.NET Framework 4 - ASP.NET

**ASP.NET Authentication**

Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity. Once an identity has been authenticated, the authorization process determines whether that identity has access to a given resource.

ASP.NET implements authentication through authentication providers, the code modules that contain the code necessary to authenticate the requestor's credentials. The topics in this section describe the authentication providers built into ASP.NET.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifIn This Section

|  |  |
| --- | --- |
| **Term** | **Definition** |
| [Windows Authentication Provider](http://msdn.microsoft.com/en-us/library/907hb5w9.aspx) | Provides information on how to use Windows authentication in conjunction with Microsoft Internet Information Services (IIS) authentication to secure ASP.NET applications. |
| [Forms Authentication Provider](http://msdn.microsoft.com/en-us/library/9wff0kyh.aspx) | Provides information on how to create an application-specific login form and perform authentication using your own code. A convenient way to work with forms authentication is to use ASP.NET membership and ASP.NET login controls, which together provide a way to collect user credentials, authenticate them, and manage them, using little or no code. For more information, see [Managing Users by Using Membership](http://msdn.microsoft.com/en-us/library/tw292whz.aspx) and [ASP.NET Login Controls Overview](http://msdn.microsoft.com/en-us/library/ms178329.aspx). |

You might also consider using Windows Live ID for user authentication. For information about how to use Windows Live ID to authenticate users for you website, see [Windows Live ID SDK](http://go.microsoft.com/fwlink/?LinkId=106346).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifRelated Sections

[ASP.NET Web Application Security](http://msdn.microsoft.com/en-us/library/330a99hc.aspx)

Provides links to additional topics that describe how to secure your ASP.NET Web applications.

[ASP.NET Configuration Overview](http://msdn.microsoft.com/en-us/library/ms178683.aspx)

Provides links to topics on how to use configuration files and the configuration API to manage the run-time behavior of ASP.NET.

.NET Framework 4 - ASP.NET

**Windows Authentication Provider**

Windows Authentication treats the user identity supplied by Microsoft Internet Information Services (IIS) as the authenticated user in an ASP.NET application. IIS provides a number of authentication mechanisms to verify user identity, including anonymous authentication, Windows integrated (NTLM) authentication, Windows integrated (Kerberos) authentication, Basic (base64 encoded) authentication, Digest authentication, and authentication based on client certificates.

Windows Authentication is implemented in ASP.NET using the [WindowsAuthenticationModule](http://msdn.microsoft.com/en-us/library/system.web.security.windowsauthenticationmodule.aspx) module. The module constructs a [WindowsIdentity](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsidentity.aspx) based on the credentials supplied by IIS and sets the identity as the current [User](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.user.aspx) property value for the application.

Windows Authentication is the default authentication mechanism for ASP.NET applications and is identified as the authentication mode for an application using the [authentication](http://msdn.microsoft.com/en-us/library/532aee0e.aspx) configuration element, as shown in the following code example.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl05_code');" \o "Copy Code)

<system.web>

<authentication mode="Windows"/>

</system.web>

Impersonating the Windows Identity

Although the Windows Authentication mode sets the value of the current [User](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.user.aspx) property to a [WindowsIdentity](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsidentity.aspx) based on the credentials supplied by IIS, it does not modify the Windows identity that is supplied to the operating system. The Windows identity supplied to the operating system is used for permission checking, such as NTFS file permissions, or for connecting to a database using integrated security. By default, this Windows identity is the identity of the ASP.NET process. On Microsoft Windows 2000 and Windows XP Professional, this is the identity of the ASP.NET worker process, which is the local ASPNET account. On Windows Server 2003, this is the identity of the IIS Application Pool that the ASP.NET application is part of. By default, this is the NETWORK SERVICE account.

You can configure the Windows identity of your ASP.NET application as the Windows identity supplied by IIS by enabling impersonation. That is, you instruct your ASP.NET application to impersonate the identity supplied by IIS for all tasks that the Windows operating system authenticates, including file and network access.

To enable impersonation for your Web application, in the application's Web.config file set the **impersonate** attribute of the [identity](http://msdn.microsoft.com/en-us/library/72wdk8cc.aspx) element to **true**, as shown in the following code example.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl27_ctl00_ctl03_code');" \o "Copy Code)

<system.web>

<authentication mode="Windows"/>

<identity impersonate="true"/>

</system.web>

For more information on the ASP.NET process identity, see [Configuring ASP.NET Process Identity](http://msdn.microsoft.com/en-us/library/dwc1xthy.aspx). For more information on impersonation, see the [Impersonate](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsidentity.impersonate.aspx) method.

