Using .Net what limitations (if any) are there in using the XmlSerializer?

Asked 16 years, 3 months ago Modified 16 years ago Viewed 7k times

8

Using .Net what limitations (if any) are there in using the XmlSerializer? For example, can you serialize Images to XML?

.net xml serialization

Share

Improve this question

Follow

edited Nov 26, 2008 at 9:52

skolima

32.7k • 27 • 118 • 152

asked Sep 20, 2008 at 20:45

Gary Willoughby

52.5k • 42 • 139 • 204

Re-tag please. This is not a C#-specific question. Tag with ".net" instead. – Alex Lyman Sep 20, 2008 at 21:33

8 Answers

Sorted by: Highest score (default)

I generally find the XmlSerializer to be a poor choice for any POCO that's more than just a DTO. If you require specific XML, you can go the Xml*Attribute and/or

23

IXmlSerializable route - but you're left with an object pretty mangled.

For some purposes, it still an obvious choice - even with it's limitations. But, for simply storing and reloading data, I've found BinaryFormatter to be a much easier choice with less pitfalls.

Here's a list of some annoyances with XmlSerializer - most I've been bitten by at one point or another, others I found over at MSDN:

- Requires a public, no args constructor
- Only serializes public read/write properties and fields
- Requires all types to be known
- Actually calls into get_* and set_*, so validation, etc. will be run. This may be good or bad (think about the order of the calls as well)
- Will only serialize IEnumerable or ICollection collections conforming to specific rules

> The XmlSerializer gives special treatment to classes that implement IEnumerable or ICollection. A class that implements IEnumerable must implement a public Add method that takes a single parameter. The Add method's parameter must be of the same type as is returned from the Current property on the value returned from GetEnumerator, or one of that type's bases.

> A class that implements ICollection (such as CollectionBase) in addition to IEnumerable must have a public Item indexed property (indexer in C#) that takes an integer, and it must have a public Count property of type integer. The parameter to the Add method must be the same type as is returned from the Item property, or one of that type's bases. For classes that implement ICollection, values to be serialized are retrieved from the indexed Item property, not by calling GetEnumerator.

- Does not serialize IDictionary
- Uses dynamically generated assemblies, which may not get unloaded from the app domain.

> To increase performance, the XML serialization infrastructure dynamically generates assemblies to serialize and deserialize specified types. The infrastructure finds and reuses those assemblies. This behavior occurs only when using the following constructors:
>
> XmlSerializer.XmlSerializer(Type)
> XmlSerializer.XmlSerializer(Type, String)
>
> If you use any of the other constructors, multiple versions of the same assembly are generated and never unloaded, which results in a memory leak and poor performance.

- Cannot serialize ArrayList[] or List<T>[]
- Has other weird edge cases

> The XmlSerializer cannot be instantiated to serialize an enumeration if the following conditions are true: The enumeration is of type unsigned long (ulong in C#) and the enumeration contains any member with a value larger than 9,223,372,036,854,775,807.
>
> The XmlSerializer class no longer serializes objects that are marked as [Obsolete].
>
> You must have permission to write to the temporary directory (as defined by the TEMP environment variable) to deserialize an object.

- Requires reading .InnerException to get any useful info on errors

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Sep 21, 2008 at 0:26

Mark Brackett

85.6k ● 17 ● 111 ● 155

1 Jeez, that's a lot! of limitations! Thanks for the detail. +1 vote – Phil Jun 9, 2010 at 4:51

XmlSerializer expects a no parameter constructor but it does not need to be public. Internal is good enough. Unfortunately BinaryFormatter.Serialize and .Deserialize are now deprecated and considered to be security risks. This is very unfortunate. – H2ONaCl Feb 25, 2021 at 4:08

What do you mean by saying... "requires all types to be known" ? Please advise. – H2ONaCl Feb 25, 2021 at 4:17

19

The XmlSerializer has a few drawbacks.

