# Convert JSON String To C# Object

Asked 8 years, 8 months ago Active 5 months ago Viewed 550k times

169

Trying to convert a JSON string into an object in C#. Using a really simple test case:

```
JavaScriptSerializer json_serializer = new JavaScriptSerializer();
object routes_list = json_serializer.DeserializeObject("{ \"test\":\"some data\" }");
```

The problem is that routes_list never gets set; it's an undefined object. Any ideas?

20

c# json

edited Nov 26 '12 at 17:54
yoozer8
5,870 4 40 79

asked Jan 6 '11 at 1:38
Justin
14.7k 56 144 236

1 @Greg: I actually recommend the `JavaScriptSerializer` over MS's version as it won't accept anything else but WCF's custom JSON formatting (e.g. date fields that look like dates but aren't surrounded in DATE() fail miserably) – Brad Christie Jan 6 '11 at 1:50

Also, look at this Parsing JSON objects with JavascriptSerializer in .NET article, which is actually a great tutorial. – scatmoi Oct 23 '12 at 17:50

Where are you getting JavaScriptSerializer? It is unrecognized in my C# .NET 3.5 project. – B. Clay Shannon Oct 15 '13 at 20:47

1 @B. Clay Shannon This resolved it for me stackoverflow.com/questions/7000811/… – Fuzzybear Feb 27 '15 at 18:05

## 13 Answers

It looks like you're trying to deserialize to a raw object. You could create a Class that represents the object that you're converting to. This would be most useful in cases where you're dealing with larger objects or JSON Strings.

```
class Test {

    String test;

    String getTest() { return test; }
    void setTest(String test) { this.test = test; }

}
```

Then your deserialization code would be:

```
JavaScriptSerializer json_serializer = new JavaScriptSerializer();
Test routes_list =
       (Test)json_serializer.DeserializeObject("{ \"test\":\"some data\" }");
```

More information can be found in this tutorial: http://www.codeproject.com/Tips/79435/Deserialize-JSON-with-Csharp.aspx

edited Mar 13 '15 at 20:13
Phas1c
570 6 21

answered Jan 6 '11 at 1:53
jmort253
29.5k 9 86 107

1 But in pointed article autoproperties are used. It's worth mentioning too. – Ivan Kochurkin Sep 18 '12 at 12:50

11 Sorry, but this code sample does not work. DeserializeObject gives an exception. Use var routes_list = serializer.Deserialize<Test>("{\"test\":\"some data\"}"); instead. Also, you don't need get/setTest( ), and String test, should be public. This looks more like java than C#. – dvallejo Oct 8 '13 at 16:47

as Dan Vallejo mentioned, this is an incorrect solution. After all, setTest(String test) is not returning, which is compile error as well. – Payam Aug 18 '14 at 3:05

1 Can also use this : json_serializer.Deserialize<Test>("{ \"test\":\"some data\" }"); //instead of (Test)json_serializer..... – Bashar Abu Shamaa Dec 23 '15 at 12:26

1 If you are unsure of the format for your class object, try this link. It translates your Json string into the right classes. Saved me a ton of time! – jade290 Jan 22 '16 at 16:03

Or, you can use the Newtownsoft.Json library as follows:

```
using Newtonsoft.Json;
...
var result = JsonConvert.DeserializeObject<T>(json);
```

Where `T` is your object type that matches your JSON string.

edited Jun 19 '18 at 16:01 — Legends — 9,227 5 50 87

answered Feb 15 '13 at 22:04 — tripletdad99 — 1,887 1 8 2

3 It's so easy and I thought so long about how to do this. – maracuja-juice Jun 12 '17 at 12:21

1 This was the answer I was looking for... – Prabo Sep 28 '18 at 14:23

This one worked! It should be the accepted answer! – Sam Aug 30 at 10:01

55

You probably don't want to just declare routes_list as an `object` type. It doesn't have a .test property, so you really aren't going to get a nice object back. This is one of those places where you would be better off defining a class or a struct, or make use of the dynamic keyword.

If you really want this code to work as you have it, you'll need to know that the object returned by DeserializeObject is a generic dictionary of string,object. Here's the code to do it that way:

```
var json_serializer = new JavaScriptSerializer();
var routes_list = (IDictionary<string, object>)json_serializer.DeserializeObject("{
\"test\":\"some data\" }");
Console.WriteLine(routes_list["test"]);
```

If you want to use the dynamic keyword, you can read how here.

