External Format Processing

Parsing JSON, JSON.NET

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg/

Have a Question?

Software University

sli.do

#csharp-db

Table of Contents

JSON Data Format

Definition and Syntax

JSON Data Format

- **JSON** (**J**ava**S**cript **O**bject **N**otation) is a lightweight data format
  - Human and machine-readable plain text
  - Based on **JavaScript** objects
  - Independent of development platforms and languages
  - JSON data consists of
    - Values (**strings**, **numbers**, **etc.**)
    - Key-value pairs: **`{ key : value }`**
    - Arrays: **`[value1, value2, …]`**

```
{
    "firstName": "Pesho",
    "courses": ["C#", "JS", "ASP.NET"]
    "age": 23,
    "hasDriverLicense": true,
    "date": "2012-04-23T18:25:43.511Z",
    // ...
}
```

JSON Data Format

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

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

```
3.14
```

```
true
```

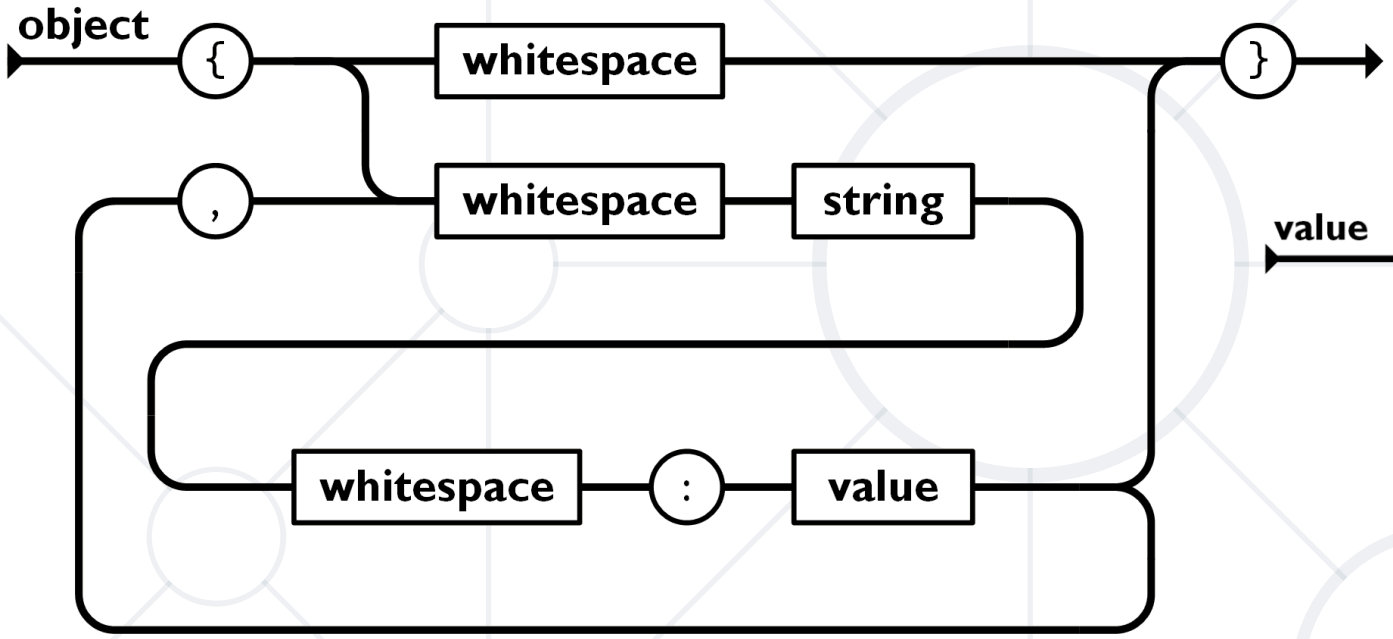
  - **Arrays** are valid JSON

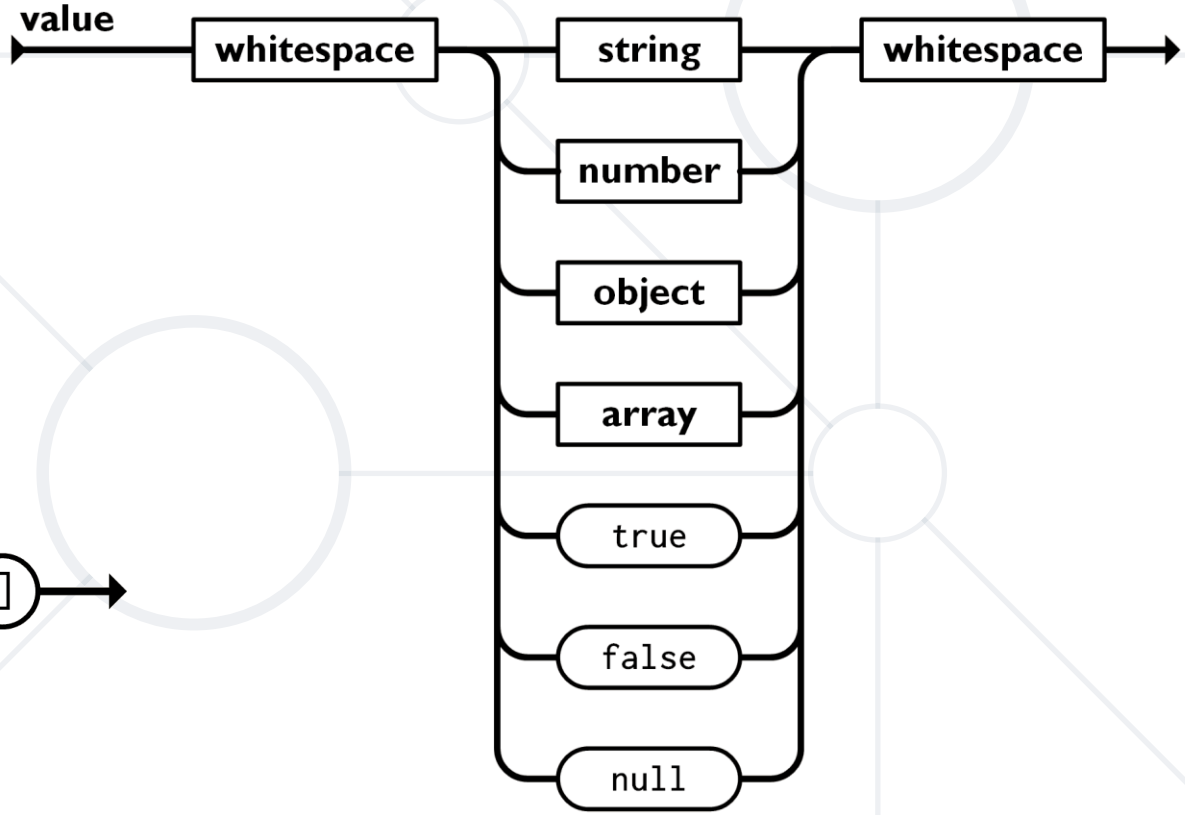
```
[5, "text", true]
```

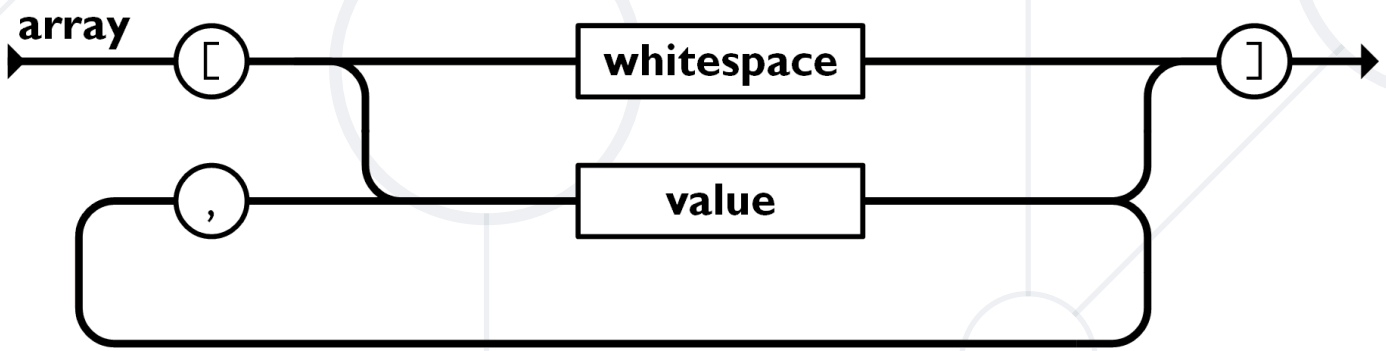
  - **Objects** are valid JSON (key-value pairs)

```
{
  "firstName": "Svetlin", "lastName": "Nakov",
  "jobTitle": "Technical Trainer", "age": 40
}
```

