**Defining ASP.NET Profile Properties**

This page is specific to

**Microsoft Visual Studio 2008/.NET Framework 3.5**

Other versions are also available for the following:

[Microsoft Visual Studio 2005/.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/d8b58y5d(VS.80).aspx)

[.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/d8b58y5d(VS.85).aspx)

[Microsoft Visual Studio 2010/.NET Framework 4](http://msdn.microsoft.com/en-us/library/d8b58y5d(VS.100).aspx)

The ASP.NET profile feature allows you to store simple (scalar) values, collections and other complex types, and user-defined types.

http://i.msdn.microsoft.com/Global/Images/clear.gif Property Definition Information

When you define a property in the profile, you specify a name that you will use to refer to the property. For example, if you want to store a postal code, you can name the property PostalCode and you can then get and set the property value as Profile.PostalCode.

You can optionally define the following additional characteristics for each property:

* **type**   Specifies the type for the property. The default is [String](http://msdn.microsoft.com/en-us/library/system.string.aspx). You can specify any .NET class as the type ([Int32](http://msdn.microsoft.com/en-us/library/system.int32.aspx), [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx), [StringCollection](http://msdn.microsoft.com/en-us/library/system.collections.specialized.stringcollection.aspx), and so on). If the type is not defined in the .NET Framework, you must ensure that your Web application has access to the type. You can include the type's compiled assembly in the Web site's Bin directory or in the global assembly cache (GAC), or you can put the source code for the type in the Web site's App\_Code directory.
* **serializeAs**   Specifies the serialization formatter (string, binary, XML, or provider-specific serialization). For details, see [Serialization](http://msdn.microsoft.com/en-us/library/7ay27kt9.aspx). The default serialization is string.
* **allowAnonymous**   Specifies a Boolean value that indicates whether the property is managed for anonymous users. By default, this is **false**. If you want the property to be available for unauthenticated users, you can set the property to **true**.
* **defaultValue**   Specifies a value that the property is initialized with.
* **readOnly**   Specifies a Boolean value that indicates whether the property can be modified.
* **provider**   Specifies a provider specific to the property. By default, all properties are managed using the default provider specified for profile properties, but individual properties can also use different providers.
* **customProviderData**   Specifies an optional string containing custom information that is passed to the profile provider. Individual providers can implement custom logic for using this data.

Additionally, profile properties can be organized as groups of properties using the group configuration element, which is discussed later in this topic.

http://i.msdn.microsoft.com/Global/Images/clear.gif Working with Scalar Values

Storing scalar values such as strings, numerical values, or [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx) values in a profile requires only minimal configuration. You must supply a name and the type. The profile system will convert the value from the specified type to a string and back as required for storage. When you access the property via the [Profile](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.profile.aspx) property, it will be typed appropriately.

For example, if you want to store the user's name, weight, and birth date, you can define a property called Name of type [String](http://msdn.microsoft.com/en-us/library/system.string.aspx), another named Weight that is of type [Int32](http://msdn.microsoft.com/en-us/library/system.int32.aspx), and a third named BirthDate of type [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx). In the configuration file, the property definitions will look like the following:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl33_ctl00_ctl05_code');" \o "Copy Code)

<profile defaultProvider="AspNetSqlProfileProvider">

<properties>

<add name="Name" />

<add name="Weight" type="System.Int32" />

<add name="BirthDate"

type="System.DateTime" />

</properties>

</profile>

For the Name property, you do not need to explicitly specify a type, because the property is of type [String](http://msdn.microsoft.com/en-us/library/system.string.aspx), the default. For any other type, you must provide a fully qualified type reference.

When you get or set the property values, you will need to work with the appropriate type in your code. The following code example shows how to work with the BirthDate property:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl33_ctl00_ctl07_code');" \o "Copy Code)

Dim bday As DateTime = Profile.BirthDate

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl33_ctl00_ctl08_code');" \o "Copy Code)

DateTime bday = Profile.BirthDate;

http://i.msdn.microsoft.com/Global/Images/clear.gif Working with Complex Property Types

You can also store complex types, such as collections, in user profiles. For complex types, you must provide information about how the type should be serialized so that the profile system can get and set the property value as the appropriate type.

