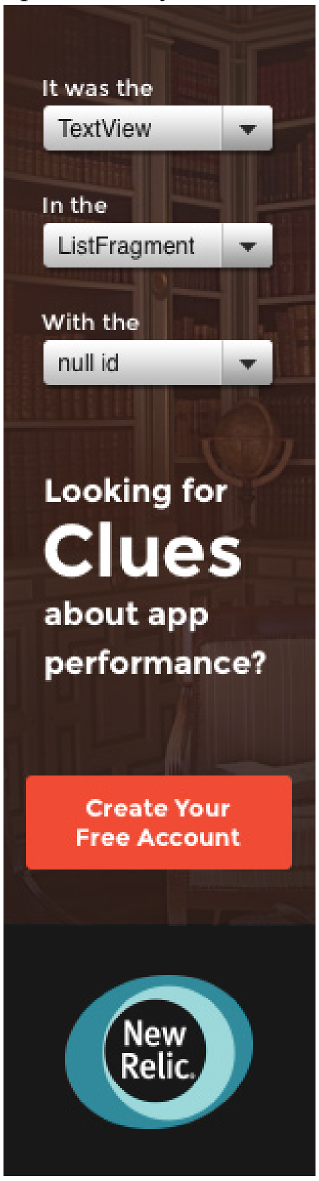
Introducing System.Web.Providers - ASP.NET Universal Providers for Session, Membership, Roles and User Profile on SQL Compact and SQL Azure

junho 16, '11 Posted in ASP.NET | ASP.NET MVC | NuGetPOW
Sponsored By



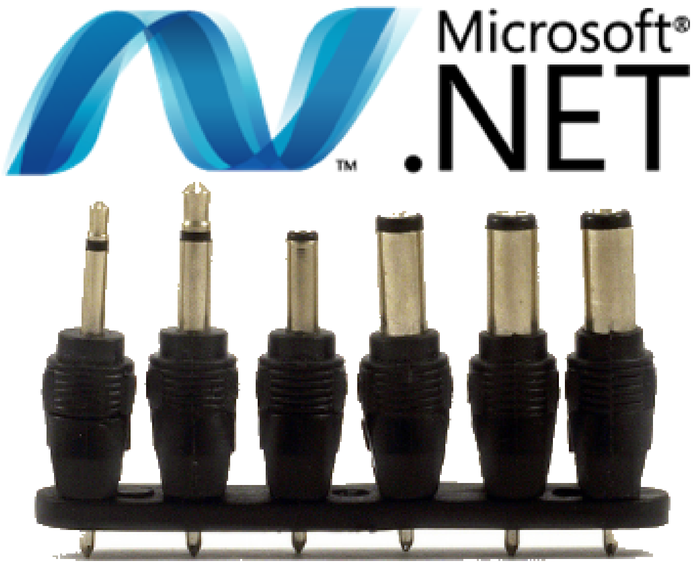
UPDATE #2: Note that the NuGet package has changed its name from System.Web.Providers to Microsoft.AspNet.Providers.

UPDATE: Note that in MVC 4 and ASP.NET 4 and 4.5 the default hash is now HMACSHA256

I always like to remind folks of the equation ASP.NET > (ASP.NET MVC + ASP.NET WebForms). The whole "base of the pyramid" of ASP.NET has lots of things you can use in you applications. Some of these useful bits are Session State, Membership (Users), Roles, Profile data and the provider model that underlies it. Using these isn't for everyone but they are very useful for most applications, even ones as large as the ASP.NET site itself.

Today the Web Platform and Tools team (WPT) is releasing an Alpha of the **ASP.NET Universal Providers** that will extend Session, Membership, Roles and Profile support to SQL Compact Edition and SQL Azure. Other than supporting additional storage options, the providers work like the existing SQL-based providers.

