**How to: Build and Run the Profile Provider Example**

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/8zs47k7y(VS.80).aspx)

[.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/8zs47k7y(VS.85).aspx)

[Microsoft Visual Studio 2010/.NET Framework 4](http://msdn.microsoft.com/en-us/library/8zs47k7y(VS.100).aspx)

The topics in this section include the code for an ASP.NET profile provider example. The example provider uses the .NET Framework data provider for ODBC to connect to an ODBC data source. The example uses an Access database as its data source.

This topic describes implementation details about the profile provider example and describes how to build the example and configure an ASP.NET application to use the example provider.

|  |
| --- |
| **NoteNote:** |
| Because data sources support differing SQL syntax, some SQL commands will work with one data source and not with another. As a result, we recommend that you create a profile provider specific to your data source even if you are using the .NET Framework Data Provider for ODBC or the .NET Framework Data Provider for OLEDB to access your data source. For example, you might create separate providers such as SybaseProfileProvider, OracleProfileProvider, and so on. |

To view the code for the example provider, see [Profile Provider Implementation Example](http://msdn.microsoft.com/en-us/library/ta63b872.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Creating the Provider Database

The provider is designed to read and write data in three database tables: Profiles, StockSymbols, and ProfileData. The example profile provider supports two profile properties: a list of stock symbols and a ZIP (postal) code.

Each user profile is uniquely identified in the Profiles table of the database. Profile information such as the application name and last activity date is contained in this table. The Profiles table includes an automatically incremented primary key that is used to uniquely identify each row and relate the table to other tables in the database that contain profile property values.

The StockSymbols table contains the values for the list of stock symbols stored and retrieved using the StockSymbols property. The StockSymbols table has a one-to-many relationship with the Profiles table, because a single user profile can track any number of stock symbols.

The ZipCode property value is stored in the ProfileData table. The ProfileData table has a one-to-one relationship with the Profiles table.

**To create the database tables for storing profiles**

1. Create a new Microsoft Access database or open an existing one.

|  |
| --- |
| **NoteNote:** |
| If you store your database in the file folders of your Web application, we recommend that you store the database in the application's App\_Data folder. The contents of the App\_Data are available to your application code, but are not exposed to Web requests. In addition, the ASP.NET process has permissions to read and write in the App\_Data folder. |

1. In Access or another tool that can create tables in an Access database, issue the following data-definition query to create the Profiles table:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl01_code');" \o "Copy Code)

CREATE TABLE Profiles

(

UniqueID AutoIncrement NOT NULL PRIMARY KEY,

Username Text (255) NOT NULL,

ApplicationName Text (255) NOT NULL,

IsAnonymous YesNo,

LastActivityDate DateTime,

LastUpdatedDate DateTime,

CONSTRAINT PKProfiles UNIQUE (Username, ApplicationName)

)

1. Issue the following data-definition query to create the StockSymbols table:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl02_code');" \o "Copy Code)

CREATE TABLE StockSymbols

(

UniqueID Integer,

StockSymbol Text (10),

CONSTRAINT FKProfiles1 FOREIGN KEY (UniqueID)

REFERENCES Profiles

)

1. Issue the following data-definition query to create the ProfileData table:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl03_code');" \o "Copy Code)

CREATE TABLE ProfileData

(

UniqueID Integer,

ZipCode Text (10),

CONSTRAINT FKProfiles2 FOREIGN KEY (UniqueID)

REFERENCES Profiles

)

http://i.msdn.microsoft.com/Global/Images/clear.gif Providing Event Log Access

If the example provider encounters an exception when working with the data source, the provider writes the details of the exception to the Windows Application event log instead of returning the exception to the ASP.NET application. This is done as a security measure to keep private information about the data source from being exposed in the ASP.NET application.

The example provider specifies an event [Source](http://msdn.microsoft.com/en-us/library/system.diagnostics.eventlog.source.aspx) of "OdbcProfileProvider". Before your ASP.NET application will be able to write to the Application event log successfully, you will need to create access to the event log.

|  |
| --- |
| **NoteNote:** |
| If you do not want the example provider to write exceptions to the event log, you can set the writeExceptionsToEventLog attribute to **false** in the provider configuration in the Web.config file. |

**To provide access to the event log**

* Using Regedit.exe or another Windows registry editing tool, create the following registry key:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl30_ctl00_ctl02_code');" \o "Copy Code)

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application\OdbcProfileProvider

http://i.msdn.microsoft.com/Global/Images/clear.gif Building the Example Provider

In order to use the example provider, you must make the code available to your Web application. You can do so in two ways:

* Put the example provider source code into the application's App\_Code folder. In that case, you do not need to manually compile the example code.
* Compile the code and put the resulting assembly into the application's Bin folder, or strongly name it and place it in the global assembly cache.

