FREE eBook

LEARNING

json.net

Free unaffiliated eBook created from
Stack Overflow contributors.

#json.net

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: json-net

It is an unofficial and free json.net ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official json.net.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with json.net

Remarks

[Json.NET][1] is a popular high-performance JSON framework for .NET. It's a **.Net** Assembly that exposes a wide variety of classes and methods to help perform common tasks with json data.

This includes the ability to do things like serializing json data into class instances, data files as well as deserializing json data that might be coming from many different sources like from an API end point or from `.json` data files.

Versions

Version	Release Date
Json.NET 9.0.1 (current version)	2016-06-22
Json.NET 8.0.3	2016-03-14
Json.NET 8.0.2	2016-01-09
Json.NET 8.0.1	2015-12-29
Json.NET 7.0.1	2015-06-22
Json.NET 6.0.8	2015-01-11
Json.NET 6.0.7	2014-12-23
Json.NET 6.0.6	2014-10-24
Json.NET 6.0.5	2014-09-06
Json.NET 6.0.4	2014-08-03
Json.NET 6.0.3	2014-04-27
Json.NET 6.0.2	2014-03-30
Json.NET 6.0.1	2014-02-01
Json.NET 5.0.8	2013-10-17
Json.NET 5.0.7	2013-10-14
Json.NET 5.0.6	2013-06-06
Json.NET 5.0.5	2013-05-08

Version	Release Date
Json.NET 5.0.4	2013-04-25
Json.NET 5.0.3	2013-04-14
Json.NET 5.0.2	2013-04-08
Json.NET 5.0.1	2013-04-07
Json.NET 4.5.11	2012-11-20
Json.NET 4.5.10	2012-10-07
Json.NET 4.5.9	2012-09-08
Json.NET 4.5.8	2012-08-04
Json.NET 4.5.7	2012-06-09
Json.NET 4.5.6	2012-05-30
Json.NET 4.5.5	2012-05-08
Json.NET 4.5.4	2012-04-24
Json.NET 4.5.3	2012-04-13
Json.NET 4.5.2	2012-04-11
Json.NET 4.5.1	2012-03-20
Json.NET 4.0.8	2012-02-12
Json.NET 4.0.7	2012-01-24
Json.NET 4.0.6	2012-01-23
Json.NET 4.0.5	2011-12-10
Json.NET 4.0.4	2011-11-19
Json.NET 4.0.3	2011-10-02
Json.NET 4.0.2	2011-04-23
Json.NET 4.0.1	2011-04-22
Json.NET 3.5.8	2011-01-08

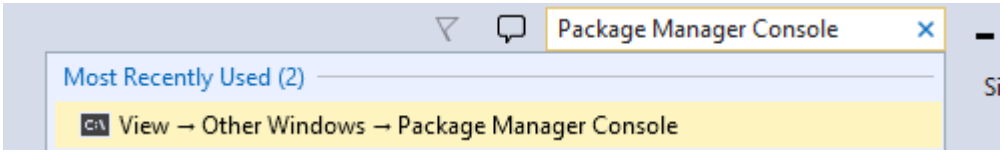
Examples

How to Install Json.Net in Visual Studio Projects

You can install Json.Net into your Visual Studio Project in 1 of 2 ways.