Enabling Authorization using NTFS ACLs

You can improve the security of your ASP.NET application by securing the application's files using the NTFS file system and Access Control Lists (ACLs). ACLs enable you to specify which users and groups of users have access to your application's files. For a list of the minimum required NTFS file permissions that a Windows identity needs to run as the identity of an ASP.NET page, see [ASP.NET Required Access Control Lists (ACLs)](http://msdn.microsoft.com/en-us/library/kwzs111e.aspx).

|  |
| --- |
| **NoteNote** |
| You can also use ASP.NET roles to manage user authorization for pages and sections of your Web application. For more information, see [Managing Authorization Using Roles](http://msdn.microsoft.com/en-us/library/9ab2fxh0.aspx). |

.NET Framework 4 - ASP.NET

**Forms Authentication Provider**

Forms authentication enables you to authenticate the user name and password of your users using a login form that you create. Unauthenticated requests are redirected to a login page, where the user provides credentials and submits the form. If the application authenticates the request, the system issues a ticket that contains a key for reestablishing the identity for subsequent requests.

The topics in this section describe how to use forms authentication to create a custom login system.

|  |
| --- |
| **NoteNote** |
| A convenient way to use forms authentication is to use ASP.NET membership (which stores user credentials) and the ASP.NET login controls (which you can use to create a login page). |

.NET Framework 4 - ASP.NET

**ASP.NET Forms Authentication Overview**

Forms authentication lets you authenticate users by using your own code and then maintain an authentication token in a cookie or in the page URL. Forms authentication participates in the ASP.NET page life cycle through the [FormsAuthenticationModule](http://msdn.microsoft.com/en-us/library/system.web.security.formsauthenticationmodule.aspx) class. You can access forms authentication information and capabilities through the [FormsAuthentication](http://msdn.microsoft.com/en-us/library/system.web.security.formsauthentication.aspx) class.

To use forms authentication, you create a login page that collects credentials from the user and that includes code to authenticate the credentials. Typically you configure the application to redirect requests to the login page when users try to access a protected resource, such as a page that requires authentication. If the user's credentials are valid, you can call methods of the [FormsAuthentication](http://msdn.microsoft.com/en-us/library/system.web.security.formsauthentication.aspx) class to redirect the request back to the originally requested resource with an appropriate authentication ticket (cookie). If you do not want the redirection, you can just get the forms authentication cookie or set it. On subsequent requests, the user's browser passes the authentication cookie with the request, which then bypasses the login page.

You configure forms authentication by using the [authentication](http://msdn.microsoft.com/en-us/library/532aee0e.aspx) configuration element. In the simplest case, you have a login page. In the configuration file, you specify a URL to redirect unauthenticated requests to the login page. You then define valid credentials, either in the Web.config file or in a separate file. The following example shows a section from a configuration file that specifies a login page and authentication credentials for the [Authenticate](http://msdn.microsoft.com/en-us/library/system.web.security.formsauthentication.authenticate.aspx) method. The passwords have been encrypted by using the [HashPasswordForStoringInConfigFile](http://msdn.microsoft.com/en-us/library/system.web.security.formsauthentication.hashpasswordforstoringinconfigfile.aspx) method.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

<authentication mode="Forms">

<forms name="SavingsPlan" loginUrl="/Login.aspx">

<credentials passwordFormat="SHA1">

<user name="Kim"

password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>

<user name="John"

password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>

</credentials>

</forms>

</authentication>

After successful authentication, the [FormsAuthenticationModule](http://msdn.microsoft.com/en-us/library/system.web.security.formsauthenticationmodule.aspx) module sets the value of the [User](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.user.aspx) property to a reference to the authenticated user. The following code example shows how to programmatically read the identity of the forms-authenticated user.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl10_code');" \o "Copy Code)

Dim authUser2 As String = User.Identity.Name

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl11_code');" \o "Copy Code)

String authUser2 = User.Identity.Name;

Forms Authentication, ASP.NET Membership, and Login Controls

A convenient way to work with forms authentication is to use ASP.NET membership and ASP.NET login controls. ASP.NET membership lets you store and manage user information and includes methods to authenticate users. ASP.NET login controls work with ASP.NET membership. They encapsulate the logic to prompt users for credentials, validate users, recover or replace passwords, and so on. In effect, ASP.NET membership and ASP.NET login controls provide a layer of abstraction over forms authentication. These features replace most or all the work that you would ordinarily have to do to use forms authentication. For more information, see [Managing Users by Using Membership](http://msdn.microsoft.com/en-us/library/tw292whz.aspx) and the [ASP.NET Login Controls Overview](http://msdn.microsoft.com/en-us/library/ms178329.aspx).