**To test the example provider without manually compiling**

1. If your application does not already have one, create a folder named App\_Code under the application root.
2. In the App\_Code folder, create a text file named OdbcProfileProvider.vb or OdbcProfileProvider.cs, depending on what programming language you want to use.

|  |
| --- |
| **NoteNote:** |
| If you already have source code in the App\_Code directory of your application, use the version of the example provider written in the same language as the existing code in the directory. ASP.NET will compile the provider when a page from your ASP.NET application is first requested. |

1. Copy and paste the example source code (in the appropriate language) into the corresponding text file and save the file.

**To compile the example provider**

1. Create a folder for the source files for custom controls and related classes.
2. In the source code folder, create a text file named OdbcProfileProvider.vb or OdbcProfileProvider.cs, depending on what programming language you want to use.

|  |
| --- |
| **NoteNote:** |
| Do not put the source code in the App\_Code folder if you also intend to manually compile the example and add its assembly to the Bin folder. If you do, the provider's type will exist in both the compiled assembly and in the dynamically generated assembly created by ASP.NET from the App\_Code folder. This will create an ambiguous reference when loading the provider and any code that references the type will generate a compiler error. |

1. Copy and paste the source code for each sample into the corresponding text file and save the file.
2. Run the following command from the source code folder to compile the controls and related classes into an assembly.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl31_ctl00_ctl02_code');" \o "Copy Code)

vbc /out:OdbcProfileProvider.dll /t:library OdbcProfileProvider.vb /r:System.Web.dll /r:System.Configuration.dll

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl31_ctl00_ctl03_code');" \o "Copy Code)

csc /out:OdbcProfileProvider.dll /t:library OdbcProfileProvider.cs /r:System.Web.dll /r:System.Configuration.dll

The **/t:library** compiler option tells the compiler to create a library (rather than an executable assembly). The **/out** option provides a name for the assembly and the **/r** option lists the assemblies that are linked to your assembly.

|  |
| --- |
| **NoteNote:** |
| If you cannot execute the compiler command, you must add the .NET Framework installation path to the **PATH** variable before running the command. In Windows, right-click **My Computer**, click **Properties**, click the **Advanced** tab, and then click the **Environment Variables** button. In the **System variables** list, double-click the **Path** variable. In the **Variable value** text box, add a semicolon (;) to the end of the existing values in the text box, and then type the path of your .NET Framework installation. The .NET Framework is usually installed in the Windows installation folder at \Microsoft.NET\Framework\*versionNumber*. |

http://i.msdn.microsoft.com/Global/Images/clear.gif Configuring the Example Provider in an ASP.NET Application

To use the example provider with an ASP.NET Web application, you must configure the application to register the provider.

The example configuration assumes that your Web site is set up to use forms authentication and includes an ASP.NET page called Login.aspx that allows users to log in and establish a user identity. The example configuration also supports anonymous authentication, so users are not required to log in.

**To configure an application to use the example provider**

1. Create an ASP.NET page named Login.aspx and do one of the following:
   * Add a [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control to it, if the application is already configured to use ASP.NET membership.
   * Create a login form and use forms authentication to authenticate users. For details, see [How to: Implement Simple Forms Authentication](http://msdn.microsoft.com/en-us/library/xdt4thhy.aspx).

|  |
| --- |
| **NoteNote:** |
| The configuration element required for forms authentication is illustrated in Step 4. |

1. Create a DSN named "OdbcProfile" on your computer and configure it to include connection information to the Access database you created earlier.
2. If your ASP.NET application does not already have one, create a text file named Web.config in the application's root folder.
3. In the Web.config file, add the following elements:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl32_ctl00_ctl03_code');" \o "Copy Code)

<configuration>

<connectionStrings>

<add name="OdbcProfile" connectionString="DSN=OdbcProfile;" />

</connectionStrings>

<system.web>

<authentication mode="Forms" >

<forms loginUrl="login.aspx"

name=".ASPXFORMSAUTH" />

</authentication>

<anonymousIdentification enabled="true" />

<profile defaultProvider="OdbcProvider">

<providers>

<add

name="OdbcProvider"

type="Samples.AspNet.Profile.OdbcProfileProvider"

connectionStringName="OdbcProfile" />

</providers>

<properties>

<add name="ZipCode"

allowAnonymous="true" />

<add name="CityAndState"

provider="AspNetSqlProfileProvider"

allowAnonymous="true" />

<add name="StockSymbols"

type="System.Collections.ArrayList"

allowAnonymous="true" />

</properties>

</profile>

</system.web>

</configuration>