How do I turn a C# object into a JSON string in .NET?

Ask Question

736

124

I have classes like these:

```
class MyDate
{
    int year, month, day;
}

class Lad
{
    string firstName;
    string lastName;
    MyDate dateOfBirth;
}
```

And I would like to turn a `Lad` object into a JSON string like this:

```
{
    "firstName":"Markoff",
    "lastName":"Chaney",
    "dateOfBirth":
    {
        "year":"1901",
        "month":"4",
        "day":"30"
    }
}
```

(without the formatting). I found this link, but it uses a namespace that's not in .NET 4. I also heard about JSON.NET, but their site seems to be down at the moment, and I'm not keen on using external DLL files. Are there other options besides manually creating a JSON string writer?

c# .net json serialization

edited Mar 2 '18 at 10:56
Liam
16.3k 16 76 130

asked Jun 1 '11 at 12:59
Hui
4,294 8 20 20

1 JSON.net can be loaded here An other and faster (as they say - I did not test it myself) solution is ServiceStack.Text I would not recommend rolling your own JSON parser. It will likely be slower and more error prone or you have to invest lots of time. – Zebi Jun 1 '11 at 13:01

yes. C# has a type called JavaScriptSerializer – Glenn Ferrie Jun 1 '11 at 13:02

possible duplicate of Generics / JSON JavaScriptSerializer C# – Filip Ekberg Jun 1 '11 at 13:03

2 Hm.. as far as I can see you should be able to use: msdn.microsoft.com/en-us/library/… Which is also in .Net 4.0 according to the MSDN page. You should be able to use the Serialize(Object obj) method: msdn.microsoft.com/en-us/library/bb292287.aspx Am I missing something here? Btw. you link seems to be a some code and not a link – Holger Jun 1 '11 at 13:04

Not to mention it has no dependencies on the System.Web.Xyz namespaces... – Dave Jellison Mar 6 '13 at 18:58

14 Answers

You could use the `JavaScriptSerializer` class (add reference to `System.Web.Extensions`):

```
using System.Web.Script.Serialization;
```

771

```
var json = new JavaScriptSerializer().Serialize(obj);
```

A full example:

```
using System;
using System.Web.Script.Serialization;

public class MyDate
{
    public int year;
    public int month;
    public int day;
}

public class Lad
{
    public string firstName;
    public string lastName;
    public MyDate dateOfBirth;
}

class Program
{
    static void Main()
    {
        var obj = new Lad
        {
            firstName = "Markoff",
            lastName = "Chaney",
            dateOfBirth = new MyDate
            {
                year = 1901,
                month = 4,
                day = 30
            }
        };
        var json = new JavaScriptSerializer().Serialize(obj);
        Console.WriteLine(json);
    }
}
```

edited Oct 12 '15 at 19:33

Adrian H.
361 5 16

answered Jun 1 '11 at 13:05

Darin Dimitrov
847k 223 3022 2749

61 Please have in mind that Microsoft suggests to use JSON.net instead of this solution. I think that this answer became inappropriate. Take a look at willsteel's answer. Source: https://msdn.microsoft.com/en-us/library/system.web.script.serialization.javascriptserializer.aspx. – rzelek Feb 1 '16 at 15:12

8 @DarinDimitrov you should consider adding a hint about JSON.net. Microsoft recommends it over JavascriptSerializer: msdn.microsoft.com/en-us/library/... You could also add a hint to msdn.microsoft.com/en-us/library/... which is the framework included approach – Mafii Jul 6 '16 at 10:19

here is **online tool** to convert your `classes` to `json` format, hope helps someone. – stom Jan 25 '17 at 8:24

using System.Web.Script.Serialization; – Happy Bird Feb 13 '17 at 15:31

5 Why would Microsoft recommend a 3rd party solution over their own? Their wording is very odd as well: "Json.NET should be used serialization and deserialization. Provides serialization and deserialization functionality for AJAX-enabled applications." – Protector one Mar 17 '17 at 9:53

809

Since we all love one liners

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

```
Newtonsoft.Json.JsonConvert.SerializeObject(new {foo = "bar"})
```

