

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.clr/2006-09/msg00044.html>

- *From:* Stuart Carnie <stuart.carnie@xxxxxxxxxxxxxx>
 - *Date:* Tue, 19 Sep 2006 13:19:05 -0700
-

Sure thing, your particular example is certainly better :). What I presented was just an example. I'm sure you are aware that LCG is significantly more powerful than that.

How about a generic 'copy public read/write properties' delegate. The delegate is strongly typed, and you can generate one for any object that has public, read/write properties.

Runtime version = 2.0.50727.42

Iterations = 1,000,000

00:00:00.0625004

FirstName: John, LastName: Doe, Age: 64

Cheers,

Stu

```
/* ***** COPY BELOW ***** */
```

```
using System;
using System.Reflection;
using System.Reflection.Emit;
```

```
public class Person
{
    private string _firstName;
    private string _lastName;
    private int _age;
```

```
    public string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
```

```
    public string LastName
    {
        get { return _lastName; }
        set { _lastName = value; }
    }
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
}
```

```
public int Age  
{  
    get { return _age; }  
    set { _age = value; }  
}
```

```
delegate void CopyPublicPropertiesDelegate<T>(T source, T dest);
```

```
class Program  
{  
    static void Main(string[] args)  
    {  
        Console.WriteLine("Runtime version = {0}", Environment.Version);
```

```
        int iterations = args.Length > 0 ? int.Parse(args[0]) : 1000000;  
        Console.WriteLine("Iterations = {0}", iterations.ToString("N0"));
```

```
        Person srcperson = new Person();  
        srcperson.FirstName = "John";  
        srcperson.LastName = "Doe";  
        srcperson.Age = 64;
```

```
        Person dstperson = new Person();
```

```
        DateTime start = DateTime.Now;
```

```
        CopyPublicPropertiesDelegate<Person> cd = GenerateCopyDelegate<Person>();
```

```
        for (int i = 0; i < iterations; i++)  
        {  
            cd(srcperson, dstperson);  
        }
```

```
        Console.WriteLine(DateTime.Now - start);
```

```
        Console.WriteLine("FirstName: {0}, LastName: {1}, Age: {2}", dstperson.FirstName, dstperson.LastName,  
        dstperson.Age);  
    }
```

```
    static CopyPublicPropertiesDelegate<T> GenerateCopyDelegate<T>()  
    {  
        Type type = typeof(T);
```

```
        DynamicMethod dm = new DynamicMethod(type.Name, null, new Type[] { typeof(T), typeof(T) },  
        typeof(Program).Module);
```

```
        ILGenerator il = dm.GetILGenerator();
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
PropertyInfo[] props = type.GetProperties(BindingFlags.Public | BindingFlags.Instance);

foreach (PropertyInfo prop in props)
{
    if (prop.CanRead && prop.CanWrite)
    {
        il.Emit(OpCodes.Ldarg_1);
        il.Emit(OpCodes.Ldarg_0);
        il.EmitCall(OpCodes.Callvirt, prop.GetGetMethod(), null);
        il.EmitCall(OpCodes.Callvirt, prop.GetSetMethod(), null);
    }
}
il.Emit(OpCodes.Ret);

return (CopyPublicPropertiesDelegate<T>)dm.CreateDelegate(typeof(CopyPublicPropertiesDelegate<T>));
}

}
```

Atif Aziz wrote:

Hi Stuart,

I'm well aware of LCG, and in fact, I'm not even sure why you need it in 2.0. You can get the same results without LCG using purely delegates. Define the delegate like this:

```
delegate void ValueSetter<TContainer, TValue>(TContainer container, TValue value);
```

Then create it like this:

```
ValueSetter<Person, string> fnSetter = (ValueSetter<Person, string>)
Delegate.CreateDelegate(stringSetterType, fn.GetSetMethod());
```

And use it as you did:

```
fnSetter(person, "John");
```

Less code and doesn't require LCG or IL magic and is even strongly typed on the container type (Person). Incidentally, this version performs 145 times faster than using reflection.

Anyhow I'm interested in understanding where the performance issues are occurring. Generally, things should get faster with each version of the framework, but going twice as slow as previous version mandates at least a warning or notice in the documentation. Don't you think? The problem is that we can't upgrade all applications to 2.0 overnight. Moreover, loading 1.x assemblies is 2.0 apps will upgrade automatically at runtime but suffer twice the performance if they're internally using reflection.

BTW, have you tried making Person private? The performance slows down by a factor of 4 times than when its public. Here the behavior is fairly identical across 1.x and 2.0. If it's a CAS issue, then I can't imagine why I'm being more taxed for accessing private types more

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

than public ones! It's also quite a tax for a fully trusted application.

"Stuart Carnie" wrote:

I'd suggest you take advantage of Lightweight Code Generation (LCG).

I knocked a v2.0 sample together using your example.