Forms Authentication and the Authentication Service

You can also access forms authentication as a Windows Communication Framework (WCF) service by using the ASP.NET authentication service. The authentication service enables you to use forms authentication from any application that can send and consume messages in SOAP format. The authentication service accepts user credentials and returns a forms authentication cookie.

For example, you can log in users from an application that was not developed with the .NET Framework. For more information, see [Windows Communication Foundation Authentication Service Overview](http://msdn.microsoft.com/en-us/library/bb386582.aspx).

Designing Distributed Applications with Visual Studio .NET

**ASP.NET Authentication**

ASP.NET implements additional authentication schemes using authentication providers, which are separate from and apply only after the IIS authentication schemes. ASP.NET supports the following authentication providers:

* Windows (default)
* Forms
* Passport
* None

To enable an authentication provider for an ASP.NET application, use the authentication element in either machine.config or Web.config as follows:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl01_code');" \o "Copy Code)

<system.web>

<!-- mode=[Windows|Forms|Passport|None] -->

<authentication mode="Windows" />

</system.web>

Each ASP.NET authentication provider supports an OnAuthenticate event that occurs during the authentication process, which you can use to implement a custom authorization scheme. The primary purpose of this event is to attach a custom object that implements the [IPrincipal Interface](http://msdn.microsoft.com/en-us/library/system.security.principal.iprincipal(VS.71).aspx) to the context.

Which ASP.NET authentication provider you use typically depends upon which IIS authentication scheme you choose. If you are using any of the IIS authentication schemes other than Anonymous, you will likely use the Windows authentication provider. Otherwise, you will use Forms, Passport, or None.

For more information, see [<authentication> Element](http://msdn.microsoft.com/en-us/library/532aee0e(VS.71).aspx) and [ASP.NET Authentication](http://msdn.microsoft.com/en-us/library/aa719553(VS.71).aspx).

**Windows**

The Windows authentication provider relies upon IIS to perform the required authentication of a client. After IIS authenticates a client, it passes a security token to ASP.NET. ASP.NET constructs and attaches an object of the WindowsPrincipal Class to the application context based on the security token it receives from IIS. For more information, see [Windows Authentication Provider](http://msdn.microsoft.com/en-us/library/aa720716(VS.71).aspx) and [WindowsPrincipal Class](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsprincipal(VS.71).aspx).

**Pro**

* Authenticates using Windows accounts, so you do not need to write any custom authentication code.

**Con**

* May require the use and management of individual Windows user accounts.

In addition, each IIS authentication scheme has its own associated pros and cons, which you should consider when choosing a security model. For more information, see [IIS Authentication](http://msdn.microsoft.com/en-us/library/aa292114(VS.71).aspx).

**Implementation**

To implement Windows authentication, refer to the applicable IIS Authentication schemes. For more information, see [IIS Authentication](http://msdn.microsoft.com/en-us/library/aa292114(VS.71).aspx).

**Forms (Cookie)**

The Forms authentication provider is an authentication scheme that makes it possible for the application to collect credentials using an HTML form directly from the client. The client submits credentials directly to your application code for authentication. If your application authenticates the client, it issues a cookie to the client that the client presents on subsequent requests. If a request for a protected resource does not contain the cookie, the application redirects the client to the logon page. When authenticating credentials, the application can store credentials in a number of ways, such as a configuration file or a SQL Server database. For more information, see [Forms Authentication Provider](http://msdn.microsoft.com/en-us/library/aa720709(VS.71).aspx).

**Note** An ISAPI server extension only handles those resources for which it has an application mapping. For example, the ASP.NET ISAPI server extension only has application mappings for particular resources, such as .asax, .ascx, .aspx, .asmx, and .config files to name a few. By default, the ASP.NET ISAPI server extension, and subsequently the Forms authentication provider, does not process any requests for non-ASP.NET resources, such as .htm, .jpg or .gif files.

**Pros**

* Makes it possible for custom authentication schemes using arbitrary criteria.
* Can be used for authentication or personalization.
* Does not require corresponding Windows accounts.

**Cons**

* Is subject to replay attacks for the lifetime of the cookie, unless using SSL/TLS.
* Is only applicable for resources mapped to Aspnet\_isapi.dll.

