

Re: serialization without preregistering subclasses

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-10/msg01661.html>

- *From:* "Nicholas Paldino [.NET/C# MVP]" <mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 11 Oct 2007 21:27:44 -0400
-

Why not just use the BinaryFormatter class? Serialize it to a MemoryStream and then return the byte array which the object is serialized to. The serialization engine will handle all of these issues for you.

--
- Nicholas Paldino [.NET/C# MVP]
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Me" <noone@xxxxxxx> wrote in message news:pan.2007.10.12.00.05.10@xxxxxxxxxxx

Hi.

Object serialization is a complicated issue that has few (good) solutions. In languages such as C++ and C# the programmer needs a way to unserialize an object to the type it was originally created. This usually involves some (often convoluted) lookup method that requires all the subclasses to be pre-registered in a lookup table or a static switch/case structure. In my native language (C++) there is no elegant way to do this.

In C# we have reflection, and my question is how I can use reflection information to rebuild a subclass object that has been previously serialized. I prefer to do binary serialization where the serialized object's binary representation starts with a (class name) key to tell what type of class it originally was.

Consider the following:

```
public class message {  
    virtual byte [] serialize() {}  
    static message unserialize(byte[] a) {}  
}  
  
public class datamessage: message {}  
  
public class controlmessage: message {}  
  
public class adminmessage: message {}
```

Re: serialization without preregistering subclasses

How can I use reflection information to create an object of the correct subclass for binary serialization of a message that starts with the correct class name as a key to indicate what class the serialized data belongs to?

Please don't go off on tangents about problems with subclass attributes that are references to other objects because I intend to do deep copies of all subclass attribute objects so that point is moot.

Any public how-tos or FAQ documents that I should view to find out more about this?

Thanks