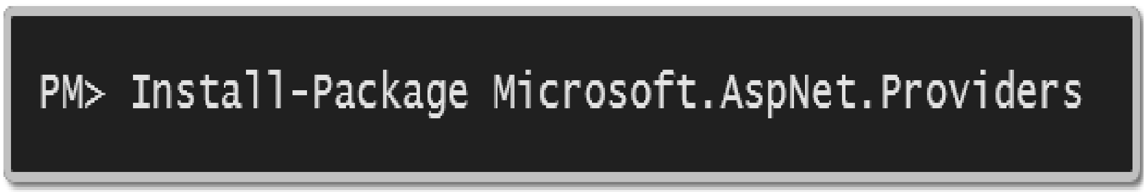
Today these are being released via a NuGet Package, but it's very likely that these Universal Providers will be the default in the next version of ASP.NET.



To enable the providers, the NuGet package adds configuration entries in the *web.config* file. The configuration for these providers is the same as the existing `SqlMembershipProvider` class, but the `type` parameter is set to the type of the new providers, as shown in the following table:

SQL Provider Types	Equivalent Type for Universal Providers
`System.Web.Security.SqlMembershipProvider`	`System.Web.Providers.DefaultMembershipProvider`
`System.Web.Profile.SqlProfileProvider`	`System.Web.Providers.DefaultProfileProvider`
`System.Web.Security.SqlRoleProvider`	`System.Web.Providers.DefaultRoleProvider`
`(Built in provider)`	`System.Web.Providers.DefaultSessionStateProvider`

If you install these, the NuGet package will swap your defaultProviders in your web.config. You can certainly pick and choose the settings for each as well. Here we're changing Profile, Membership, RoleManager and SessionState. The latter is nice as it better allows your session-state-using Azure apps to scale with SQL Azure as the backend storage.



```
PM> Install-Package Microsoft.AspNet.Providers
```

Using these Universal "Default Profile Providers" means all you have to do is set the right connection string and your applications that use these services will work with SQL Server (plus Express), SQL Server Compact and SQL Azure with no code changes from you.

```
<configuration>
    <connectionstrings>
        <add name="DefaultConnection" connectionstring="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\aspnet.mdf;Initial
Catalog=aspnet;Integrated Security=True;User Instance=True;MultipleActiveResultSets=True"
providername="System.Data.SqlClient">
    </add></connectionstrings>
    <system.web>
      <profile defaultprovider="DefaultProfileProvider">
        <providers>
          <add name="DefaultProfileProvider"
type="System.Web.Providers.DefaultProfileProvider"
connectionstringname="DefaultConnection" applicationname="/">
        </add></providers>
      </profile>
      <membership defaultprovider="DefaultMembershipProvider">
```

```
        <providers>
           <add name="DefaultMembershipProvider"
type="System.Web.Providers.DefaultMembershipProvider"
connectionstringname="DefaultConnection" enablepasswordretrieval="false"
<br="">             enablePasswordReset="true" requiresQuestionAndAnswer="false"
requiresUniqueEmail="false"
             maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6"
minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
             applicationName="/" />
        </add></providers>
      </membership>
      <rolemanager defaultprovider="DefaultRoleProvider">
        <providers>
           <add name="DefaultRoleProvider" type="System.Web.Providers.DefaultRoleProvider"
connectionstringname="DefaultConnection" applicationname="/">
        </add></providers>
      </rolemanager>
      <sessionstate mode="Custom" customprovider="DefaultSessionProvider">
        <providers>
          <add name="DefaultSessionProvider"
type="System.Web.Providers.DefaultSessionStateProvider"
connectionstringname="DefaultConnection" applicationname="/">
        </add></providers>
      </sessionstate>
    </system.web>
</configuration>
```

Selecting a Data Store

By default, the NuGet package sets the connection string to use a SQL Server Express database (wrapped here for readability):

```
"Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\aspnetdb.mdf;
    Initial Catalog=aspnet;Integrated Security=True;
    User Instance=True;MultipleActiveResultSets=True" providerName="System.Data.SqlClient"
```

