {0}:", jsonObj["name"]);

jsonObj["places"].Select(
    pl =>
        string.Format("{0}) {1} ({2})",
                      index++,
                      pl["name"],
                      string.Join(", ",
                                  pl["categories"].Select(
                                      cat => cat["name"])))))
    .Print();
```

LINQ-to-JSON

Live Demo

XML to JSON and JSON to XML

XML to JSON and JSON to XML

- ◆ Conversions from JSON to XML are done using two methods:
 - ◆ XML to JSON

```
string jsonFromXml = JsonConvert.SerializeXmlNode(doc);
```

- ◆ JSON to XML

```
XDocument xmlFromJson = JsonConvert.DeserializeXmlNode(json);
```

XML to JSON and JSON to XML

Live Demo

Processing JSON in .NET

Questions?

Using JSON.NET and the Telerik Academy Forums RSS feed implement the following:

1. The RSS feed is at <http://forums.academy.telerik.com/feed/qa.rss>
2. Download the content of the feed programmatically
 - You can use `WebClient.DownloadFile()`
3. Parse the XML from the feed to JSON
4. Using LINQ-to-JSON select all the question titles and print them to the console
5. Parse the JSON string to POCO
6. Using the parsed objects create a HTML page that lists all questions from the RSS their categories and a link to the question's page