1. It must know all the types being serialized. You cannot pass it something by interface that represents a type that the serializer does not know.
2. It cannot do circular references.
3. It will serializes the same object multiple times if referenced multiple times in the object graph.
4. Cannot handle private field serialization.

I (stupidly) wrote my own serializer to get around some of these problems. Don't do that; it is a lot of work and you will find subtle bugs in it months down the road. The only thing I gained in writing my own serializer and formatter was a greater appreciation of the minutia involved in object graph serialization.

I found the NetDataContractSerializer when WCF came out. It does all the stuff from above that XmlSerializer doesn't do. It drives the serialization in a similar fashion to the XmlSerializer. One decorates various properties or fields with attributes to inform the serializer what to

serialize. I replaced the custom serializer I had written with the NetDataContractSerializer and was very happy with the results. I would highly recommend it.

Share Improve this answer

Follow

answered Sep 20, 2008 at 21:07

Jason Jackson

17.2k ● 8 ● 51 ● 75

I think, with 3.5SP1 you don't even need to decorate with DataContract/DataMember anymore. – Christian.K Nov 26, 2008 at 11:52

+1: NetDataContractSerializer - that's a new one to me. Thanks! – x0n Dec 21, 2011 at 4:19

3

Another problem is that calling the constructor of XmlSerializer will compile code at runtime and will generate a temp DLL (in the %temp% folder) with the code to do the de/serialization.

You can watch the code if you add the following lines to app.config:

```
<system.diagnostics>
  <switches>
    <add name="XmlSerialization.Compilation" value="
  </switches>
</system.diagnostics>
```

This takes a lot of time the first time you serialize a class and needs code with permissions for compiling and writing to disk.

A way to get around that is to precompile these DLL using the sGen.exe tool that comes with VS 2005+.

Look here for more information.

Share Improve this answer

Follow

answered Nov 27, 2008 at 11:34

Tomer Pintel

1,587 ● 1 ● 14 ● 15

2

Not sure if there's any limitation.. But there was a memory leak bug in XmlSerialization in .NET 1.1, you sort of had to create a cache serializer object to get around with this issue... In fact, Im not sure if this issue has been fixed in .net 2.0 or newer...

Share Improve this answer

Follow

answered Sep 20, 2008 at 20:51

RWendi

1,422 ● 5 ● 21 ● 38

> Why did someone downvote this ? The windows service I wrote has been plagged by this XMLSerializer memory leak, and it is good to spread the word IMHO. – Larry Sep 18, 2012 at 19:15

1

Any class you write can theoretically be fed through XmlSerializer. Howerver, it only has access to the public fields, and the classes need to be marked with the correct attributes (e.g. XmlAttribute). Even in the basic framework, not everything supports XmlSerializer. System.Collections.Generic.Dictionary<> for instance.

Share Improve this answer

Follow

answered Sep 20, 2008 at 20:55

Scott Pedersen

1,311 ● 1 ● 10 ● 19

1

The one limitation that I can think of is that XmlSerialization is opt-out; meaning any properties of a class that you don't want serialized MUST be decorated with [XmlIgnore]. Contrast that to DataContractSerializer where all properties are opt-in, you must explicitly declare inclusion attributes. Here's a good write-up.

Images or their binary arrays are serialized as base64 encoded text by XmlSerializer.

Share Improve this answer

Follow

edited Sep 20, 2008 at 20:56

answered Sep 20, 2008 at 20:51

Kris

6,068 ● 2 ● 21 ● 16

1

For example, you can't serialize classes implementing IDictionary interface.

Share Improve this answer

Follow

answered Sep 20, 2008 at 21:17

starec

121 ● 2 ● 7

0

For collections they need to have an Add method taking a single argument. If you just need a text format and not specifically xml you might try JSON. I've developed one for .NET, JsonExSerializer, and there are others available as well at http://www.json.org.

Share Improve this answer

Follow

answered Sep 21, 2008 at 0:17

Ted Elliott

3,483 ● 1 ● 29 ● 33