Documentation: **Serializing and Deserializing JSON**

edited Jan 30 '15 at 22:09
James Newton-King
31k 19 95 117

answered Oct 2 '13 at 12:39
willsteel
12.1k 4 25 31

113 Newtonsoft serializer is way faster and mor customizable then built in. Highly recommend to use it. Thanks for the answer @willsteel – Andrei Oct 2 '13 at 13:04

8 @JosefPfleger the pricing is for JSON.NET Schema, not JSON.NET the regular serializer, which is MIT – David Cumps Jun 3 '15 at 19:51

1 @DavidCumps right, thank you for the clarification – Josef Pfleger Jun 10 '15 at 9:03

1 My testing showed that Newtonsoft is slower than JavaScriptSerializer class. (.NET 4.5.2) – nemke Sep 28 '15 at 11:34

26 If you read the MSDN documentation for JavaScriptSerializer, it flat out says use JSON.net. – dsghi Nov 5 '15 at 6:10

59

Use Json.Net library, you can download it from Nuget Packet Manager.

Serializing to Json String:

```
var obj = new Lad
        {
            firstName = "Markoff",
            lastName = "Chaney",
            dateOfBirth = new MyDate
```

```
            {
                year = 1901,
                month = 4,
                day = 30
            }
        };

var jsonString = Newtonsoft.Json.JsonConvert.SerializeObject(obj);
```

Deserializing to Object:

```
var obj = Newtonsoft.Json.JsonConvert.DeserializeObject<Lad>(jsonStr
```

answered Jun 21 '17 at 8:15

Gokulan P H
975 5 10

53

Use the `DataContractJsonSerializer` class: MSDN1, MSDN2.

My example: HERE.

It can also safely deserialize objects from a JSON string, unlike `JavaScriptSerializer`. But personally I still prefer Json.NET.

edited Jul 15 '16 at 14:09

answered Jun 1 '11 at 13:03

Edgar
2,951 4 34 54

5 Not to mention it has no dependencies on the System.Web.Xyz namespaces... – Dave Jellison Mar 6 '13 at 18:58

1 Still don't see any examples on *that page*, but here are some on MSDN

and elsewhere -> the last one uses extension methods to achieve one-liners. – Cristi Diaconescu Jan 17 '15 at 21:26

Oh, I missed the 2nd MSDN link :) – Cristi Diaconescu Jan 19 '15 at 8:48

2 It doesn't serialize plain classes. The error reported "Consider marking it with the DataContractAttribute attribute, and marking all of its members you want serialized with the DataMemberAttribute attribute. If the type is a collection, consider marking it with the CollectionDataContractAttribute." – Michael Freidgeim Jul 14 '16 at 2:56

1 @MichaelFreidgeim Which is better depends on the requirements. The attributes let you configure how the property is serialized. – Edgar Jul 15 '16 at 13:49

21

Wooou! Really better using a JSON framework :)

Here is my example using Json.NET (http://james.newtonking.com/json):

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

namespace com.blogspot.jeanjmichel.jsontest.model
{
    public class Contact
    {
        private Int64 id;
        private String name;
        List<Address> addresses;

        public Int64 Id
        {
            set { this.id = value; }
            get { return this.id; }
        }

        public String Name
        {
```

```
        set { this.name = value; }
        get { return this.name; }
    }

    public List<Address> Addresses
    {
        set { this.addresses = value; }
        get { return this.addresses; }
    }

    public String ToJSONRepresentation()
    {
        StringBuilder sb = new StringBuilder();
        JsonWriter jw = new JsonTextWriter(new StringWriter(sb))

        jw.Formatting = Formatting.Indented;
        jw.WriteStartObject();
        jw.WritePropertyName("id");
        jw.WriteValue(this.Id);
        jw.WritePropertyName("name");
        jw.WriteValue(this.Name);

        jw.WritePropertyName("addresses");
        jw.WriteStartArray();

        int i;
        i = 0;

        for (i = 0; i < addresses.Count; i++)
        {
            jw.WriteStartObject();
            jw.WritePropertyName("id");
            jw.WriteValue(addresses[i].Id);
            jw.WritePropertyName("streetAddress");
            jw.WriteValue(addresses[i].StreetAddress);
            jw.WritePropertyName("complement");
            jw.WriteValue(addresses[i].Complement);
            jw.WritePropertyName("city");
            jw.WriteValue(addresses[i].City);
            jw.WritePropertyName("province");
            jw.WriteValue(addresses[i].Province);
            jw.WritePropertyName("country");
            jw.WriteValue(addresses[i].Country);
            jw.WritePropertyName("postalCode");
            jw.WriteValue(addresses[i].PostalCode);
            jw.WriteEndObject();
        }
```