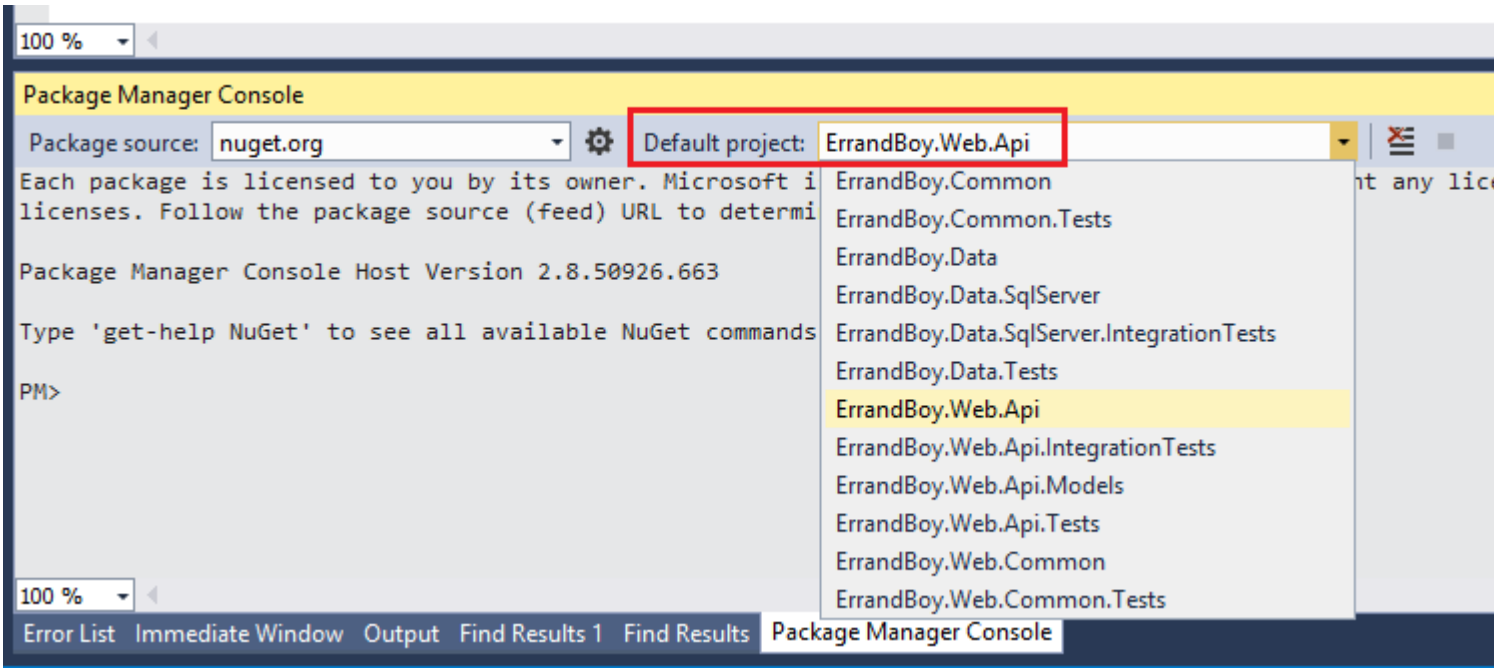
Install Json.Net using the Package Manager Console.

1. Open the **Package Manager Console** window in Visual Studio either by typing **package manager console** in the Quick Launch box and selecting it



or by clicking **View** -> **Other Windows** -> **Package Manager Console**.

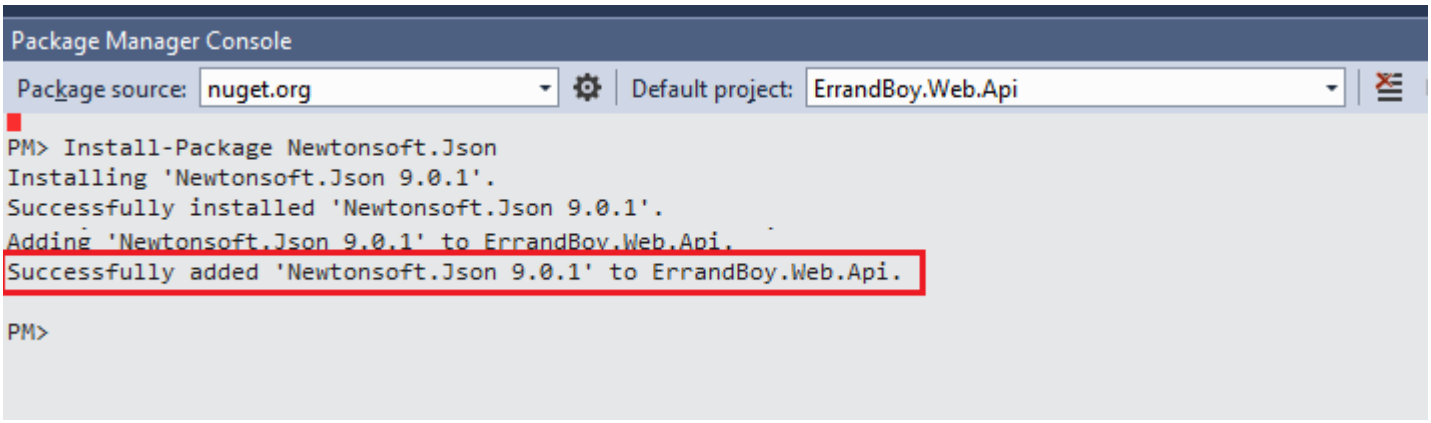
2. Once the Package Manager Console is visible, select the project within your solution, into which you want to install Json.Net, by selecting it from the **Default Project** dropdown.



