**ASP.NET Profile Properties Overview**

[**http://msdn.microsoft.com/en-us/library/2y3fs9xs%28v=vs.100%29.aspx**](http://msdn.microsoft.com/en-us/library/2y3fs9xs%28v=vs.100%29.aspx)

**.NET Framework 4**

[Other Versions](javascript:;)

19 out of 30 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/2y3fs9xs%28v=vs.100%29.aspx#feedback)

In many applications, you want to store and use information that is unique to a user. When a user visits your site, you can use the information you have stored to present the user with a personalized version of your Web application. Personalizing an application requires a number of elements: you must store the information using a unique user identifier, be able to recognize users when they visit again, and then fetch the user information as needed. To simplify your applications, you can use the ASP.NET profile feature, which can perform all of these tasks for you.

The ASP.NET profile feature associates information with an individual user and stores the information in a persistent format. Profiles allow you to manage user information without requiring you to create and maintain your own database. In addition, the ASP.NET profile feature makes the user information available using a strongly typed API that you can access from anywhere in your application.

You can store objects of any type using profiles. The profile feature provides a generic storage feature that allows you to define and maintain almost any kind of data while still making the data available in a type-safe manner.

[How ASP.NET Profiles Work](javascript:void(0))

To use profiles, you first enable profiles by modifying the configuration file for your ASP.NET Web application. As part of the configuration, you specify a profile provider, which is the underlying class that performs the low-level tasks of storing and retrieving profile data. You can use the profile provider included with the .NET Framework, which stores profile data in SQL Server, or you can create and use your own profile provider as described in the topic [Implementing a Profile Provider](http://msdn.microsoft.com/en-us/library/0580x1f5%28v=vs.100%29.aspx). You can specify an instance of the [SqlProfileProvider](http://msdn.microsoft.com/en-us/library/system.web.profile.sqlprofileprovider%28v=vs.100%29.aspx) that connects to a database of your choosing, or you can use the default instance of the [SqlProfileProvider](http://msdn.microsoft.com/en-us/library/system.web.profile.sqlprofileprovider%28v=vs.100%29.aspx) that stores profile data on the local Web server.

You configure the profile feature by defining a list of properties whose values you want to maintain. For example, you might want to store the user's postal code so that your application can offer region-specific information, such as weather reports. In the configuration file, you would define a profile property named PostalCode. The profile section of the configuration file might look like the following:

<profile>

<properties>

<add name="PostalCode" />

</properties>

</profile>

When your application runs, ASP.NET creates a ProfileCommon class, which is a dynamically generated class that inherits the [ProfileBase](http://msdn.microsoft.com/en-us/library/system.web.profile.profilebase%28v=vs.100%29.aspx) class. The dynamic ProfileCommon class includes properties created from the profile property definitions you specify in your application configuration. An instance of this dynamic ProfileCommon class is then set as the value of the [Profile](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.profile%28v=vs.100%29.aspx) property of the current [HttpContext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext%28v=vs.100%29.aspx) and is available to pages in your application.

In your application, you collect the value or values you want to store and assign them to the profile properties you have defined. For example, your application's home page might contain a text box that prompts the user to enter a postal code. When the user enters a postal code, you set a [Profile](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.profile%28v=vs.100%29.aspx) property to store the value for the current user, as in the following example:

C#

[VB](http://msdn.microsoft.com/en-us/library/2y3fs9xs%28v=vs.100%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

Profile.PostalCode = txtPostalCode.Text;

When you set a value for Profile.PostalCode, the value is automatically stored for the current user. You do not need to write any code to determine who the current user is or explicitly store the value in a database—the profile feature performs these tasks for you.

When you want to use the value, you can get it in much the same way that you set it. For example, the following code example shows how to call an imaginary function named GetWeatherInfo, passing it the current user's postal code as stored in a profile:

C#

[VB](http://msdn.microsoft.com/en-us/library/2y3fs9xs%28v=vs.100%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

weatherInfo = GetWeatherInfo( Profile.PostalCode );

You do not need to explicitly determine who the user is or perform any database lookups. Simply getting the property value out of a profile causes ASP.NET to perform the necessary actions to identify the current user and look up the value in the persistent profile store.

[See Also](javascript:void(0))

**Concepts**

[ASP.NET Profile Properties Overview](http://msdn.microsoft.com/en-us/library/2y3fs9xs%28v=vs.100%29.aspx)

[User Identification for ASP.NET Profile Properties](http://msdn.microsoft.com/en-us/library/ewfkf772%28v=vs.100%29.aspx)

[Defining ASP.NET Profile Properties](http://msdn.microsoft.com/en-us/library/d8b58y5d%28v=vs.100%29.aspx)

[ASP.NET Profile Providers](http://msdn.microsoft.com/en-us/library/014bec1k%28v=vs.100%29.aspx)