```
            jw.WriteEndArray();

            jw.WriteEndObject();

            return sb.ToString();
        }

        public Contact()
        {
        }

        public Contact(Int64 id, String personName, List<Address> ad
        {
            this.id = id;
            this.name = personName;
            this.addresses = addresses;
        }

        public Contact(String JSONRepresentation)
        {
            //To do
        }
    }
}
```

The test:

```
using System;
using System.Collections.Generic;
using com.blogspot.jeanjmichel.jsontest.model;

namespace com.blogspot.jeanjmichel.jsontest.main
{
    public class Program
    {
        static void Main(string[] args)
        {
            List<Address> addresses = new List<Address>();
            addresses.Add(new Address(1, "Rua Dr. Fernandes Coelho, 
"São Paulo", "São Paulo", "Brazil", "05423040"));
            addresses.Add(new Address(2, "Avenida Senador Teotônio V
"São Paulo", "São Paulo", "Brazil", null));

            Contact contact = new Contact(1, "Ayrton Senna", address
```

```
            Console.WriteLine(contact.ToJSONRepresentation());
            Console.ReadKey();
        }
    }
}
```

The result:

```
{
  "id": 1,
  "name": "Ayrton Senna",
  "addresses": [
    {
      "id": 1,
      "streetAddress": "Rua Dr. Fernandes Coelho, 85",
      "complement": "15º andar",
      "city": "São Paulo",
      "province": "São Paulo",
      "country": "Brazil",
      "postalCode": "05423040"
    },
    {
      "id": 2,
      "streetAddress": "Avenida Senador Teotônio Vilela, 241",
      "complement": null,
      "city": "São Paulo",
      "province": "São Paulo",
      "country": "Brazil",
      "postalCode": null
    }
  ]
}
```

Now I will implement the constructor method that will receives a JSON string and populates the class' fields.

edited May 7 '15 at 15:23

Majid

8,146 8 57 101

answered Feb 4 '14 at 20:07

Jean J. Michel

376 3 10

1 Good post, this is the most current way to do it. – MatthewD May 30 '16 at 13:33

4

If you are in an ASP.NET MVC web controller it's as simple as:

```
string ladAsJson = Json(Lad);
```

Can't believe no one has mentioned this.

answered Sep 16 '16 at 15:53

micahhoover
1,280 4 24 43

I get an error about not being able to cast jsonresult to string. – csga5000 Sep 23 '16 at 5:30

It will compile with implicit typing: var ladAsJson = Json(Lad). – ewomack Nov 7 '16 at 19:55

4

If they are not very big, whats probably your case export it as Json. Also this makes portable among all plattforms

```
using Newtonsoft.Json;
    [TestMethod]
       public void ExportJson()
       {
       double[,] b = new double[,] {
           { 110, 120, 130, 140, 150 },
           { 1110, 1120, 1130, 1140, 1150 },
```

```
            { 1000, 1, 5 ,9, 1000},
            {1110, 2, 6 ,10,1110},
            {1220, 3, 7 ,11,1220},
            {1330, 4, 8 ,12,1330} };


        string jsonStr = JsonConvert.SerializeObject(b);

        Console.WriteLine(jsonStr);

        string path =
"X:\\Programming\\workspaceEclipse\\PyTutorials\\src\\tensorflow_tut

        File.WriteAllText(path, jsonStr);
    }
```

answered Jan 28 '18 at 16:00

user8426627
96 8

2

Use this tools for generate C# class, then use this code to serialize your object

```
var json = new JavaScriptSerializer().Serialize(obj);
```

edited Aug 17 '18 at 15:14

Mathis Perrier
21 6

answered Nov 21 '14 at 22:18

Artem Polischuk
193 2 13

2

As easy as this, works for dynamic objects as well (type object):

```
string json = new
System.Web.Script.Serialization.JavaScriptSerializer().Serialize(MYO
```

edited Apr 27 '15 at 6:24

Chandan Kumar
3,127 3 27 54

answered Jun 26 '14 at 14:02

MarzSocks
2,801 2 15 28

there is no default script under web. :(– Mahdi Rafatjah Jul 11 '16 at 7:17

You are looking for this:msdn.microsoft.com/en-us/library/... – MarzSocks Jul 11 '16 at 19:08

I kind of tried that but no. Script I guess I should add it as reference. So thanks a lot – Mahdi Rafatjah Jul 12 '16 at 3:43

2

Use the below code for converting XML to JSON.