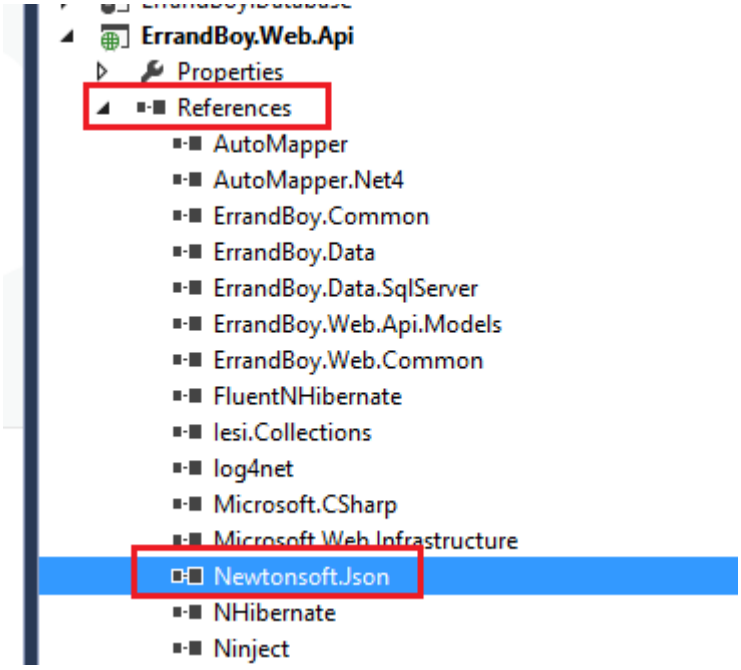
3. Now type the following command and press enter.

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

3. This will install the latest version of Json.Net. You will see the installation progress in the Package Manager Console. If successful, you will the message that it was successfully installed into your selected Project.



4. Once installed successfully, you will now be able to see the Json.Net Assembly in the references in your selected Project. The name of the Json.Net assembly is `Newtonsoft.Json`



Json.Net is now ready for use in your project!