If you want to use SQL Server Compact, change the connection string as shown in the following example:

```
<connectionstrings>
  <add name="Sql_CE" connectionstring="Data Source=|DataDirectory|\MyWebSite.sdf;"
providername="System.Data.SqlServerCe.4.0">
  </add>
</connectionstrings>
```

If you want to use SQL Azure, change the connection string like this example (wrapped for readability):

```
<connectionstrings>
  <add name="Sql_Azure" connectionstring="data source=myDNSName;Initial Catalog=myDatabase;
         User ID=myUserName;Password=myPassword;
         Encrypt=true;Trusted_Connection=false;
         MultipleActiveResultSets=True" <br="">     providerName="System.Data.SqlClient"/>
  </add>
</connectionstrings>
```

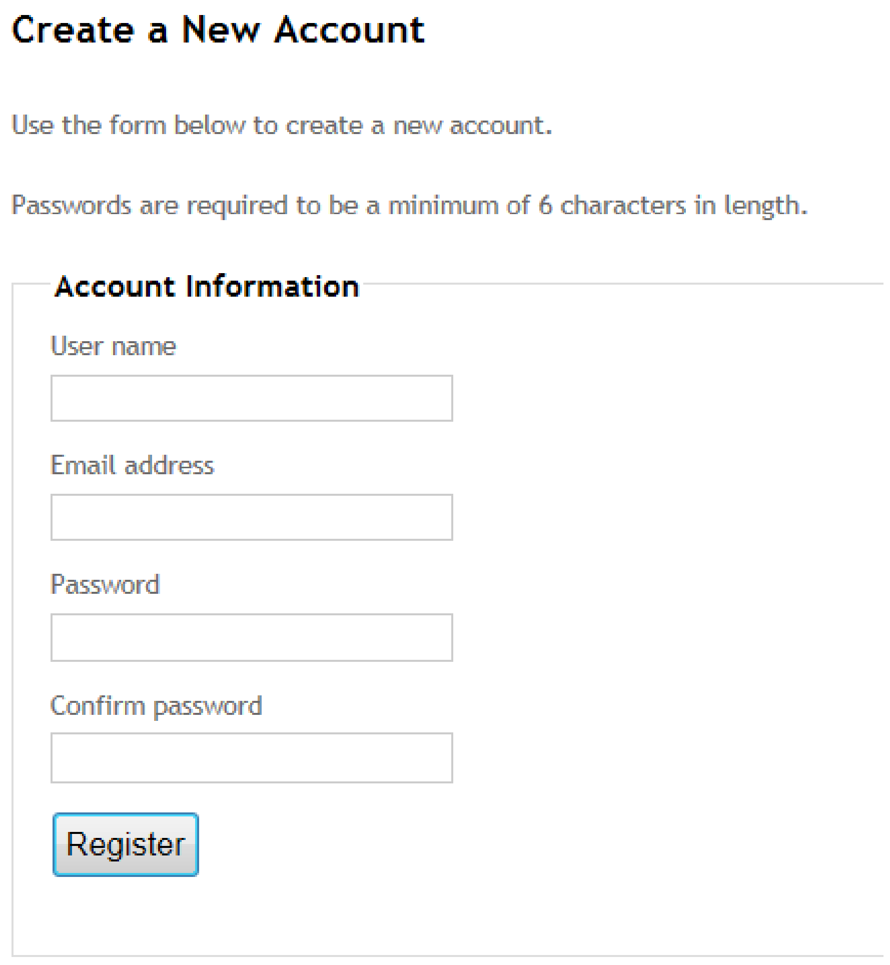
Even though this release is primarily about extending support to all versions of SQL Server, I realize that y'all might not even know about what these things do, so I thought I'd spend a little time explaining. I notice also that there's some confusion on StackOverflow and other sites on how to use Membership and Profile and the like on ASP.NET MVC, so I'll use that for the examples.

Example of Membership, Roles and Profile in ASP.NET MVC (with the Universal Providers)

I'll fire up VS and File | New Project on a new ASP.NET MVC 3 Project. Then I'll right click on References and select Add | Library Reference. The NuGet package id is "**Microsoft.AspNet.Providers**." After this package is installed, I can also install SQL Compact Edition via NuGet if I like and set the connection string to SQL Compact as shown above.