```
var json = new JavaScriptSerializer().Serialize(obj);
```

edited Nov 9 '14 at 9:46

Peter Mortensen
13.7k 19 86 113

answered Mar 6 '14 at 9:39

Hithesh
352 2 9

1

I would vote for ServiceStack's JSON Serializer:

```
using ServiceStack.Text

string jsonString = new { FirstName = "James" }.ToJson();
```

It is also the fastest JSON serializer available for .NET: http://www.servicestack.net/benchmarks/

answered Oct 2 '13 at 13:39
James
1,157 13 44

4 Those are very old benchmarks there. I've just test all three current versions of Newtonsoft, ServiceStack and JavaScriptSerializer and currently Newtonsoft is the fastest. Tho they all do quite fast. – Michael Logutov Oct 25 '13 at 9:22

1 ServiceStack doesn't appear to be free. – joelnet Dec 19 '13 at 0:25

@joelnet this is now the case, but was free when answering the question. However it is free for small sites, and I am still using it even though it is paid, it is a superb framework. – James Dec 19 '13 at 10:21

5 Dead link for the benchmark – Thomas May 7 '15 at 12:03

Some benchmarks here, though there's non for the serialization on its own: docs.servicestack.net/real-world-performance – JohnLBevan Dec 13 '18 at 9:39

0

There is this really nifty utility right here: http://csharp2json.io/

answered Apr 24 '18 at 18:03
jallen

335 7 31

0

Serializer

```
 public static void WriteToJsonFile<T>(string filePath, T objectToWr
false) where T : new()
{
        var contentsToWriteToFile = JsonConvert.SerializeObject(obje
JsonSerializerSettings
        {
            Formatting = Formatting.Indented,
        });
        using (var writer = new StreamWriter(filePath, append))
        {
            writer.Write(contentsToWriteToFile);
        }
}
```

Object

```
namespace MyConfig
{
    public class AppConfigurationSettings
    {
        public AppConfigurationSettings()
        {
            /* initialize the object if you want to output a new doc
             * for use as a template or default settings possibly wh
             * an app is started.
             */
            if (AppSettings == null) { AppSettings=new AppSettings()
        }

        public AppSettings AppSettings { get; set; }
    }

    public class AppSettings
    {
        public bool DebugMode { get; set; } = false;
    }
}
```

Implementation

```
var jsonObject = new AppConfigurationSettings();
WriteToJsonFile<AppConfigurationSettings>(file.FullName, jsonObject)
```

Output

```
{
  "AppSettings": {
    "DebugMode": false
  }
}
```

edited Feb 3 '16 at 19:31

answered Feb 3 '16 at 18:00

C0r3yh
1,569 19 24

-7

Take care to create your class with the right attribute too:

Create this class with <Serializable> attribute as per the example C# example followed by vb.net exmpale

C#

```
using Microsoft.VisualBasic;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Script.Serialization;
```

```
namespace Samples
{
[Serializable()]
public class Customer
{


    private int _idcustomer;
    public int IDCustomer {
        get { return _idcustomer; }
        set { _idcustomer = value; }
    }


    private System.DateTime _RegistrationDate;
    public System.DateTime RegistrationDate {
        get { return _RegistrationDate; }
        set { _RegistrationDate = value; }
    }


    private string _Name;
    public string Name {
        get { return _Name; }
        set { _Name = value; }
    }


    private string _Surname;
    public string Surname {
        get { return _Surname; }
        set { _Surname = value; }
    }
}


[Serializable()]
public class Product
{


    private int _ProductID;
    public int ProductID {
        get { return _ProductID; }
        set { _ProductID = value; }
    }
```

```
        private string _ProductName;
        public string ProductName {
            get { return _ProductName; }
            set { _ProductName = value; }
        }


        private int _Price;
        public int Price {
            get { return _Price; }
            set { _Price = value; }
        }


        private bool _inStock;
        public bool inStock {
            get { return _inStock; }
            set { _inStock = value; }
        }
    }