Object, Array and Value in JSON





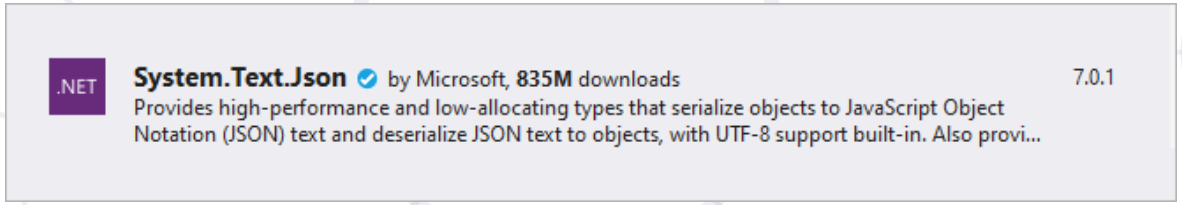


Processing JSON

Parsing JSON in C# and .NET with System.Text.Json

Built-in JSON Support

- .NET has built-in JSON support through the `System.Text.Json` NuGet Package



  - It supports **serializing** objects and **deserializing** (parsing) strings

- Include **the following namespaces** into your project

```
using System.Text.Json;
using System.Text.Json.Serialization;
```

Serializing JSON

- The `System.Text.Json` serializer can read and write JSON

```
class WeatherForecast
{
    public DateTime Date { get; set; } = DateTime.Now;
    public int TemperatureC { get; set; } = 30;
    public string Summary { get; set; } = "Hot summer day";
}

static void Main()
{
    WeatherForecast forecast = new WeatherForecast();
    string weatherInfo = JsonSerializer.Serialize(forecast);
    Console.WriteLine(weatherInfo);
}
```

Serializing JSON

- Creating a JSON file

```
static void Main()
{
    WeatherForecast forecast = new WeatherForecast();
    string weatherInfo = JsonSerializer.Serialize(forecast);
    File.WriteAllText(file, weatherInfo);
}
```

```
{"Date":"2020-07-16T13:33:25","TemperatureC":30,"Summary":"Hot summer day"}
```

Deserializing JSON

- To deserialize from a file, we read the file into a string and then use the **Deserialize** method

```
static void Main()
{
    string jsonString = File.ReadAllText(file);
    WeatherForecast forecast =
      JsonSerializer.Deserialize<WeatherForecast>(jsonString);
}
```

Name	Value	Type
forecast	{JsonDemo.Program.WeatherForecast}	JsonDemo.Program.WeatherForecast
Date	{5.2.2023 r. 20:16:32}	System.DateTime
Summary	"Hot summer day" View	string
TemperatureC	30	int

JSON.NET

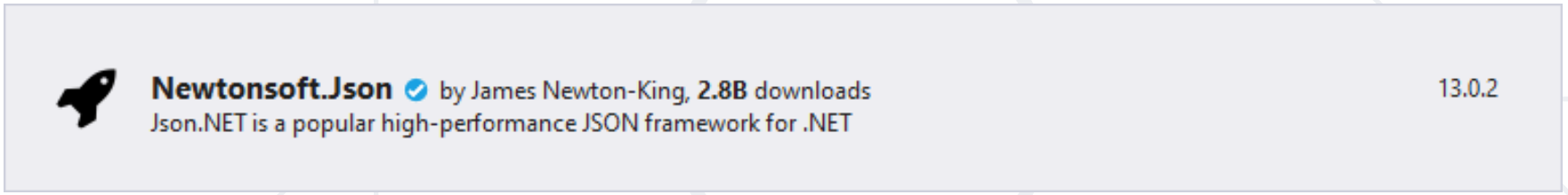
Better JSON Parsing for .NET Developers

What is JSON.NET?