**Implementation**

To implement forms authentication you must create your own logon page and redirect URL for unauthenticated clients. You must also create your own scheme for account authentication. The following is an example of a Web.config configuration using Forms authentication:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl10_code');" \o "Copy Code)

<!-- Web.config file -->

<system.web>

<authentication mode="Forms">

<forms forms="401kApp" loginUrl="/login.aspx" />

</authentication>

</system.web>

Because you are implementing your own authentication, you will typically configure IIS for Anonymous authentication.

**Passport**

The Passport authentication provider is a centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites. Passport is a forms-based authentication service. When member sites register with Passport, the Passport service grants a site-specific key. The Passport logon server uses this key to encrypt and decrypt the query strings passed between the member site and the Passport logon server. For more information, see [Passport Authentication Provider](http://msdn.microsoft.com/en-us/library/f8e50t0f(VS.71).aspx).

**Pros**

* Supports single sign-in across multiple domains.
* Compatible with all browsers.

**Con**

* Places an external dependency for the authentication process.

**Implementation**

To implement Passport, you must register your site with the Passport service, accept the license agreement, and install the Passport SDK prior to use. You must configure your application's Web.config file as follows:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

<!-- Web.config file -->

<system.web>

<authentication mode="Passport" />

</system.web>

For more information, see the Microsoft Passport Web site (<http://www.passport.com/>).

**None (Custom Authentication)**

Specify "None" as the authentication provider when users are not authenticated at all or if you plan to develop custom authentication code. For example, you may want to develop your own authentication scheme using an ISAPI filter that authenticates users and manually creates an object of the GenericPrincipal Class. For more information, see [GenericPrincipal Class](http://msdn.microsoft.com/en-us/library/system.security.principal.genericprincipal(VS.71).aspx).

**Note** An ISAPI server extension only handles those resources for which it has an application mapping. For example, the ASP.NET ISAPI server extension only has application mappings for particular resources, such as .asax, .ascx, .aspx, .asmx, and .config files to name a few. By default, the ASP.NET ISAPI server extension, and subsequently the None (custom) authentication provider, does not process any requests for non-ASP.NET resources, such as .htm, .jpg or .gif files.

**Pros**

* Offers total control of the authentication process providing the greatest flexibility.
* Provides the highest performance if you do not implement an authentication method.

**Cons**

* Custom-built authentication schemes are seldom as secure as those provided by the operating system.
* Requires extra work to custom-build an authentication scheme.

**Implementation**

To implement no authentication or to develop your own custom authentication, create a custom ISAPI filter to bypass IIS authentication. Use the following Web.config configuration:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl15_code');" \o "Copy Code)

<!-- Web.config file -->

<system.web>

<authentication mode="None" />

</system.web>

.NET Framework Developer's Guide

**ASP.NET Authentication**

Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity. Once an identity has been authenticated, the authorization process determines whether that identity has access to a given resource.

ASP.NET implements authentication through authentication providers, the code modules that contain the code necessary to authenticate the requestor's credentials. ASP.NET supports the authentication providers described in the following table.

|  |  |
| --- | --- |
| **ASP.NET authentication provider** | **Description** |
| **Forms authentication** | A system by which unauthenticated requests are redirected to an HTML form using HTTP client-side redirection. The user provides credentials and submits the form. If the application authenticates the request, the system issues a cookie that contains the credentials or a key for reacquiring the identity. Subsequent requests are issued with the cookie in the request headers; they are authenticated and authorized by an ASP.NET event handler using whatever validation method the application developer specifies. |
| **Passport authentication** | Centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites. |
| **Windows authentication** | ASP.NET uses Windows authentication in conjunction with Microsoft Internet Information Services (IIS) authentication. Authentication is performed by IIS in one of three ways: basic, digest, or Integrated Windows Authentication. When IIS authentication is complete, ASP.NET uses the authenticated identity to authorize access. |

To enable an authentication provider for an ASP.NET application, you only need to create an entry for the application configuration file as follows.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl01_code');" \o "Copy Code)

// Web.config file

<authentication mode= "[Windows|Forms|Passport|None]"/>

The mode is set to one of the authentication modes: **Windows**, **Forms**, **Passport**, or **None**. The default is Windows. If the mode is **None**, ASP.NET does not apply any additional authentication to the request - this can be useful when you want to implement a custom authentication scheme, or if you are solely using anonymous authentication and want the highest possible level of performance.

The authentication mode cannot be set at a level below the application root directory. As is the case with other ASP.NET modules, subdirectories in the URL space inherit authentication modules unless explicitly overridden.