    [Serializable()]
    public class Order
    {


        private int _OrderId;
        public int OrderID {
            get { return _OrderId; }
            set { _OrderId = value; }
        }


        private int _customerID;
        public int CustomerID {
            get { return _customerID; }
            set { _customerID = value; }
        }


        private List<Product> _ProductsList;
        public List<Product> ProductsList {
            get { return _ProductsList; }
            set { _ProductsList = value; }
        }
```

```
    private System.DateTime _PurchaseDate;
    public System.DateTime PurchaseDate {
        get { return _PurchaseDate; }
        set { _PurchaseDate = value; }
    }


    private string _PaymentMethod;
    public string PaymentMethod {
        get { return _PaymentMethod; }
        set { _PaymentMethod = value; }
    }

    public string ToJson()
    {
        string json = string.Empty;
        JavaScriptSerializer js = new JavaScriptSerializer();
        json = js.Serialize(this);
        js = null;
        return json;
    }

}

}
```

VBNET EXAMPLE

```
Imports System
Imports System.Web
Imports System.Web.Script.Serialization
Namespace Samples
<Serializable()>
Public Class Customer

    Private _idcustomer As Integer

    Public Property IDCustomer() As Integer
        Get
            Return _idcustomer
        End Get
        Set(ByVal value As Integer)
            _idcustomer = value
        End Set
```

```
        End Property

        Private _RegistrationDate As Date

        Public Property RegistrationDate() As Date
            Get
                Return _RegistrationDate
            End Get
            Set(ByVal value As Date)
                _RegistrationDate = value
            End Set
        End Property

        Private _Name As String

        Public Property Name() As String
            Get
                Return _Name
            End Get
            Set(ByVal value As String)
                _Name = value
            End Set
        End Property

        Private _Surname As String

        Public Property Surname() As String
            Get
                Return _Surname
            End Get
            Set(ByVal value As String)
                _Surname = value
            End Set
        End Property
    End Class


    <Serializable()>
    Public Class Product

        Private _ProductID As Integer

        Public Property ProductID() As Integer
            Get
                Return _ProductID
            End Get
            Set(ByVal value As Integer)
```

```
                _ProductID = value
            End Set
        End Property

        Private _ProductName As String

        Public Property ProductName() As String
            Get
                Return _ProductName
            End Get
            Set(ByVal value As String)
                _ProductName = value
            End Set
        End Property

        Private _Price As Integer

        Public Property Price() As Integer
            Get
                Return _Price
            End Get
            Set(ByVal value As Integer)
                _Price = value
            End Set
        End Property

        Private _inStock As Boolean

        Public Property inStock() As Boolean
            Get
                Return _inStock
            End Get
            Set(ByVal value As Boolean)
                _inStock = value
            End Set
        End Property
    End Class


    <Serializable>
    Public Class Order

        Private _OrderId As Integer

        Public Property OrderID() As Integer
            Get
                Return _OrderId
```

```
        End Get
        Set(ByVal value As Integer)
            _OrderId = value
        End Set
    End Property

    Private _customerID As Integer

    Public Property CustomerID() As Integer
        Get
            Return _customerID
        End Get
        Set(ByVal value As Integer)
            _customerID = value
        End Set
    End Property

    Private _ProductsList As List(Of Product)

    Public Property ProductsList() As List(Of Product)
        Get
            Return _ProductsList
        End Get
        Set(ByVal value As List(Of Product))
            _ProductsList = value
        End Set
    End Property

    Private _PurchaseDate As Date

    Public Property PurchaseDate() As Date
        Get
            Return _PurchaseDate
        End Get
        Set(ByVal value As Date)
            _PurchaseDate = value
        End Set
    End Property

    Private _PaymentMethod As String

    Public Property PaymentMethod() As String
        Get
            Return _PaymentMethod
        End Get
        Set(ByVal value As String)
            _PaymentMethod = value
```

```
        End Set
    End Property

    Public Function ToJson() As String
        Dim json As String = String.Empty
        Dim js As New JavaScriptSerializer
        json = js.Serialize(Me)
        js = Nothing
        Return json
    End Function

End Class
```

End Namespace

The second step is to create a simple test data like this:

C#

