

Re: Article: Permissions View Tool (Permview.exe .Net FrameWork Tools Series)

Source: <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2004-11/0107.html>

From: Alvin Bruney [MVP] (*vapor*)

Date: 11/02/04

Date: Mon, 1 Nov 2004 22:11:24 -0600

very nice, very thorough as well.

--

Regards,

Alvin Bruney

[ASP.NET MVP <http://mvp.support.microsoft.com/default.aspx>]

Got tidbits? Get it here... <http://tinyurl.com/27cok>

"Namratha Shah (Nasha)" <namrathal@gmail.com> wrote in message
news:e76740DwEHA.356@TK2MSFTNGP10.phx.gbl...

> Hi All,

>

> Permissions View Tool is shipped with .Net FrameWork. This tool is used to
> view the Declarative and Requested permissions set on an assembly.

>

> So let us create an assembly set some requested and declarative
> permissions

> on it and view them with Permview.

>

> Let us start by creating an assembly and setting some Requested
> permissions

> on it.

>

> 1) Open a new WindowsApplication.

> 2) Add a button to the form named Button1.

> 3) Add a Class to this Application Named MyClass.

> 4) Add a method to this class named Add which adds to numbers and returns
> the result.

> 5) In the Button Click event of Button1 call the Add method of the MyClass
> as shown below.

>

> MyClass myClass = new MyClass();

> MessageBox.Show(myClass.Add(1,2).ToString());

>

> 6) Run the application. click on the Button and you will get the result as
> "3".Now we will add requested permission to this assembly.

> 7) Go to AssemblyInfo.cs and add the below lines

>

> [assembly: FileIOPermission(SecurityAction.RequestMinimum,Write="c:\\")]

> [assembly: UIPermission(SecurityAction.RequestRefuse, Window =

> UIPermissionWindow.AllWindows)]

>

> By setting these permissions we say that to execute this application

> minimum

> write permission on C: is required. Along with that we have also requested

```
> an another UI permission which refuses to show any windows on the screen
> hence no forms or message boxes will be displayed.
>
> 8) Run the application. The form will not be loaded instead you will see
> the
> below error message:
>
> Additional information: Request for the permission of type
> System.Security.Permissions.UIPermission, mscorlib, Version=1.0.3300.0,
> Culture=neutral, PublicKeyToken=b77a5c561934e089 failed.
>
> 9) We can view the permissions set on an assembly with permview.exe.
> 10) Go to VS.Net command prompt and go to the folder which contains your
> application exe.
> 11) Type the following command :permview WindowsApplication2.exe <or your
> application name>
> 12) You will see all the permissions set on the assembly as below:
>
> minimal permission set:
> <PermissionSet class="System.Security.PermissionSet"
>     version="1">
>   <IPermission class="System.Security.Permissions.FileIOPermission,
> mscorlib, Version=1.0.3300.0, Culture=neutral,
> PublicKeyToken=b77a5c561934e089" version="1"     Write="c:\"/>
> </PermissionSet>
> optional permission set:
> Not specified
> refused permission set:
> <PermissionSet class="System.Security.PermissionSet"
>     version="1">
>   <IPermission class="System.Security.Permissions.UIPermission, mscorlib,
> Versi
> on=1.0.3300.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
>     version="1"
>     Window="AllWindows"/>
> </PermissionSet>
>
> 13) Since we have set RequestRefuse permission for UIPermission we were
> unable to see the form so let us change that to RequestOptional as follows
> :
>
> [assembly: UIPermission(SecurityAction.RequestOptional, Window =
> UIPermissionWindow.AllWindows)]
>
>
> These permissions on the assembly that we saw were requested permissions
> on
> the assembly. Permview is also used to view declarative security on
> classes
> or methods.
>
> 15) Let us add Declarative security to our MyClass as follows:
>
> [PrincipalPermission (SecurityAction.Demand, Role="Administrator")] // add
> this above the class declaration
>
> 16) According to the above permission only those users who belong to the
> "Administrator" role will be able to instantiate this class or call any
> method on this class.
> 17) Let us view the declarative security of this class.
> 18) Go to the VS.NET command prompt and type the following command :
> permview /decl WindowsApplication2.exe.
```

```
>
> Note : - "/decl" shows the declarative permissions set on the classes and
> methods of the assembly.
>
> 19) You will see the class declarative permissions (as shown below) along
> with the requested assembly permissions.
>
> Class WindowsApplication2.Class1 NonCasDemand permission set:
> <PermissionSet class="System.Security.PermissionSet"
>     version="1">
>   <Permission class="System.Security.Permissions.PrincipalPermission,
> mscorlib,
> Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
>     version="1">
>     <Identity Authenticated="true"
>       ID="mycomputer\Administrator"/>
>   </Permission>
> </PermissionSet>
>
> 20) You will be able to execute the method call if the current user is an
> Administrator on the machine.
>
> In this way we can use Permview.exe to view the declarative and requested
> permissions set on any assembly.
>
> There are 3 types of requested permissions which can be granted on
> assembly
> :
>
> Minimum permissions: (RequestMinimum)  Permissions your code must have in
> order to run.
> Optional permissions: (RequestOptional)  Permissions your code can use,
> but can run effectively without.
> Refused permissions: (RequestRefuse)    Permissions that you want to
> ensure will never be granted to your code, even if security policy allows
> them to be granted.
>
>
> -- Please post your queries and comments for my articles in the usergroup
> for the benefit of all. I hope this step from my end is helpful to all of
> us.
> Regards,
>
> Namratha (Nasha)
>
>
>
```