If you declare a class or struct, you can call Deserialize instead of DeserializeObject like so:

```
class MyProgram {
    struct MyObj {
        public string test { get; set; }
    }
```

```
        var json_serializer = new JavaScriptSerializer();
        MyObj routes_list = json_serializer.Deserialize<MyObj>("{ \"test\":\"some data\"
}");
        Console.WriteLine(routes_list.test);

        Console.WriteLine("Done...");
        Console.ReadKey(true);
    }
}
```

edited May 23 '17 at 12:18 — Community ♦ 1 1

answered Jan 6 '11 at 1:51 — mattmc3 12.4k 5 64 93

Doing: json_serializer = new JavaScriptSerializer(); object routes_list = (IDictionary<string, object>)json_serializer.DeserializeObject("{ \"test\":\"some data here\" }"); Still getting 'routes_list' does not exist in the current context. – Justin Jan 6 '11 at 2:00

1 Don't use `object routes_list` . Use `var` or explicitly repeat yourself and declare routes_list as an IDictionary<string,object>. – mattmc3 Jan 6 '11 at 2:06

This worked for me, Thanks!!, The seleced solution did NOT! – Jhollman Cutcsa Jan 23 '15 at 15:10

29

Using dynamic object with JavaScriptSerializer.

```
JavaScriptSerializer serializer = new JavaScriptSerializer();
dynamic item = serializer.Deserialize<object>("{ \"test\":\"some data\" }");
string test= item["test"];

//test Result = "some data"
```

answered Apr 16 '13 at 11:51 — İbrahim Özbölük 2,718 17 20

Newtonsoft is faster than java script serializer. ... this one depends on the Newtonsoft NuGet package, which is popular and better than

```
var myclass = Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(Jsonstring);

Myclass oMyclass = Newtonsoft.Json.JsonConvert.DeserializeObject<Myclass>(Jsonstring);
```

answered Jan 21 '15 at 7:41

MSTdev
**3,167** 2 16 35

---

13

Here's a simple class I cobbled together from various posts.... It's been tested for about 15 minutes, but *seems* to work for my purposes. It uses `JavascriptSerializer` to do the work, which can be referenced in your app using the info detailed in this post.

The below code can be run in LinqPad to test it out by:

- Right clicking on your script tab in LinqPad, and choosing "Query Properties"
- Referencing the "System.Web.Extensions.dll" in "Additional References"
- Adding an "Additional Namespace Imports" of "System.Web.Script.Serialization".

Hope it helps!

```
void Main()
{
  string json = @"
  {
    'glossary':
    {
      'title': 'example glossary',
       'GlossDiv':
       {
         'title': 'S',
         'GlossList':
         {
           'GlossEntry':
           {
             'ID': 'SGML',
             'ItemNumber': 2,
             'SortAs': 'SGML',
             'GlossTerm': 'Standard Generalized Markup Language',
```

```
                {
                  'para': 'A meta-markup language, used to create markup languages such as
DocBook.',
                  'GlossSeeAlso': ['GML', 'XML']
                },
                'GlossSee': 'markup'
              }
            }
          }
      }
    }

    ";

    var d = new JsonDeserializer(json);
    d.GetString("glossary.title").Dump();
    d.GetString("glossary.GlossDiv.title").Dump();
    d.GetString("glossary.GlossDiv.GlossList.GlossEntry.ID").Dump();
    d.GetInt("glossary.GlossDiv.GlossList.GlossEntry.ItemNumber").Dump();
    d.GetObject("glossary.GlossDiv.GlossList.GlossEntry.GlossDef").Dump();
    d.GetObject("glossary.GlossDiv.GlossList.GlossEntry.GlossDef.GlossSeeAlso").Dump();
    d.GetObject("Some Path That Doesnt Exist.Or.Another").Dump();
}


// Define other methods and classes here

public class JsonDeserializer
{
    private IDictionary<string, object> jsonData { get; set; }

    public JsonDeserializer(string json)
    {
        var json_serializer = new JavaScriptSerializer();

        jsonData = (IDictionary<string, object>)json_serializer.DeserializeObject(json);
    }

    public string GetString(string path)
    {
        return (string) GetObject(path);
    }

    public int? GetInt(string path)
    {
        int? result = null;
```

```
    {
      return result;
    }

    if (o is string)
    {
      result = Int32.Parse((string)o);
    }
    else
    {
      result = (Int32) o;
    }

    return result;
  }

  public object GetObject(string path)
  {
    object result = null;

    var curr = jsonData;
    var paths = path.Split('.');
    var pathCount = paths.Count();

    try
    {
      for (int i = 0; i < pathCount; i++)
      {
        var key = paths[i];
        if (i == (pathCount - 1))
        {
          result = curr[key];
        }
        else
        {
          curr = (IDictionary<string, object>)curr[key];
        }
      }
    }
    catch
    {
      // Probably means an invalid path (ie object doesn't exist)
    }

    return result;
  }
```

edited May 23 '17 at 12:18
Community ♦
1 1

answered Sep 16 '11 at 14:55
Brad Parks
32.9k 41 175 241

11

You can accomplished your requirement easily by using Newtonsoft.Json library. I am writing down the one example below have a look into it.

