

## Re: Reflection – Need advice

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-11/msg01380.html>

---

- *From:* "Peter Morris" <[mrpmorrisNO@xxxxxxxxxxxxxxx](mailto:mrpmorrisNO@xxxxxxxxxxxxxxx)>
  - *Date:* Tue, 18 Nov 2008 08:19:16 -0000
- 

Hi Cralis

I kept quiet and smiled.

I always ask, no matter how stupid I suspect I might look as a consequence. It's a good way to learn new stuff, actually increases your confidence in your ability, and quickly shows up the other guy as a fraud if he doesn't know the subject well enough to explain it properly :-)

What is reflection!?

```
public class Person
{
    private string name;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }

    private void DoSomething(string x)
    {
    }
}
```

Using reflection on the assembly you can get a list of Type. Using that Type you can get a list of constructors, methods, properties, method's parameters etc. You can then use those references to do stuff like invoke the methods and so on.

I used reflection recently when I had to send a text-based command to objects implementing ISignalTarget.

```
public interface ISignalTarget
{
```

## Re: Reflection – Need advice

```
void AcceptSignal(ISignal signal);  
}
```

I created a base class which automatically dispatched the signal to an appropriately named method, so if signal.ID was "Copy" I would look for

```
(any exposure) void Accept_CopySignal(CopySignal signal)  
{  
}
```

and if found I would invoke it.

When you add .NET attributes to your code I believe reflection is used to discover those.

In summary, reflection is possible because .NET understands how to read the contents of the assembly. As a consequence you are able to read those contents yourself, and also execute methods etc too (MethodInfo.Invoke).

One last example. If you have assemblies in a "Plugins" folder of your app that you want loaded dynamically, maybe because other people are allowed to write their own (like you can with Photoshop). You could load each of the assemblies in the Plugins folder, then get all types in the assembly that implement IMyPlugin. If that type is also decorated with your own PluginAttribute you could

- 01: Read PluginAttribute.DisplayName and MenuLocation to display the plugin in the menu.
- 02: When the user clicks the menu you can create an instance of the plugin and execute it.

The new ASP MVC framework uses reflection to take a url like this

<http://localhost/Employee/View/1>

and execute a method like this

```
public class EmployeeController : .....  
{  
    public ActionResult View(int id) { ..... }  
}
```

I tried to think of as many examples as possible in the hope you could identify with one :-)

Pete

====

<http://mrpmorris.blogspot.com>

<http://www.capableobjects.com>

.