```
  void Main() {
    List<Samples.Product> ListProducts = new List<Samples.Product>()
    ListProducts.Add(new Samples.Product(), With, {.inStock=False,.P
.ProductID=1,.ProductName=BookOne);
    ListProducts.Add(new Samples.Product(), With, {.inStock=False,. 
.ProductID=2, .ProductName=Hotels California);
    ListProducts.Add(new Samples.Product(), With,
{.inStock=False,.Price=10,.ProductID=3,.ProductName=Cbr);
    ListProducts.Add(new Samples.Product(), With,
{.inStock=False,.Price=10,.ProductID=4,.ProductName=Mustang);
    ListProducts.Add(new Samples.Product(), With, {.inStock=False,.P
.ProductID=15,.ProductName=Anything);
    ListProducts.Add(new Samples.Product(), With,
{.inStock=False,.Price=10,.ProductID=38,.ProductName=Monster Truck);
    Samples.Customer Customer = new Samples.Customer();
    // With...
    Customer.IDCustomer = 1;
    Customer.Name = "Customer1";
    Customer.RegistrationDate = Now;
    Customer.Surname = "SurnameCustomer";
    Samples.Order Order = new Samples.Order();
    // With...
    Order.CustomerID = Customer.IDCustomer;
    Order.OrderID = 1;
    Order.PaymentMethod = "PayPal";
    Order.ProductsList = ListProducts;
```

```
    Order.PurchaseDate = Now;
    Console.WriteLine(Order.ToJson);
    Console.ReadLine();
}
```

VB.NET

```
Sub Main()
    Dim ListProducts As New List(Of Samples.Product)

    ListProducts.Add(New Samples.Product With {.inStock = False, .Pr
                     .ProductID = 1, .ProductName = "BookOne"})
    ListProducts.Add(New Samples.Product With {.inStock = False, .Pr
                     .ProductID = 2, .ProductName = "Hotels Californ
    ListProducts.Add(New Samples.Product With {.inStock = False, .Pr
                     .ProductID = 3, .ProductName = "Cbr"})
    ListProducts.Add(New Samples.Product With {.inStock = False, .Pr
                     .ProductID = 4, .ProductName = "Mustang"})
    ListProducts.Add(New Samples.Product With {.inStock = False, .Pr
                     .ProductID = 15, .ProductName = "Anything"})
    ListProducts.Add(New Samples.Product With {.inStock = False, .Pr
                     .ProductID = 38, .ProductName = "Monster Truck"

    Dim Customer As New Samples.Customer
    With {.IDCustomer = 1, .Name = "Customer1",.RegistrationDate = N
="SurnameCustomer"}

    Dim Order As New Samples.Order With {
        .CustomerID = Customer.IDCustomer,
        .OrderID =       1,
        .PaymentMethod = "PayPal",
        .ProductsList = ListProducts,
        .PurchaseDate = Now
    }
    Console.WriteLine(Order.ToJson)
    Console.ReadLine()
End Sub
```

And this is the final result:


```
{"OrderID":1,"CustomerID":1,"ProductsList":[{"ProductID":1,"ProductN
 e","Price":10,"inStock":false},{"ProductID":2,"ProductName":"Hotels
 "Price":10,"inStock":false},{"ProductID":3,"ProductName":"Cbr","Pri
 ck":false},{"ProductID":4,"ProductName":"Mustang","Price":10,"inSto
```

```
ProductID":15,"ProductName":"Anything","Price":10,"inStock":false},
38,"ProductName":"Monster Truck","Price":10,"inStock":false}],"Purcl
Date(1396642206155)\/","PaymentMethod":"PayPal"}
```


Remember to add a reference to system.web.extension.dll in order to achive your goal.

edited Jun 5 '16 at 12:54

answered Apr 4 '14 at 20:30

makemoney2010
1,238 7 9

11 Sorry, but what was the point of offering your answer to a well-answered question especially when it's not even in the same programming language? – JW Lim May 10 '14 at 2:22

It is simple :) => anyone con go here and translate it developerfusion.com/tools/convert/csharp-to-vb i forget to mention in the post. Usually i use to post both suggestion in this case i forget it :) lol that's all. C# and vbnet can be translated in a simple manner and it is very usefull. – makemoney2010 May 10 '14 at 15:57

protected by Brian Rogers Nov 23 '14 at 0:39

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?