Remember, this is a very functional Alpha, but there may be bugs (report them!) so it might be updated a few times before the next version of ASP.NET is released.

First, I'll run my app and click Register and make a new user named "Scott."

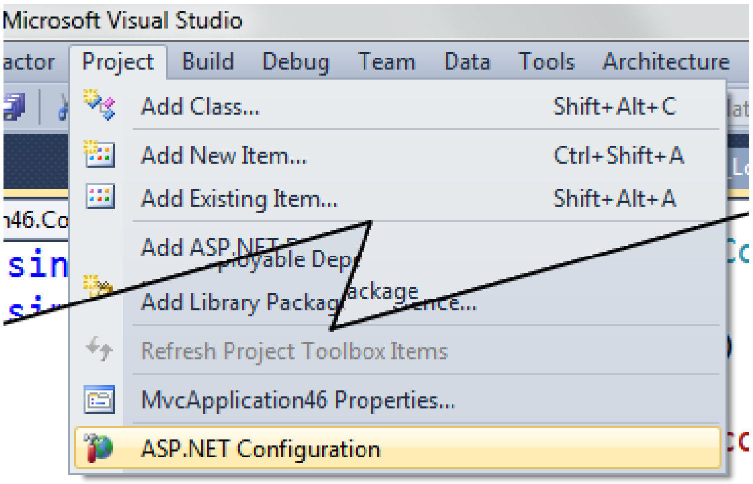

Create a New Account

Use the form below to create a new account.

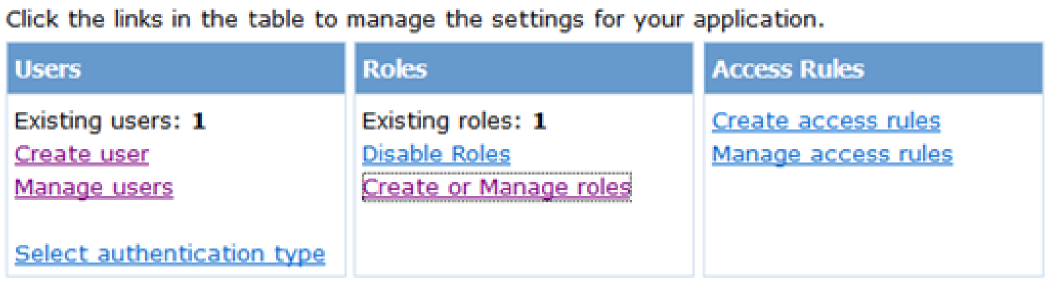
Passwords are required to be a minimum of 6 characters in length.

Adding Roles to a User

Next, from Visual Studio Project menu, I visit ASP.NET Configuration. (I could also write my own admin section and do this programmatically, if I liked).

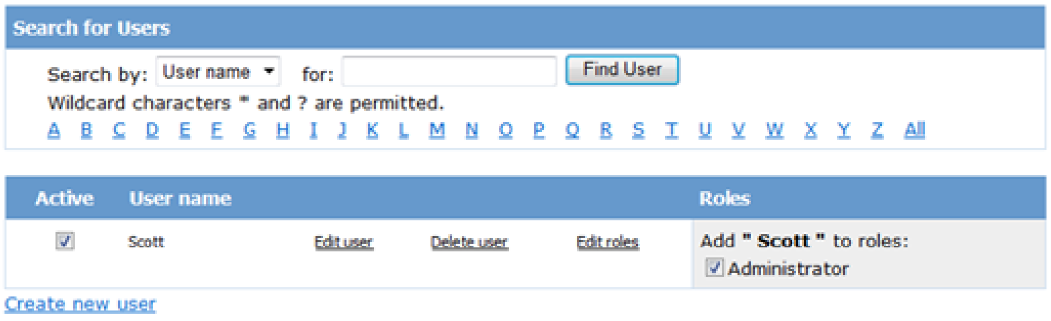

Microsoft Visual Studio
Project
Build
Debug
Team
Data
Tools
Architecture
Add Class... Shift+Alt+C
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A
Add ASP.NET Deployable Dep
Add Library Package Reference...
Refresh Project Toolbox Items
MvcApplication46 Properties...
ASP.NET Configuration

Then from the Security tab, under Roles, I'll create a new Role:


Click the links in the table to manage the settings for your application.