The following example shows a property definition for a value typed as a collection:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl00_code');" \o "Copy Code)

<profile defaultProvider="AspNetSqlProfileProvider">

<properties>

<add name="FavoriteURLs"

type="System.Collections.Specialized.StringCollection"

serializeAs="Xml" />

</properties>

</profile>

To set a property of this type, you might use code such as the following:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl01_code');" \o "Copy Code)

Dim favorites As System.Collections.Specialized.StringCollection

favorites = Profile.FavoriteURLs

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl02_code');" \o "Copy Code)

System.Collections.Specialized.StringCollection favorites;

favorites = Profile.FavoriteURLs;

http://i.msdn.microsoft.com/Global/Images/clear.gif Working with User-Defined Property Types

You can also store and use profile property values that are instances of classes that you create yourself. The class you create must support serialization for the members you want to store in the user profile.

The following code example illustrates a simple ShoppingCart class that maintains a collection of Cart items that in turn store an item identifier, name, and cost:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl35_ctl00_ctl00_code');" \o "Copy Code)

Namespace Samples.AspNet.Profile

<Serializable()> \_

Public Class ShoppingCart

Public Created As DateTime

Public LastUpdated As DateTime

Public CartItems As Dictionary(Of String, CartItem) = \_

New Dictionary(Of String, CartItem)()

End Class

<Serializable()> \_

Public Class CartItem

Public Sub New(itemId As Integer, itemName As String, \_

itemCost As Double)

ID = itemId

Name = itemName

Cost = itemCost

End Sub

Dim ID As Integer

Dim Name As String

Dim Cost As Double

End Class

End Namespace

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl35_ctl00_ctl01_code');" \o "Copy Code)

namespace Samples.AspNet.Profile

{

[Serializable]

public class ShoppingCart {

public DateTime Created;

public DateTime LastUpdated;

public Dictionary<string, CartItem> CartItems = new Dictionary<string, CartItem>();

}

[Serializable]

public class CartItem {

public CartItem(int itemId, string itemName, double itemCost)

{

ID = itemId;

Name = itemName;

Cost = itemCost;

}

int ID;

string Name;

double Cost;

}

}

To configure the user profile to use store instances of this class, add the following section to the application's Web.config file:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl35_ctl00_ctl02_code');" \o "Copy Code)

<profile defaultProvider="AspNetSqlProfileProvider">

<properties>

<add name="MyCart"

type="Samples.AspNet.Profile.ShoppingCart"

serializeAs="Binary" />

</properties>

</profile>

To store the custom type data in the user profile, create an instance of the custom type as you would in any application, and then assign it to the profile property you defined for that type. The following code example shows how to work with a profile property created as a custom type:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl35_ctl00_ctl03_code');" \o "Copy Code)

Dim bookCart As ShoppingCart = New ShoppingCart()

bookCart.CartItems.Add("Contoso", \_

New CartItem(37843, "Widget", 49.99))

bookCart.CartItems.Add("Microsoft", \_

New CartItem(39232, "Software", 49.99))

Profile.MyCart = bookCart

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl35_ctl00_ctl04_code');" \o "Copy Code)

ShoppingCart bookCart = new ShoppingCart();

bookCart.CartItems.Add("Contoso", new CartItem(37843, "Widget", 49.99));

bookCart.CartItems.Add("Microsoft", new

CartItem(39232, "Software", 49.99));

Profile.MyCart = bookCart;

http://i.msdn.microsoft.com/Global/Images/clear.gif Working with Property Groups

Properties can be organized in the user profile as groups of properties. Profile property groups are specified using the group configuration element. For example, the different properties of user's address information can be grouped together in an Address group. You can then access the grouped properties using the group identifier and the property name (for example, Profile.Address.Street or Profile.Address.City). The following example shows a profile property configuration that organizes some properties in a group.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl36_ctl00_ctl00_code');" \o "Copy Code)

<profile enabled="true">

<properties>

<add name="PostalCode" />

<group name="Address">

<add name="Street" />

<add name="City" />

<add name="CountryOrRegion" />

</group>

</properties>

</profile>