**ASP.NET Impersonation**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/xh507fc5(d=printer,v=vs.90).ASPX)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/xh507fc5(d=printer,v=vs.85).ASPX)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/xh507fc5(d=printer,v=vs.80).ASPX)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/xh507fc5(d=printer,v=vs.71).ASPX)

When using impersonation, ASP.NET applications can execute with the Windows identity (user account) of the user making the request. Impersonation is commonly used in applications that rely on Microsoft Internet Information Services (IIS) to authenticate the user.

ASP.NET impersonation is disabled by default. If impersonation is enabled for an ASP.NET application, that application runs in the context of the identity whose access token IIS passes to ASP.NET. That token can be either an authenticated user token, such as a token for a logged-in Windows user, or the token that IIS provides for anonymous users (typically, the IUSR\_MACHINENAME identity).

When impersonation is enabled, only your application code runs under the context of the impersonated user. Applications are compiled and configuration information is loaded using the identity of the ASP.NET process. For more information, see [Configuring ASP.NET Process Identity](http://msdn.microsoft.com/en-us/library/dwc1xthy.ASPX). The compiled application is put in the Temporary ASP.NET files directory. The application identity that is being impersonated needs to have read/write access to this directory. The impersonated application identity also requires at least read access to the files in your application directory and subdirectories. For more information, see [ASP.NET Required Access Control Lists (ACLs)](http://msdn.microsoft.com/en-us/library/kwzs111e.ASPX).

|  |
| --- |
| **Description: NoteNote** |
| Because ASP.NET uses the Windows identity of the ASP.NET process when compiling applications and loading configuration information, you must keep application code and configuration information private between applications on a server that hosts multiple applications. On Windows Server 2003 you can create multiple application pools and specify a unique identity for each application pool. You can then restrict access to application files using access control lists (ACLs) (if file system is formatted using NTFS) and these identities. For example, consider two applications, App1 and App2, where the information in each application must be kept private. You can put App1 in the ApplicationPool1 application pool which has an identity of ID\_ApplicationPool1. You can put App2 in the ApplicationPool2 application pool which has an identity of ID\_ApplicationPool2. The ID\_ApplicationPool1 account is given access to the files in App1, but denied access to the files in App2. ID\_ApplicationPool2 is given access to the files in App2, but denied access to the files in App1. Note that you cannot make this separation on Windows 2000 or Windows XP Professional, because on those operating systems, the process identity for all ASP.NET applications is a single identity. |

You control impersonation using the [identity](http://msdn.microsoft.com/en-us/library/72wdk8cc.ASPX) configuration element. As with other configuration directives, this directive applies hierarchically. A minimal configuration file to enable impersonation for an application might look like the following example:

<configuration>

<system.web>

<identity impersonate="true"/>

</system.web>

</configuration>

You can also add support for specific names to run an application as a configurable identity, as shown in the following example:

<identity impersonate="true"

userName="contoso\Jane"

password="\*\*\*\*\*\*\*\*" />

Substitute the correct password for the value listed in the previous example.

|  |
| --- |
| **Description: NoteNote** |
| In the preceding example, the user name and password are stored in clear text in the configuration file. To improve the security of your application, it is recommended that you restrict the access to your Web.config file using an Access Control List (ACL) and that you encrypt the [identity](http://msdn.microsoft.com/en-us/library/72wdk8cc.ASPX)configuration element in your Web.config file using protected configuration. For more information, see [Encrypting Configuration Information Using Protected Configuration](http://msdn.microsoft.com/en-us/library/53tyfkaw.ASPX). |

The configuration illustrated in the example enables the entire application to run using the contoso\Jane identity, regardless of the identity of the request. This type of impersonation can be delegated to another computer. That is, if you specify the user name and password for the impersonated user, you can connect to another computer on the network and request resources, such as files or access to SQL Server, using integrated security. If you enable impersonation and do not specify a domain account as the identity, you will not be able to connect to another computer on the network unless your IIS application is configured to use Basic authentication.

|  |
| --- |
| **Description: NoteNote** |
| On Windows 2000, you cannot impersonate using specific user credentials for the identity of the ASP.NET worker process. But you can enable impersonation without specific user credentials so that your application impersonates the identity determined by IIS. For more information, see article 810204, "PRB: Per Request Impersonation Does Not Work on Windows 2000 with ASP.NET," in the Microsoft Knowledge Base at http://support.microsoft.com. |

[Reading the Impersonated Identity](javascript:void(0))

The following code example shows how to programmatically read the identity of the impersonated user:

C#

[VB](http://msdn.microsoft.com/en-us/library/xh507fc5(d=printer).ASPX?cs-save-lang=1&cs-lang=vb#code-snippet-3)

String username =

System.Security.Principal.WindowsIdentity.GetCurrent().Name;