Users	Roles	Access Rules
Existing users: **1**	Existing roles: **1**	Create access rules
Create user	Disable Roles	Manage access rules
Manage users	Create or Manage roles	
Select authentication type		

Then I'll find the Scott User and add him to the Administrator Role:

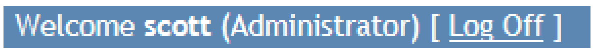

Search for Users

Search by: User name for: Find User
Wildcard characters * and ? are permitted.
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z All

Active	User name				Roles
✓	Scott	Edit user	Delete user	Edit roles	Add " **Scott** " to roles: ✓ Administrator

Create new user

I want to show something if a user is an Administrator. I'll add a little chunk of code to the default website's _LogOnPartial. cshtml so we'll see [Administrator] next to their name of they are one.



I'll add a small line where I ask "User.IsInRole()" like this:

```
@if(Request.IsAuthenticated) {
    <text>Welcome <strong>@User.Identity.Name</strong>
    @(User.IsInRole("Administrator") ? "(Administrator)" : String.Empty)
    [ @Html.ActionLink("Log Off", "LogOff", "Account") ]</text>
}
else {
    @:[ @Html.ActionLink("Log On", "LogOn", "Account") ]
}
```

```
@if (ViewBag.Profile != null) {
    <text>Hey, your birthday is @ViewBag.Profile.Birthdate.ToString("d")! Congrats.
</text>
}
```

So now I have some Roles I can assign to users and check against. I can set whatever roles I want, like Bronze, Silver, Gold, etc.

Adding Profile Information to Users

Let's say I want Users to have a Birthday and I want that to be part of the User Profile. I can just use the Profile object and ask for things via string like this:

```
DateTime? birthday2 = HttpContext.Profile["Birthdate"] as DateTime?; //alternative syntax
```

However, perhaps I'd rather have a stronger typed syntax for my profile.

NOTE: *I've already brought up the issue that User hangs off Controller in MVC 3 but Profile is simply missing. Perhaps that will be fixed in MVC 4. I believe it was a oversight. You shouldn't be digging around in HttpContext if you want your code testable*

I'll make a small CustomProfile object like this that extends ProfileBase:

```
public class MyCustomProfile : ProfileBase
{
    public DateTime? Birthdate {
        get { return this["Birthdate"] as DateTime?; }
        set { this["Birthdate"] = value; }
    }
}
```

Alternatively, I could put the "getting" of the profile in the custom class in a static, or I could use Dependency Injection. It depends on how you want to get to it.

```
public static MyCustomProfile GetUserProfile(string username)
{
    return Create(username) as MyCustomProfile;
}
public static MyCustomProfile GetUserProfile()
{
    return Create(Membership.GetUser().UserName) as MyCustomProfile;
}
```

Then in web.config, I'll update the so the system know the derived Profile class I want used when I ask for one:

