

Visual Studio how to serialize object from debugger

Asked 6 years ago Active 13 days ago Viewed 28k times



69



26

I'm trying to investigate a bug in a crash dump (so I can not change the code). I have a really complicated object (thousands of lines in the serialized representation) and its state is inconsistent. To investigate its state the Visual Studio debugger view is useless. But the object has a data contract. I'd like to serialize it and then use my favorite text editor to navigate across the object. Is it possible to do the from the debugger?

c#

visual-studio-2012

edited Jul 1 '15 at 18:52



Josh

1,228

1

13

17

asked Sep 13 '13 at 19:57



xvorsx

1,236

1

12

18

Note, that if you have some custom container class, or some other class what you want to see many times during debug, but the IntelliSense and QuickView cannot figure it out, you can write an extension for VS which helps to show your custom class in debug. – [Csaba Toth](#) Sep 13 '13 at 20:34

Many good techniques can also be found [here] (stackoverflow.com/questions/360277/...) – [Josh](#) Jul 1 '15 at 19:43

10 Answers



67



Some time ago I wrote this one-liner serializing an object to a file on the disk. Copy/paste it to your Immediate window, and replace `obj` (it's referenced twice) with your object. It'll save a `text.xml` file to `c:\temp`, change it to your liking.



```
(new System.Xml.Serialization.XmlSerializer(obj.GetType())).Serialize(new  
System.IO.StreamWriter(@"c:\temp\text.xml"), obj)
```

Don't expect any magic though, if the object cannot be serialized, it'll throw an exception.

edited Aug 29 at 17:37

answered Jul 3 '15 at 6:32

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

- 2 This worked for me in the immediate window. upvoted, – [Pankaj Kumar](#) Feb 16 '16 at 13:40
- 1 when I use this on the VS 2015 immediate window, I get this 'error': "Evaluation of native methods in this context is not supported." Ideas? – [Vetras](#) Oct 14 '16 at 10:25 
- When I run it, I get the following message: identifier "System" is undefined – [Rasoul](#) Aug 29 '17 at 8:00 
- Had an old VB.NET project, had to put it like this otherwise I was getting an error about expression syntax, if anyone needs: new System.Xml.Serialization.XmlSerializer(obj.GetType()).Serialize(New System.IO.StreamWriter("C:\temp\temp.txt"), obj) – [Liquid Core](#) Jan 16 '18 at 16:26
- This is spectacular. I had to use a different directory due to write permissions, but it worked like a charm. Import into Excel and it's even pretty. – [BuddyZ](#) Mar 19 at 21:18



119



With any luck you have Json.Net in you appdomain already. In which case pop this into your Immediate window:

```
Newtonsoft.Json.JsonConvert.SerializeObject(someVariable)
```

answered Apr 29 '14 at 10:30



[mcintyre321](#)

9,182 8 53 94

- 12 Wish I could upvote this again, especially compared to the other answers. (Sorry, but I don't need to see another line of XML in my career.) – [yzorg](#) Nov 21 '14 at 15:43
- 1 After a very long test debug session an exception happened in my program before it could write out thousands of test results to a file, I was on a breakpoint where the exception occurred and could still inspect the results collection. This tip saved me a lot of time! – [HAL9000](#) Dec 1 '15 at 3:10
- 1 You might need to actually use Json.Net somewhere else too so that it is loaded when you try to use it in the Immediate window (as opposed to just adding the reference). – [Evan](#) Feb 17 '17 at 22:25
- 1 I had to add the Newtonsoft.Json package using Nuget, and also add a line of code in the method that I had the breakpoint in to create a dummy Newtonsoft.Json.JsonArrayAttribute() class to get it to work. A most excellent solution! – [Richard Moore](#) Jan 17 '18 at 19:47



Here is a Visual Studio extension which will let you do exactly that:

<https://visualstudiogallery.msdn.microsoft.com/c6a21c68-f815-4805-000f-ed0885d8771f>

