

Re: dynamic types

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-10/5308.html>

From: Stephen Lamb (*slamb_at_easystreet.com*)

Date: 10/24/04

Date: Sat, 23 Oct 2004 20:17:03 -0700

"Stephen Lamb" <slamb@easystreet.com> wrote in message
news:10nka82r2548ked@corp.supernews.com...

> *How would one do the following at runtime? I'm really interested in steps*

2

> *and 3.*

>

> *1. Read data from somewhere that describes types and instances.*

> *2. Construct new types from data.*

> *3. Construct instances of types from data.*

> *3. Use new types and instances in a type safe manner.*

>

> *e.g.*

>

> *1. Read data from somewhere that describes types and instances.*

> *Create a type that is a struct called Foo with members E1 and E2 that are*

32

> *bit ints.*

> *Construct a Foo with E1 = 23 and E2 = 5.*

>

> *2. Construct new types from data.*

> *public struct Foo*

> {

> *public Int32 E1;*

> *public Int32 E2;*

> }

>

> *3. Construct instances of types from data.*

> *Foo foo = new Foo();*

> *foo.E1 = 23;*

> *foo.E2 = 5;*

>

> *4. Use new types and instances in a type safe manner.*

> *foo.E1 = 23;*

>

Thanks to all for the pointers. I had only taken a very quick look at

CodeDom so any pointers on how to use it is helpful.

As Stoitcho Goutsev pointed out, my idea isn't really possible in C# since this is a compiled language. I'm guessing it might be possible in a really dynamic system like Smalltalk.

I guess my next best option would be to automatically generate the types ahead of time using CodeDom and then compile them into the assembly that will read the construction data and instantiate instances. The code that does the reading of type data can compare the type compiled into the assembly to the type specified by the read data. I'll probably just use versioning to compare the two type descriptions and then only do the real check in debug.

Thanks,
Steve