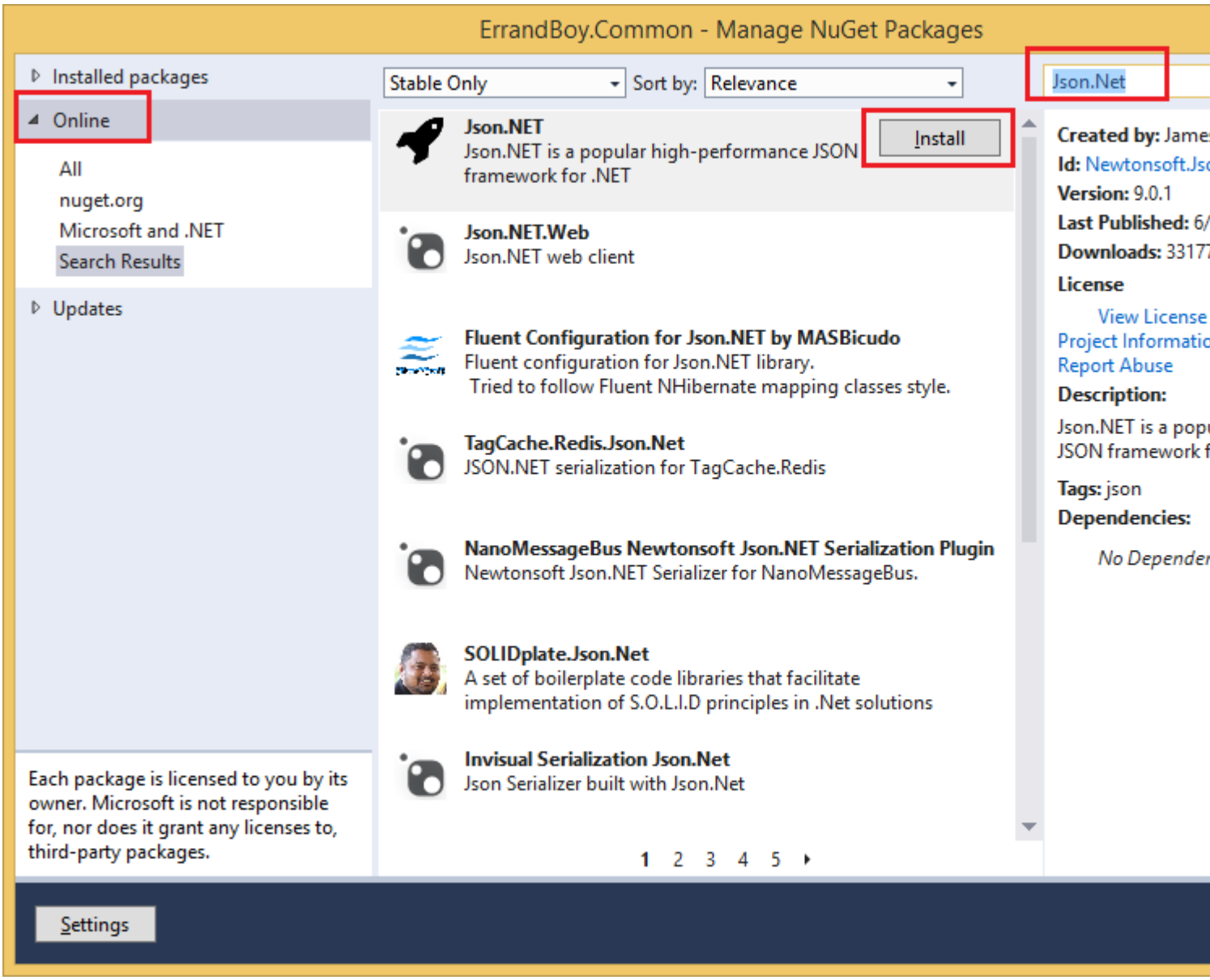
Install Json.Net using the Visual Studio Solution Explorer.

You can also install json.net using the Solution Explorer in Visual Studio.

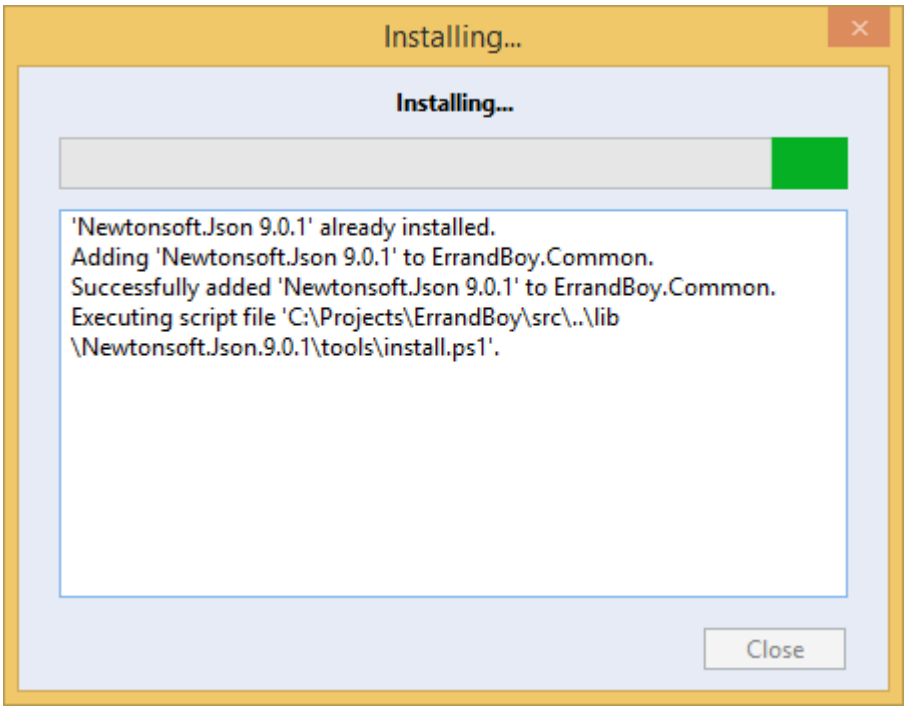
1. Right click the **References** node in your Project and click **Manage Nuget Packages...**

