

Re: How to use WindowsPrincipal properly??

Source:

<http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.framework.aspnet.security/2002-11/1668.html>

From: Ed leNoir (edleno@safeco.com)

Date: 11/01/02

From: "Ed leNoir" <edleno@safeco.com>

Date: Thu, 31 Oct 2002 18:45:23 -0800

No, WindowsBuiltInRole.Administrator is just an enumeration of the groups. Since you're using VB the ToString() is being done for you to convert it to string.

You're right, I mixed up the administrator account versus the group. The admin account is BUILTIN\Administrator, but the group has to have the machine name for it's domain name.

If you want to check if the user is in the local computers security group "Administrators", then do something like IsInRole(Dns.GetHostName() & "\Administrators") and it will NOT check against their own computer, just the machine the code is running on.

Sorry to have confused you by using the name "BUILTIN". That's just for users, not groups.

Normally integrated security is preferred by the users, since they aren't challenged with a userid and password to use the site. Of course, this is normally only used with internal sites to a corporation. For Internet sites used by the general public you have to use Basic Authentication of course.

Forms based authentication is also only used with the Internet, it's just that you can put up a pretty form instead of the basic signon provided in the browsers. And, more importantly, you can handle password changing right on the form and not make it a separate process.

The usual problem with aspnet is how to use the users credentials for validation, but NOT for database calls or network accesses. That's where LogonUser and impersonate comes into play. You can logon a set account that just has the minimum security to network and database resources to run the application. LogonUser returns a token which is then used to build a WindowsIdentity which is then used to Impersonate. You then use programmatic security – check in the aspx page if User.IsInRole("domainName\UpdateDatabaseGroup") before making database calls, then logon and impersonate your service account and make the update

database calls. Then perhaps revert back.

You need to do the impersonate if you are going to make calls to secured resources that are not on the current server. This is because a normal NTLM security token can't hop servers. With Win2k servers and workstations you can set up Kerberos and allow server hops, but most companies aren't doing that yet.

You don't have to validate the user at all normally. The o/s takes care of all that. Just put the users into a local or domain global group and do the IsInRole.

– Ed

"Kevin Yu" <kyu@nrcan.gc.ca> wrote in message news:usQR3NSgCHA.1636@tkmsftngp10...

> Ed

> thanks for the reply.

>

> in the IsInRole() function, I think the "BUILTIN\Administrators" and

> WindowsBuiltInRole.Administrator refer to

> different role, see we add users to administrators group, so if I am one of

> the administrators in the computer e.g. win2k

> machine, then the ("BUILTIN\Administrators") will return true while

> (WindowsBuiltInRole.Administrator) return false,

> I think this role is refer to the default admin account when the system is
> installed.

>

> I am basically confused with the role base authentication and the

> impersonation, not sure what exactly is the difference.

> now I have to check the machinename/domain name to validate the user, see
if

> they have an account on the domain, also

> have an account at the win2k machine in their office, that way, if they

type

> in office computer name instead of the domain

> computer name, since win2k dont have whatever socall PDC or BDC anymore,

> it's just a computer that contain the user

> list. so if I write code just to verify the user

> IsInRole("BUILTIN\Administrators") then it will return true if the user is

> an administrator

> in his/her computer in office, right?

>

> I can see that the LoginUser API call doesn take the user name and

password

> and validate the user again the specific "domain", but

> it need to get the user password, in windows integrated authentication,

how

> to get the password? do I need form authentication instead?

> but integrated authentication is more secure, correct?

microsoft.public.dotnet.framework.aspnet.security: Re: How to use WindowsPrincipal properly??

>

> Kevin

>

>

> "Ed leNoir" <EDLENO@safeco.com> wrote in message

> news:57f8df53.0210302108.58a1f76@posting.google.com...

>> Kevin,

>>

>>> The `IsInRole` is documented to require that you provide BOTH a domain

>>> name and user name in the format `domain\username`. The enumeration to

>>> string `ONLY` returns the string "Administrator", so you would have to

>>> write the code as `user.IsInRole("BUILTIN\" &`

>>> `WindowsBuiltInRole.Administrator)`.

>>>

>>> I'm not sure I understand your question about impersonation or

>>> `LogonUser`. For some strange reason dotnet allows you to make an

>>> impersonation call, but it's not easy to get the identity that you

>>> want to impersonate! So, the `LogonUser` API has to be used via a call

>>> to unmanaged code to get an impersonation token. You can then build a

>>> `WindowsIdentity` using that token, and from THAT you can do an

>>> impersonate.

>>>

>>> If you want to validate just the username and domain I think you can

>>> do a SID lookup to the domain controller using `LookupAccountName`. If

>>> you get a SID back then the name is valid in the domain (and the

>>> domain is valid also).

>>>

>>> The security API's are very confusing and are easily misused, and they

>>> don't report anything in the event log, so you REALLY have to be able

>>> to catch the error codes that are returned via `GetLastError`.

>>>

>>> – Ed

>>>

>>> "Kevin Yu" <kyu@nrcan.gc.ca> wrote in message

>>> news:<OBjTo72fCHA.1308@tkmsftngp11>...

>>>> I am working on this intranet app here need proper authentication for

>>>> users

>>>>> and redirect them according

>>>>> to their roles. I set app on IIS to use windows integrated

>>>>> authentication

>>>>> and in my code, I check when user

>>>>> login and get their identity, now I run into some minor problem, seems

>>>>> like

>>>>> the following statement return

>>>>> different result:

>>>>>

>>>>> `user.IsInRole(WindowsBuiltInRole.Administrator)` this return false

>>>>>

>>>>> and this

>>>>>

>>>>> `user.IsInRole("BUILTIN\Administrators")` this return true

Re: How to use WindowsPrincipal properly??

microsoft.public.dotnet.framework.aspnet.security: Re: How to use WindowsPrincipal properly??

> > >
> > > *for the same user? what is the difference?*
> > >
> > > *another question is how can I make sure user enter a proper domain in*
> *the*
> > > *popup login?*
> > > *say if the user dont enter the domain/computername that supposed to*
> > > *authenticate him/her,*
> > > *then I need to check domain in my code as well? since the*
> > > *user.Identity.Name will return*
> > > *DOMAIN\username, then in code need to parse the domain and username*
> *and*
> > > *validate both*
> > > *of them, I saw some other code that use Impersonation as the*
> *following:*
> > >
> > > *<DllImport("C:\\WINNT\\System32\\advapi32.dll")> _*
> > > *Public Shared Function LogonUser(lpszUsername As String, lpszDomain*
> *As*
> > > *String, lpszPassword As String, _*
> > > *dwLogonType As Integer, dwLogonProvider As Integer, ByRef*
> > > *phToken As Integer) As Boolean*
> > > *End Function*
> > >
> > > *seems like with Impersonation, there are lots more code needed. can*
> *anyone*
> > > *clarify what the differences are between*
> > > *the two?*
> > >
> > > *thanks*
>
>