

Re: Design Question

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-12/5200.html>

From: Nicole Calinoiu (*calinoiu*)

Date: 12/23/04

Date: Thu, 23 Dec 2004 09:32:44 -0500

Not sure why, but I didn't even consider that this might have anything to do with the principal when I answered yesterday. That said, even if it is reasonably simple to swap out the principal for a web service, it's still not necessarily a great idea to do so. While using Windows-integrated authentication for web services is certainly simple, investing additional effort into tweaking the behaviour of a non-standard authentication mechanism probably isn't the best use of most folks' time...

"Nicholas Paldino [.NET/C# MVP]" <mvp@spam.guard.caspershouse.com> wrote in message news:Or%230XtG6EHA.4004@tk2msftngp13.phx.gbl...

> *Or (and I think this is the easiest one of all), just use a custom
> principal, and let Code Access Security take care of the rest. Basically,
> you implement IPrincipal, and set it as the principal for the thread that
> is doing the processing.*

>
> *Attached is a console application which demonstrates how to use the
> PrincipalPermission attribute. Basically, there is an implementation of
> IPrincipal and the current thread is set to use that principal (you will
> have to do something different to have web requests use a certain
> principal, but Im sure you can do it). Then, you just apply the right
> attribute to your method, and the runtime will take care of the rest.*

>
> *Try changing the IsInRole implementation to return something else, or
> the declaration of the PrincipalPermission attribute and the call to
> DoSomething will fail.*

>
> *Hope this helps.*

>
>
>
> --

> - Nicholas Paldino [.NET/C# MVP]
> - mvp@spam.guard.caspershouse.com

>
>
> *"Nicole Calinoiu" <calinoiu REMOVETHIS AT gmail DOT com> wrote in message
> news:O5xBXYG6EHA.1260@TK2MSFTNGP12.phx.gbl...
>> Yes, but it's probably not as simple as you might have hoped. Here are*

>> *the*
>> *three main approaches:*
>>
>> *1. Implement the check as a custom permission with a corresponding*
>> *attribute*
>>
>> (<http://msdn.microsoft.com/library/en-us/cpguide/html/cpconcreatingyourowncodeaccesspermissions.asp>).
>> *This may be your best bet since you can presumably control whether the*
>> *attribute assembly is registered as a trusted assembly.*
>>
>> *2. Place the actual method work in objects that inherit from*
>> *System.ContextBoundObject. This might interfere with your planned object*
>> *hierarchy, as well as introducing an otherwise unnecessary performance*
>> *hit.*
>>
>> *3. Use a tool like XC# (<http://www.resolvecorp.com/Products.aspx>) to*
>> *generate inline code that corresponds to your custom attribute.*
>>
>> *If this truly is a security permission, #1 is probably the "cleanest"*
>> *approach. Otherwise, #3 would probably offer the best compromise between*
>> *design-time convenience and runtime performance.*
>>
>> *HTH,*
>> *Nicole*
>>
>>
>>
>> *"John Lee" <johnl@newsgroup.nospam> wrote in message*
>> *news:ejm4%237F6EHA.2180@TK2MSFTNGP10.phx.gbl...*
>>> *Hi,*
>>>
>>> *If I want to check permission on each public method of a web service,*
>>> *(assume the checking routine is ready to use and called AccessCheck),*
>>> *one*
>>> *way of doing it is to call this AccessCheck on top of each public*
>>> *method,*
>>> *I want to implement it in different way but seems missing something –*
>>>
>>> *I want to develop a custom attribute, let's say*
>>> *SecurityCheckEnabledAttribute with only Yes/No parameter, then create a*
>>> *base class for all web service classes, Is there any way to capture the*
>>> *public method call from base class at runtime and then check if the*
>>> *attribute is being applied and then check the permission?*
>>>
>>> *Thanks a lot!*
>>>
>>> *Regards,*
>>> *John*
>>>
>>
>>
>>

microsoft.public.dotnet.languages.csharp: Re: Design Question

>
>
>