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

edited Apr 16 '15 at 2:30

answered Feb 21 '15 at 8:13



Omar Elabd

1,046 13 21

-
- 4 That link seems to be broken, but [here](#) is the github project and you can find it if you search for "Object Exporter" in the "Extensions and Updates..." dialog in Visual Studio. Great extension btw! – [Niklas Söderberg](#) May 18 '15 at 6:35
-
- 2 Thank you @Omar The idea is perfect. But it takes too long time and freeze in some cases – [Wahid Bitar](#) Feb 22 '17 at 11:47
-
- 1 Agreed with @WahidBitar - Great concept - perfect for setting up unit test data, but the extension seems quite buggy, and takes Visual Studio with it when it crashes! – [Dib](#) Mar 22 '18 at 13:47
-
- This is really a very useful tool. – [Ashutosh Singh](#) Jul 19 '18 at 18:29
-

▲ I have an extension method I use:

3

```
public static void ToSerializedObjectForDebugging(this object o, FileInfo saveTo)
{
    Type t = o.GetType();
    XmlSerializer s = new XmlSerializer(t);
    using (FileStream fs = saveTo.Create())
    {
        s.Serialize(fs, o);
    }
}
```

I overload it with a string for saveTo and call it from the immediate window:

```
public static void ToSerializedObjectForDebugging(this object o, string saveTo)
{
    ToSerializedObjectForDebugging(o, new FileInfo(saveTo));
}
```

answered Sep 13 '13 at 20:09



Mike Cheel

8,442 7 54 82

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

▲ It might be possible to use the immediate window to serialize it and then copy the content to your favorite editor.

3 Another option is to override the `ToString()` method and call it while in debug mode.

▼ You can also write the contents out to a file shortly before the crash, or wrap the code into a try/catch and write the file then. I'm assuming you can identify when it's crashing.

edited Aug 29 at 17:53



Sergey

959 1 17 31

answered Sep 13 '13 at 19:59



Chuck Conway

13.7k 10 51 95

Thanks, I tried same from Watch window, but it told me "The function evaluation requires all threads to run." Immediate window solve it – [xvorsx](#) Sep 13 '13 at 20:04

▲ A variation on the answer from Omar Elabd --

1 It's not free, but there's a free trial for OzCode (<https://marketplace.visualstudio.com/items?itemName=CodeValueLtd.OzCode>).

▼ There's built-in exporting to JSON within the context/hover menu there, and it works a bit better than the Object Export extension (the trade-off for it not being free).

<http://o.oz-code.com/features#export> (demo)

I know this is a few years after the fact, but I'm leaving an answer here because this worked for me, and someone else may find it useful.

answered Jun 28 '18 at 21:21



Michael Armes

864 2 11 25

▲ In case you have a circular reference, run this in the immediate Window:

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```
ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Serialize  
});
```

edited Aug 30 at 16:46



SQLGeorge

2,614 1 15 43

answered Feb 7 at 17:47



InGeek

778 2 11 29

I've been using [ObjectDumper.Net](#).

1

It works well, especially if you have live unit testing. I can easily view a variable's value in the console when a test runs, saving me from debugging manually.

[This](#) may help if you're using XUnit.

edited Aug 30 at 17:04

answered Apr 30 at 0:59



DharmaTurtle

984 1 12 27

A variation on Alexey's answer. Slightly more complicated but doesn't involve writing to a text file:

0

1) In the Immediate Window enter:

```
System.IO.StringWriter stringWriter = new System.IO.StringWriter();
```

2) In the Watch Window enter two watches:

a. `stringWriter`

b. `new System.Xml.Serialization.XmlSerializer(obj.GetType()).Serialize(stringWriter, obj)`

After you enter the second watch (the Serialize one) the `stringWriter` watch value will be set to `obj` serialized to XML. Copy and paste it. Note that the XML will be enclosed in curly braces, `{...}`, so you'll need to remove them if you want to use the XML for anything.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



0



Use this in Visual Studio's "Immediate" window, replacing `c:\directory\file.json` with the full path to the file to which you'd like to write the JSON and `myObject` with your variable to serialise:

```
System.IO.File.WriteAllText(@"c:\directory\file.json",  
Newtonsoft.Json.JsonConvert.SerializeObject(myObject))
```

answered Aug 30 at 12:42

