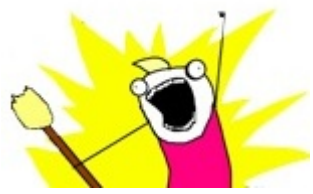

Introduction

Json.NET is a popular high-performance JSON framework for .NET

Benefits and Features

- Flexible JSON serializer for converting between .NET objects and JSON
- LINQ to JSON for manually reading and writing JSON
- High performance: faster than .NET's built-in JSON serializers
- Write indented, easy-to-read JSON
- Convert JSON to and from XML
- Supports .NET Standard 2.0, .NET 2, .NET 3.5, .NET 4, .NET 4.5, Silverlight, Windows Phone and Windows 8 Store

The JSON serializer in Json.NET is a good choice when the JSON you are reading or writing maps closely to a .NET class.

LINQ to JSON is good for situations where you are only interested in getting values from JSON, you don't have a class to serialize or deserialize to, or the JSON is radically different from your class and you need to manually read and write from your objects.

Getting Started

- Serializing and Deserializing JSON
- LINQ to JSON
- Samples

History

Json.NET grew out of projects I was working on in late 2005 involving JavaScript, AJAX, and .NET. At the time there were no libraries for working with JavaScript in .NET, so I made my own.

Starting out as a couple of static methods for escaping JavaScript strings, Json.NET evolved as features were added. To add support for reading JSON a major refactor was required, and Json.NET was split into the three major classes it still uses today: JsonReader, JsonWriter and JsonSerializer.

Json.NET was first released in June 2006. Since then Json.NET has been downloaded hundreds of thousands of times by developers from around the world. It is used in many major open source projects, including: Mono, an open source implementation of the .NET framework; RavenDB, a JSON based document database; ASP.NET SignalR, an async library for building real-time, multi-user interactive web applications; and ASP.NET Core, Microsoft's web app and service framework.

See Also

Other Resources

Serializing and Deserializing JSON
LINQ to JSON
Samples

Serialize an Object

This sample serializes an object to JSON.

Sample

Types Copy

```
public class Account
{
    public string Email { get; set; }
    public bool Active { get; set; }
    public DateTime CreatedDate { get; set; }
    public IList<string> Roles { get; set; }
}
```

Usage Copy

```
Account account = new Account
{
    Email = "james@example.com",
    Active = true,
    CreatedDate = new DateTime(2013, 1, 20, 0, 0, 0, DateTimeKind.Utc),
    Roles = new List<string>
    {
        "User",
        "Admin"
    }
};

string json = JsonConvert.SerializeObject(account, Formatting.Indented);
// {
//   "Email": "james@example.com",
//   "Active": true,
//   "CreatedDate": "2013-01-20T00:00:00Z",
//   "Roles": [
//     "User",
//     "Admin"
//   ]
// }

Console.WriteLine(json);
```

Serialize a Collection

This sample serializes a collection to JSON.

Sample

Usage

Copy

```
List<string> videogames = new List<string>
{
    "Starcraft",
    "Halo",
    "Legend of Zelda"
};

string json = JsonConvert.SerializeObject(videogames);

Console.WriteLine(json);
// ["Starcraft","Halo","Legend of Zelda"]
```

Serialize a Dictionary

This sample serializes a dictionary to JSON.

Sample

Usage Copy

```
Dictionary<string, int> points = new Dictionary<string, int>
{
    { "James", 9001 },
    { "Jo", 3474 },
    { "Jess", 11926 }
};

string json = JsonConvert.SerializeObject(points, Formatting.Indented);

Console.WriteLine(json);
// {
//   "James": 9001,
//   "Jo": 3474,
//   "Jess": 11926
// }
```

Serialize JSON to a file

This sample serializes JSON to a file.

Sample

Types

Copy

```
public class Movie
{
    public string Name { get; set; }
    public int Year { get; set; }
}
```

Usage

Copy

```
Movie movie = new Movie
{
    Name = "Bad Boys",
    Year = 1995
};

// serialize JSON to a string and then write string to a file
File.WriteAllText(@"c:\movie.json", JsonConvert.SerializeObject(movie));

// serialize JSON directly to a file
using (StreamWriter file = File.CreateText(@"c:\movie.json"))
{
    JsonSerializer serializer = new JsonSerializer();
    serializer.Serialize(file, movie);
}
```

Deserialize an Object

This sample deserializes JSON to an object.

Sample

Types

Copy

```
public class Account
{
    public string Email { get; set; }
    public bool Active { get; set; }
    public DateTime CreatedDate { get; set; }
    public IList<string> Roles { get; set; }
}
```

Usage

Copy

```
string json = @"{
  'Email': 'james@example.com',
  'Active': true,
  'CreatedDate': '2013-01-20T00:00:00Z',
  'Roles': [
    'User',
    'Admin'
  ]
}";

Account account = JsonConvert.DeserializeObject<Account>(json);

Console.WriteLine(account.Email);
// james@example.com
```

Deserialize a Collection

This sample deserializes JSON into a collection.

Sample

Usage Copy

```
string json = @"['Starcraft','Halo','Legend of Zelda']";

List<string> videogames = JsonConvert.DeserializeObject<List<string>>(json);

Console.WriteLine(string.Join(", ", videogames.ToArray()));
// Starcraft, Halo, Legend of Zelda
```