**Class for the type of object you receive:**

```
public class User
{
    public int ID { get; set; }
    public string Name { get; set; }

}
```

**Code:**

```
static void Main(string[] args)
{

      string json = "{\"ID\": 1, \"Name\": \"Abdullah\"}";

      User user = JsonConvert.DeserializeObject<User>(json);

      Console.ReadKey();
}
```

this is a very simple way to parse your json.

answered Jun 17 '15 at 12:14
Mirza Danish Baig
520 4 17

add this ddl to reference to your project: **System.Web.Extensions.dll**

```
public class IdName
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

```
   string jsonStringSingle = "{'Id': 1, 'Name':'Thulasi Ram.S'}".Replace("'", "\"");
   var entity = new JavaScriptSerializer().Deserialize<IdName>(jsonStringSingle);

   string jsonStringCollection = "[{'Id': 2, 'Name':'Thulasi Ram.S'},{'Id': 2,
'Name':'Raja Ram.S'},{'Id': 3, 'Name':'Ram.S'}]".Replace("'", "\"");
   var collection = new JavaScriptSerializer().Deserialize<IEnumerable<IdName>>
(jsonStringCollection);
```

edited Oct 26 '12 at 8:02

answered Jul 17 '12 at 12:56

Thulasiram
**7,267** 6 37 41

---

As tripletdad99 said

7

```
var result = JsonConvert.DeserializeObject<T>(json);
```

but if you don't want to create an extra object you can make it with `Dictionary` instead

```
var result = JsonConvert.DeserializeObject<Dictionary<string, string>>(json_serializer);
```

edited Oct 30 '18 at 8:45

Ognyan Dimitrov
**4,312** 1 30 58

answered Oct 26 '18 at 11:51

stanimirsp
**385** 5 15

---

Another fast and easy way to semi-automate these steps is to:

1. take the JSON you want to parse and paste it here: http://json2csharp.com/ then copy and paste the resulting into a new class (ex:

3. In visual studio go to Website -> Manage Packages and use NuGet to add Json.Net from Newtonsoft.

Now use code like:

```
WebClient client = new WebClient();

string myJSON = client.DownloadString("https://URL_FOR_JSON.com/JSON_STUFF");

var myClass = Newtonsoft.Json.JsonConvert.DeserializeObject(myJSON);
```

answered Nov 3 '17 at 5:32

Jason Hitchings
**461** 4 8

---

3

Copy your Json and paste at textbox on json2csharp and click on Generate button.

A *cs* class will be generated use that cs file as below

```
var generatedcsResponce = JsonConvert.DeserializeObject(yourJson);
```

Where `RootObject` is the name of the generated *cs* file;

edited Apr 1 at 14:34

Wictor Chaves
**282** 6 14

answered Jan 17 at 10:19

Islam
**41** 1

---

0

Convert a JSON string into an object in C#. Using below test case.. its worked for me. Here "**MenuInfo**" is my C# class object.

```
JsonTextReader reader = null;
try
{
    WebClient webClient = new WebClient();
    JObject result = JObject.Parse(webClient.DownloadString("YOUR URL"));
    reader = new JsonTextReader(new System.IO.StringReader(result.ToString()));
```

```
{}

JsonSerializer serializer = new JsonSerializer();
MenuInfo menuInfo = serializer.Deserialize<MenuInfo>(reader);
```

edited Dec 20 '16 at 8:53

MNie
775 1 9 29

answered Dec 20 '16 at 8:19

Bhaskar
31 2

---

0

First you have to include library like:

```
using System.Runtime.Serialization.Json;

DataContractJsonSerializer desc = new DataContractJsonSerializer(typeof(BlogSite));
string json = "{\"Description\":\"Share knowledge\",\"Name\":\"zahid\"}";

using (var ms = new MemoryStream(ASCIIEncoding.ASCII.GetBytes(json)))
{
    BlogSite b = (BlogSite)desc.ReadObject(ms);
    Console.WriteLine(b.Name);
    Console.WriteLine(b.Description);
}
```

edited Dec 10 '17 at 16:07

0xdb
1,774 1 11 23

answered Dec 10 '17 at 14:58

Muhammad Zahid
1

---