2. In the **Nuget Package Manager** Dialog box, make sure **Online** is selected in the left pane. Type **Json.Net** in the search box in the top right. This will display the Json.Net Nuget Package in the search results pane in the middle.

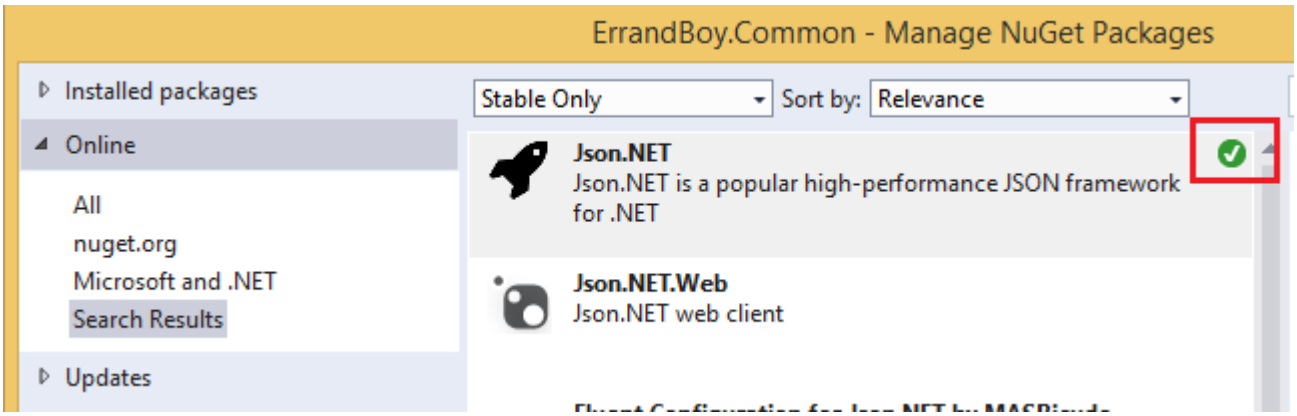
3. Click the **Install** button.



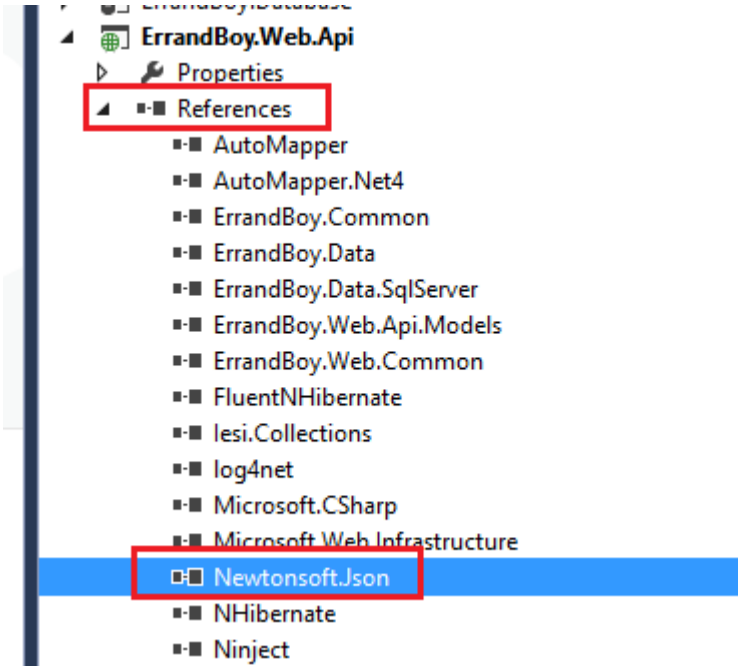
2. You will see the installation progress in the progress window that will pop up.