NO LCG:

Runtime version = 2.0.50727.42

Iterations = 1,000,000

00:00:04.8898738

USING LCG:

Runtime version = 2.0.50727.42

Iterations = 1,000,000

00:00:00.0937632

----- COPY BELOW -----

```
#define LCG
```

```
// remove this define to use the old PropertyInfo way.
```

```
using System;  
using System.Reflection;  
using System.Reflection.Emit;
```

```
public class Person  
{  
    private string _firstName;  
    private string _lastName;  
    private int _age;
```

```
    public string FirstName  
    {  
        get { return _firstName; }  
        set { _firstName = value; }  
    }
```

```
    public string LastName  
    {  
        get { return _lastName; }  
        set { _lastName = value; }  
    }
```

```
    public int Age  
    {  
        get { return _age; }  
        set { _age = value; }  
    }
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Runtime version = {0}", Environment.Version);

        int iterations = args.Length > 0 ? int.Parse(args[0]) : 1000000;
        Console.WriteLine("Iterations = {0}", iterations.ToString("N0"));

        Person person = new Person();
        person.FirstName = "Hello";

        PropertyInfo fn = typeof(Person).GetProperty("FirstName");
        PropertyInfo ln = typeof(Person).GetProperty("LastName");
        PropertyInfo age = typeof(Person).GetProperty("Age");

        DateTime start = DateTime.Now;

        #if LCG
        FastSetterDelegate<string> fns = GenerateFastSetter<string>(fn);
        FastSetterDelegate<string> lns = GenerateFastSetter<string>(ln);
        FastSetterDelegate<int> ages = GenerateFastSetter<int>(age);

        for (int i = 0; i < iterations; i++)
        {
            fns(person, "John");
            lns(person, "Doe");
            ages(person, 42);
        }
        #else
        for (int i = 0; i < iterations; i++)
        {
            fn.SetValue(person, "John", null);
            ln.SetValue(person, "Doe", null);
            age.SetValue(person, 42, null);
        }
        #endif
        Console.WriteLine(DateTime.Now - start);
    }

    static FastSetterDelegate<T> GenerateFastSetter<T>(PropertyInfo pi)
    {
        DynamicMethod dm = new DynamicMethod(pi.Name, null, new Type[] {
            typeof(object), typeof(T) }, typeof(Program).Module);

        ILGenerator il = dm.GetILGenerator();
        il.Emit(OpCodes.Ldarg_0);
        il.Emit(OpCodes.Ldarg_1);
    }
}
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
il.EmitCall(OpCodes.Callvirt, pi.GetSetMethod(), null);
il.Emit(OpCodes.Ret);

return
(FastSetterDelegate<T>)dm.CreateDelegate(typeof(FastSetterDelegate<T>));
}

delegate void FastSetterDelegate<T>(object obj, T value);
}
```

Eric wrote:

I have here a little sample which does reflection on a class.
I struck me that 1.1 is nearly two times faster.
Could me somebody give a hint why?

For the sample: Compile it via "csc.exe /optimize+
Program.cs" with the two compiler for 1.1 and 2.0.

Thanks
Eric

```
namespace ComponentProfiling
{
    using System;
    using System.ComponentModel;
    using System.Reflection;

    public sealed class Person
    {
        private string _firstName;
        private string _lastName;
        private int _age;

        public string FirstName
        {
            get { return _firstName; }
            set { _firstName = value; }
        }

        public string LastName
        {
            get { return _lastName; }
            set { _lastName = value; }
        }

        public int Age
        {
            get { return _age; }
            set { _age = value; }
        }
    }
}
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
}
}
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Runtime version = {0}",
Environment.Version);

int iterations = args.Length > 0 ? int.Parse(args[0]) :
1000000;
Console.WriteLine("Iterations = {0}",
iterations.ToString("N0"));

Person person = new Person();

PropertyDescriptorCollection properties =
TypeDescriptor.GetProperties(typeof(Person));
string test1 = typeof(TypeDescriptor).Name;
TimeSpan span1 = Measure(iterations, person,
properties["FirstName"],
properties["LastName"],
properties["Age"]);

Console.WriteLine("{0} = {1}", test1, span1);
string test2 = typeof(MyPropertyDescriptor).Name;
TimeSpan span2 = Measure(iterations, person,
new
MyPropertyDescriptor(typeof(Person).GetProperty("FirstName")),
new
MyPropertyDescriptor(typeof(Person).GetProperty("LastName")),
new
MyPropertyDescriptor(typeof(Person).GetProperty("Age")));

Console.WriteLine("{0} = {1}", test2, span2);
Console.WriteLine("{0}/{1} = {2:P}", test1, test2,
span1.TotalMilliseconds / span2.TotalMilliseconds);

Console.WriteLine("Press ENTER to end.");
Console.ReadLine();
}

static TimeSpan Measure(int iterations, Person person,
params PropertyDescriptor[] properties)
{
DateTime start;
start = DateTime.Now;
for (int i = 0; i < iterations; i++)
{
properties[0].SetValue(person, "John");
properties[1].SetValue(person, "Doe");
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
properties[2].SetValue(person, 22); }
return DateTime.Now – start;
}

public sealed class MyPropertyDescriptor :
PropertyDescriptor
{
private readonly PropertyInfo _property;

public MyPropertyDescriptor(PropertyInfo property)
:
base(property.Name, null)
{
_property = property;
}

public override bool CanResetValue(object component)
{
return false;
}

public override object GetValue(object component)
{
return _property.GetValue(component, null);
}

public override void ResetValue(object component)
{
throw new NotSupportedException();
}

public override void SetValue(object component, object
value)
{
_property.SetValue(component, value, null);
}

public override bool ShouldSerializeValue(object
component)
{
return false;
}

public override Type ComponentType
{
get { return _property.ReflectedType; }
}

public override bool IsReadOnly
{
get { return !_property.CanWrite; }
}
```

Re: Reflection in 1.1 and 2.0 – generic, fast copy delegate?

```
}  
  
public override Type PropertyType  
{  
    get { return _property.PropertyType; }  
}  
}  
}  
}
```