- **JSON.NET** is a JSON **framework** for .NET
  - **More functionality** than built-in functionality
  - Supports **LINQ-to-JSON**
  - Out-of-the-box support for parsing between **JSON** and **XML**
  - Open-source project: **http://www.newtonsoft.com**
  - **`Newtonsoft.Json`** vs **`System.Text.Json`**
    - Performance comparison
    - Table of differences

Installing JSON.NET

- To install JSON.NET use the **NuGet Package Manager**



- Or with a command in the Package Manager Console

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

General Usage

- JSON.NET exposes a static service **JsonConvert**
- Used for parsing and configuration to
  - **Serialize** an object

```
var jsonProduct = JsonConvert.SerializeObject(product);
```

  - **Deserialize** an object

```
var objProduct =
   JsonConvert.DeserializeObject<Product>(jsonProduct);
```

JSON.NET Features

- JSON.NET can be configured to
  - **Indent** the output JSON string
  - Convert JSON to **anonymous types**
  - Control the **casing** and **properties** to parse
  - Skip errors
- JSON.NET also supports
  - LINQ-to-JSON
  - Direct parsing between XML and JSON

Configuring JSON.NET

- By default, the result is a **single line of text**
- To indent the output string use **`Formatting.Indented`**

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

```
{
  "pump": {
    "Id": 0,
    "Name": "Oil Pump",
    "Description": null,
    "Cost": 25.0
  },
  "filter": {
    "Id": 0,
    "Name": "Oil Filter",
    "Description": null,
    "Cost": 15.0
  }
}
```

Configuring JSON.NET

- Deserializing to **anonymous** types

Incoming JSON

```
var json = @"{ 'firstName': 'Svetlin',
               'lastName': 'Nakov',
               'jobTitle': 'Technical Trainer' }";
var template = new
{
    FirstName = string.Empty,
    LastName = string.Empty,
    JobTitle = string.Empty
};
var person = JsonConvert.DeserializeAnonymousType(json,
template);
```

Template objects

JSON.NET Attributes

- By default JSON.NET takes each property / field from the 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; }
}
```

Parse Username to user

Skip the property

JSON.NET Parsing of Objects

- By default JSON.NET takes each property / field from the class and parses it
  - This can be controlled using **ContractResolver**

```
DefaultContractResolver contractResolver =
    new DefaultContractResolver()
    {
        NamingStrategy = new SnakeCaseNamingStrategy()
    };
var serialized = JsonConvert.SerializeObject(person,
    new JsonSerializerSettings()
    {
        ContractResolver = contractResolver,
        Formatting = Formatting.Indented
    });
```

LINQ-to-JSON

- LINQ-to-JSON works with **JObjects**
  - Create from JSON string

```
JObject obj = JObject.Parse(jsonProduct);
```

  - Reading from file

```
var people = JObject.Parse(File.ReadAllText(@"c:\people.json"))
```

  - Using **JObject**

```
foreach (JToken person in people)
{
  Console.WriteLine(person["FirstName"]); // Ivan
  Console.WriteLine(person["LastName"]); // Petrov
}
```

LINQ-to-JSON

- `JObjects` can be queried with LINQ

```
var json = JObject.Parse(@"{'products': [
  {'name': 'Fruits', 'products': ['apple', 'banana']},
  {'name': 'Vegetables', 'products': ['cucumber']}]}");

var products = json["products"].Select(t =>
  string.Format("{0} ({1})",
    t["name"],
    string.Join(", ", c["products"])
));

