Date: Vendredi 28 septembre 2012 à 22:30:09
Sujet: Programmation C# .NET

C# Copier un objet sans implémenter ICloneable

Pour cloner un objet en C#, il est recommandé d'implémenter l'interface ICloneable. On peut ensuite appeler la méthode Clone() sur l'objet implémentant la classe ICloneable.

Cependant il n'est pas toujours possible d'implémenter l'interface ICloneable si la classe de l'objet à copier ne nous appartient pas. La plupart des bibliothèques externes proposent des méthodes Clone quand cela est nécessaire mais ce n'est pas toujours le cas.

Lorsque vous n'avez pas accès à la classe de l'objet à copier, il reste une solution, copier l'objet en utilisant la serialization binaire.
Ce type de clonage fait une copie bit par bit de l'objet cible. Les références pointeront donc vers les mêmes objets que l'objet initial et seuls les types valeurs seront réellement clonés.
Ce type de copie ne peut donc pas être utilisé dans tous les cas et il faut bien comprendre les impacts avant de l'utiliser.

Voici le code d'un serializer permettant d'effectuer une copie binaire d'un objet même s'il ne possède pas de fonction Clone.

Pour l'utiliser, il vous suffit d'appeler ObjectCloner.Clone(objetACopier).

Exemple:
MyClass copie = ObjectCloner.Clone(original);

```
 using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;

public static class ObjectCloner
{
    /// &lt;summary&gt;
    /// Perform a deep Copy of the object.
    /// &lt;/summary&gt;
    /// &lt;typeparam name=&quot;T&quot;&gt;The type of object being copied.&lt;/typeparam&gt;
    /// &lt;param name=&quot;source&quot;&gt;The object instance to copy.&lt;/param&gt;
    /// &lt;returns&gt;The copied object.&lt;/returns&gt;
    public static T Clone&lt;T&gt;(T source)
    {
        if (!typeof(T).IsSerializable)
        {
```

            throw new ArgumentException ("The type must be serializable.", "source");
        }

        // Don't serialize a null object, simply return the default for that object
        if (Object.ReferenceEquals(source, null))
        {
            return default(T);
        }

        IFormatter formatter = new BinaryFormatter();
        Stream stream = new MemoryStream();
        using (stream)
        {
            formatter.Serialize(stream, source);
            stream.Seek(0, SeekOrigin. Begin);
            return (T)formatter.Deserialize (stream);
        }
    }
}