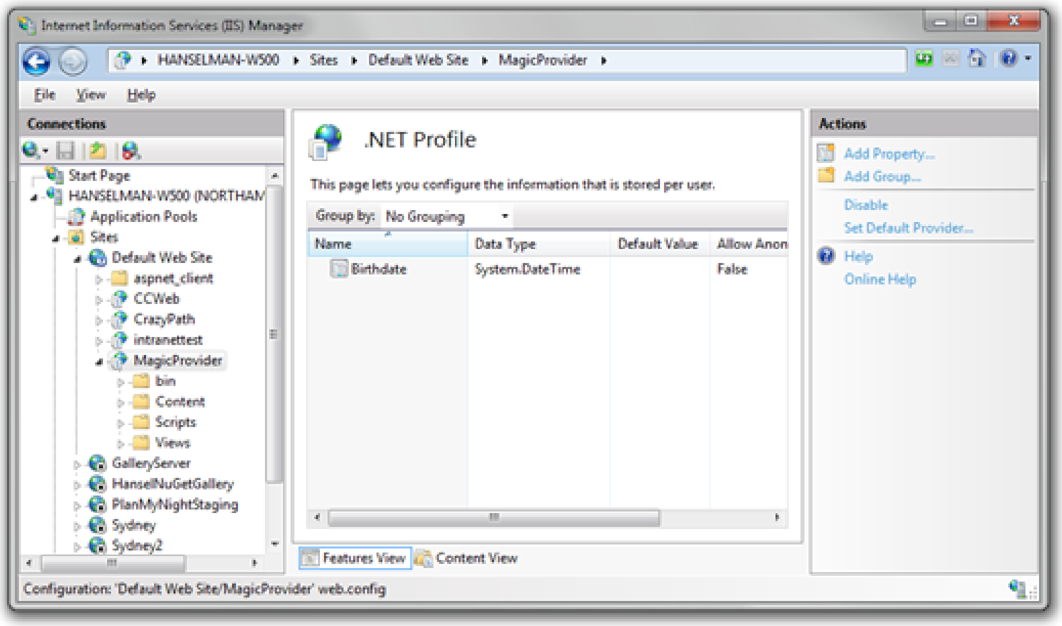
```
<profile inherits="MvcApplication46.Models.MyCustomProfile"
defaultprovider="DefaultProfileProvider">
...</profile>
```

For older website projects, I can add properties in the web.config like this. There are attributes I can use like SettingsAllowAnonymous for custom derive classes in code.

```
..
<properties>
```

```
    <add name="Birthdate" type="System.DateTime" allowanonymous="false" defaultvalue=""
readonly="false" serializeas="String">
</add></properties>
```

Or I can even use IIS7's administration interface to edit the profile details in the web.config. You can have all kinds of profile properties, group them and it's all handled for you.



.NET Profile

This page lets you configure the information that is stored per user.

Group by: No Grouping

Name	Data Type	Default Value	Allow Anon
Birthdate	System.DateTime		False

Configuration: 'Default Web Site/MagicProvider' web.config

If I like, I can ask for the User's Profile (I've got it set for only authenticated users), and set a default as well. I save explicitly, but there is an auto-save option also.

```
if (User.Identity.IsAuthenticated)
{
    var customProfile = HttpContext.Profile as MyCustomProfile;

    DateTime? birthday = customProfile.Birthdate; //Because I made a strongly typed
derived class

    if (!birthday.HasValue) {
        customProfile.Birthdate = new DateTime(1965, 1, 14); //cause that's everyone's
birthday, right?
        customProfile.Save(); //or set autosave if you like.
    }

    ViewBag.Profile = customProfile; //So the View can use it
}
```

At the very end I put the Profile in the ViewBag so it can be accessed from the View. I could also have added just the things I want to a larger ViewModel. Then I can use it later with some sanity checks:

```
@if (ViewBag.Profile != null) {
    @:Hey, your birthday is @ViewBag.Profile.Birthdate.ToString("d")! Congrats.
}
```

Expect to see more cloud-ready things like this in the coming months that'll better position your apps, new and old, to move up to Azure or even down to SQL Compact. Hope this helps. Enjoy.

Related Links

- Jon Galloway - Writing a custom ASP.NET Profile class

About Scott

Scott Hanselman is a former professor, former Chief Architect in finance, now speaker, consultant, father, diabetic, and Microsoft employee. I am a failed stand-up comic, a cornrower, and a book author.

About Newsletter

Sponsored By

Hosting By

Disclaimer: The opinions expressed herein are my own personal opinions and do not represent my employer's view in any way.