3. Once the installation is complete, you will see a green check-mark next to the Json.Net package in the **Nuget Package Manager** dialog box.



4. You will also see the `Newtonsoft.Json` Assembly in the **References** node in your solution explorer for your selected project.



This completes the installation of Json.Net. You are now ready to use it in your project to perform various operations on json data.

How to Serialize an Object to JSON using Json.Net in C#

The following example shows how you can use Json.Net to serialize the data in an C# Object's instance, to JSON string.

```
using System;
using System.Collections.Generic;
using Newtonsoft.Json;

public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public bool IsManager { get; set; }
    public DateTime JoinedDate { get; set; }
    public IList<string> Titles { get; set; }
}

public class Program
{
    public static void Main()
    {
        Employee employee = new Employee
        {
            FirstName = "Shiva",
            LastName = "Kumar",
            IsManager = true,
            JoinedDate = new DateTime(2013, 1, 20, 0, 0, 0, DateTimeKind.Utc),
            Titles = new List<string>
            {
                "Sr. Software Engineer",
                "Applications Architect"
            }
        };
```

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


Console.WriteLine(json);

    }
}
```

If you run this console program, the output of the `Console.WriteLine(json)` will be

```
{
  "FirstName": "Shiva",
  "LastName": "Kumar",
  "IsManager": true,
  "JoinedDate": "2013-01-20T00:00:00Z",
  "Titles": [
    "Sr. Software Engineer",
    "Applications Architect"
  ]
}
```

Few things to note

1. The following line performs the actual serialization of the data inside the `employee` class instance into a json string

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

2. The parameter `Formatting.Indented` tells Json.Net to serialize the data with indentation and new lines. If you don't do that, the serialized string will be one long string with no indentation or line breaks.

How to Deserialize JSON data to Object using Json.Net in C#

The following example shows how you can deserialize a JSON string containing into an Object (i.e. into an instance of a class).

```
using System;
using System.Collections.Generic;
using Newtonsoft.Json;

public class Program
{
    public class Employee
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public bool IsManager { get; set; }
        public DateTime JoinedDate { get; set; }
        public IList<string> Titles { get; set; }
    }

    public static void Main()
    {
```

```
        string json = @"{
                          'FirstName': 'Shiva',
                          'LastName': 'Kumar',
                          'IsManager': true,
                          'JoinedDate': '2014-02-10T00:00:00Z',
                          'Titles': [
                            'Sr. Software Engineer',
                            'Applications Architect'
                          ]
                        }";

        Employee employee = JsonConvert.DeserializeObject<Employee>(json);

        Console.WriteLine(employee.FirstName);
        Console.WriteLine(employee.LastName);
        Console.WriteLine(employee.JoinedDate);
        foreach (string title in employee.Titles)
        {
            Console.WriteLine("  {0}", title);
        }
    }
}
```

If you run this console program, the output of the various `Console.WriteLine` statements will be as follows.

```
Shiva
Kumar
2/10/2014 12:00:00 AM
  Sr. Software Engineer
  Applications Architect
```

Few things to note

1. The following line performs the actual deserialization of the data in the json string into the `employee` object instance of the `Employee` class.

```
Employee employee = JsonConvert.DeserializeObject<Employee>(json);
```

2. Since `employee.Titles` is a `List<string>` type, we use the `foreach` loop construct to loop through each item in that `List`.

Read Getting started with json.net online: https://riptutorial.com/json-net/topic/1861/getting-started-with-json-net

Credits

S. No	Chapters	Contributors
1	Getting started with json.net	Community, Shiva