// Fruits (apple, banana)
// Vegetables (cucumber)
```

XML-to-JSON

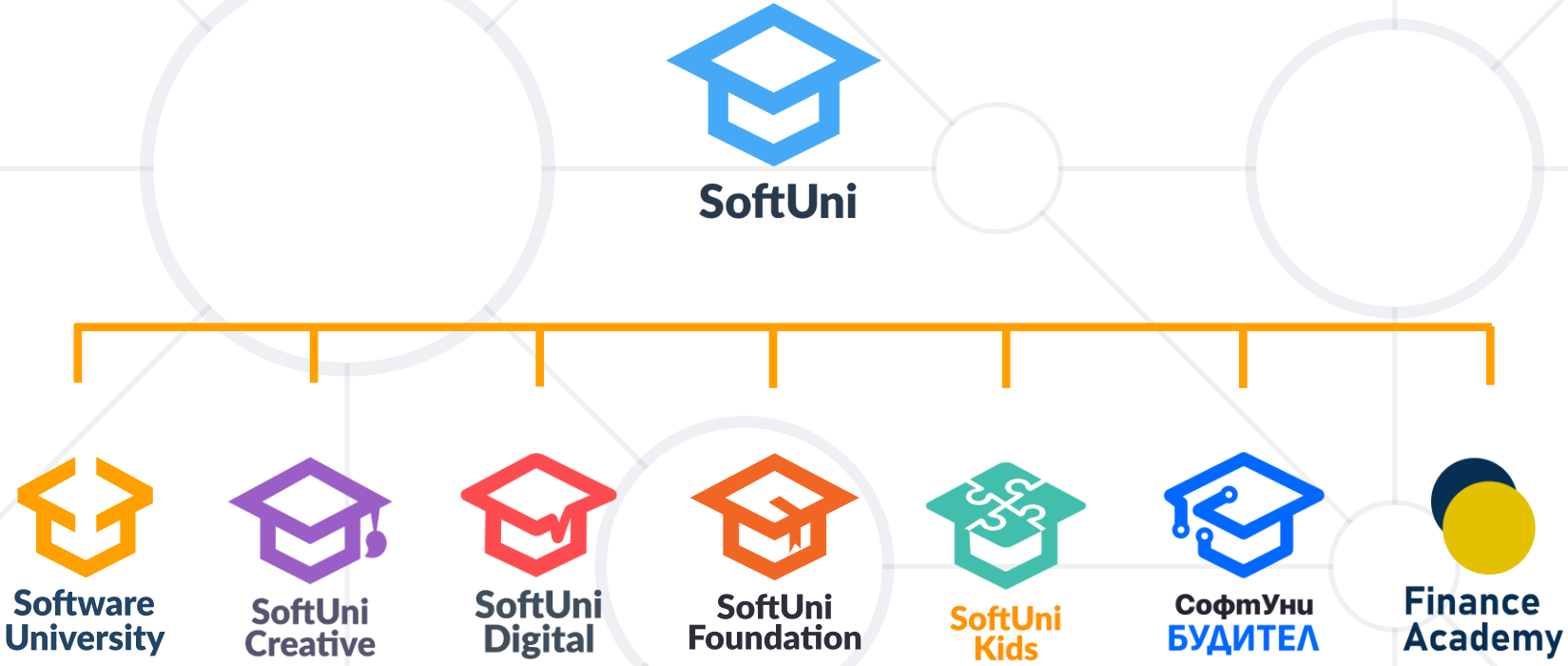
```
string xml = @"<?xml version='1.0' standalone='no'?>
 <root>
    <person id='1'>
        <name>Alan</name>
        <url>www.google.com</url>
    </person>
    <person id='2'>
        <name>Louis</name>
        <url>www.yahoo.com</url>
    </person>
</root>";

XmlDocument doc = new XmlDocument();
doc.LoadXml(xml);
string jsonText = JsonConvert.SerializeXmlNode(doc);
```

```
{
  "?xml": {
    "@version": "1.0",
    "@standalone": "no"
  },
  "root": {
    "person": [
      {
        "@id": "1",
        "name": "Alan",
        "url": "www.google.com"
      },
      {
        "@id": "2",
        "name": "Louis",
        "url": "www.yahoo.com"
      }
    ]
  }
}
```

Summary

- JSON is a cross platform text-based data format
- System.Text.Json is the JSON Parser in C#
- JSON.NET is a fast framework for working with JSON data

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

PHAR VISION

INDEAVR

Serving the high achievers

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg
- © Software University – https://softuni.bg