

## Re: Load an object reference onto the stack ???

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2006-04/msg00806.html>

---

- *From:* Barry Kelly <[barry.j.kelly@xxxxxxxxxx](mailto:barry.j.kelly@xxxxxxxxxx)>
  - *Date:* Fri, 28 Apr 2006 00:28:36 +0100
- 

"VivekR" <[KenBase@xxxxxxxxxx](mailto:KenBase@xxxxxxxxxx)> wrote:

- \* In your example, does the runtime IL code call an instance method of class App ?

The IL I generated calls a passed-in delegate. Delegates are objects which have a method called "Invoke". When you call a delegate in C#, the compiler compiles it as a call to the instance method "Invoke" on the delegate instance.

So, in order for the IL to call this Invoke method in my example, it needs to do three things:

- \* Load the object reference (i.e. the delegate) onto the stack. This will be the 'this' reference for the method call later on.
- \* Load the arguments to the method call. In the case of my example, the Invoke method's arguments are the same as the delegate's method arguments, that is to say a single argument of type string.
- \* Actually invoke the (non-virtual) instance method "Invoke".

So, my code calls an instance method called "Invoke" on the object reference "SomeMethod reference" passed as a parameter to the CompileMethod method.

- \* And I do not exactly understand why you are using List<object>, which will contain multiple objects in it.

The reason I used List<object> is because it grows automatically, and it can be converted easily to an object[] with the method ToArray(). The reason I use object[] as the 'this' pointer associated with the created delegate (passed to DynamicMethod.CreateDelegate) is because it is a general solution (i.e. it will work in all cases) to the problem: "How to load an object reference onto the stack?", for any number of object references of any type.

Re: Load an object reference onto the stack ???

\* Why are you comparing

?

\* In the statements

```
cg.Emit(OpCodes.Ldarg_0);  
cg.Emit(OpCodes.Ldc_I4, index);  
cg.Emit(OpCodes.
```