



# Tutorial: Creating an ASP.NET Web Page Using SQL Anywhere

**A whitepaper from Sybase iAnywhere**

**Date: June 2010**

**This whitepaper was written in the context of SQL Anywhere 11.  
However, its content may be applicable to previous and future releases**

## Contents

Contents .....	2
Introduction.....	3
Required Software .....	3
Overview .....	3
Installing the SQL Anywhere ASP.NET Providers .....	4
Viewing the SQL Anywhere ASP.NET Providers Schema .....	7
Create an ASP.NET Web Site and Add an Entity Data Model.....	10
Creating Web Controls to Display Data.....	16
Configuring SQL Anywhere ASP.NET Data Providers .....	20
Displaying Private Information for Authenticated Users.....	24
Enabling Dynamic Data .....	34
Removing the SQL Anywhere ASP.NET Providers .....	37
Summary .....	38

## Introduction

This whitepaper demonstrates the use of SQL Anywhere and Visual Studio 2010 to build a database-driven ASP.NET web site. The web application is a simple company directory that displays general employee information in a public page, as well as detailed information in a password-protected secure page. The data in the secure page can only be viewed after a user logs in with the correct credentials. The use of SQL Anywhere ASP.NET providers is highlighted to implement the security mechanism. Topics include information about how to install, configure and set up the SQL Anywhere ASP.NET providers as well as how SQL Anywhere stores application security details in the database. The tutorial also describes how to bind EntityDataSource objects to data-bound server controls and how to integrate the new Dynamic Data feature of the .NET Framework 4.0 in the example web site.

## Required Software

- SQL Anywhere 11.0.1 – Please note that EBF #2436 or later is required to develop a client-side ASP.NET application using Visual Studio 2010.
  - Obtain a free copy of SQL Anywhere Developer Edition from the following location: <http://www.sybase.com/detail?id=1016644>.
  - Download the latest SQL Anywhere EBFs from the following location: <http://downloads.sybase.com/swd/base.do>.
- Microsoft Visual Studio 2010 with .NET Framework 4.0.
- Optional source code (C# and Visual Basic).

## Overview

This tutorial covers the following areas:

- Installing the SQL Anywhere ASP.NET providers.
- Connecting to the demo database and viewing the SQL Anywhere ASP.NET providers' schema using Sybase Central.
- Creating an ASP.NET web site. In this section, we create the web site and add an Entity Data Model (EDM) to the web site from a SQL Anywhere database.
- Creating a GridView web control and bind it to an EntityDataSource object. The GridView control gets its data from the EDM and displays the data on the public page.
- Configuring the SQL Anywhere ASP.NET providers. This section illustrates how to register the providers in the web.config file and how to use the ASP.NET Administration Tool.
- Creating a members-only page. The information on this page will be accessible only to authenticated users.
- Formatting the display of data using the Dynamic Data feature provided by ASP.NET 4.0.

## Installing the SQL Anywhere ASP.NET Providers

The SQL Anywhere ASP.NET providers allow you to build a security mechanism using a SQL Anywhere database as backend storage. They must be installed in order to add the required schema to a designated database for storing and managing confidential user information.

SQL Anywhere includes five providers:

- **Membership Provider:** provides authentication and authorization services.
- **Roles Provider:** provides methods for creating roles, adding users to roles, and deleting roles. Use the roles provider to assign users to groups and manage permissions.
- **Profiles Provider:** provides methods for reading, storing, and retrieving user information such as user preferences.
- **Web Parts Personalization Provider:** provides methods for loading and storing the personalized content and layout of web pages.
- **Health Monitoring Provider:** provides methods for monitoring the status of deployed web applications.

Please refer to the online documentation for more information about SQL Anywhere ASP.NET Providers: [http://dcx.sybase.com/1101en/dbprogramming\\_en11/aspdotnet-providers.html](http://dcx.sybase.com/1101en/dbprogramming_en11/aspdotnet-providers.html).

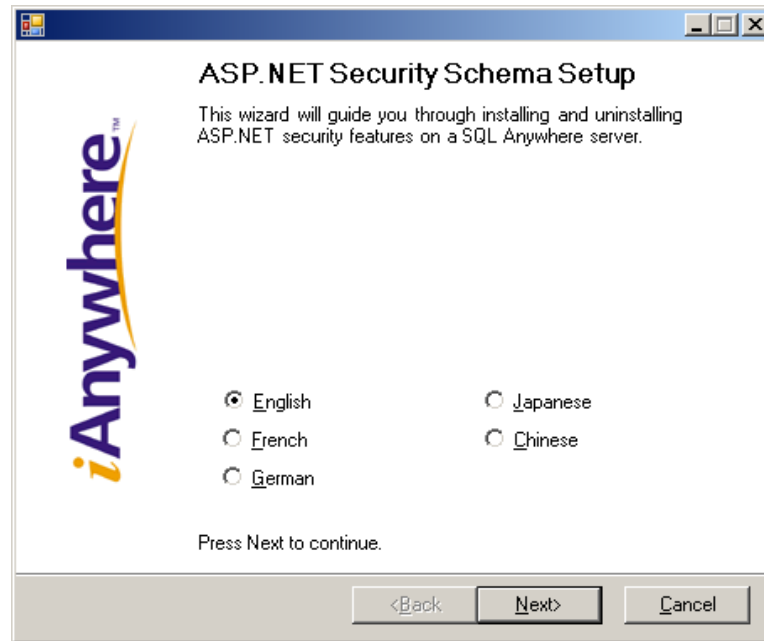
SQL Anywhere also provides a setup wizard to help you add the required security schema to the database. You can either add the SQL Anywhere ASP.NET providers to a new database or to an existing database. For simplicity, we will use the SQL Anywhere demo database ([http://dcx.sybase.com/1101en/saintro\\_en11/sademo-s-3865920.html](http://dcx.sybase.com/1101en/saintro_en11/sademo-s-3865920.html)) in the following procedures.

### Steps

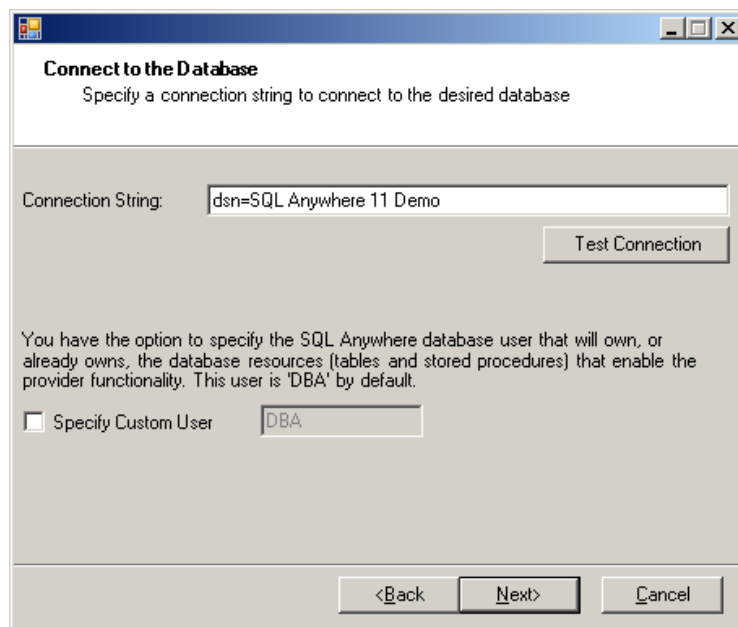
1. Using Windows Explorer, browse to the installation folder where SQL Anywhere is installed (the default installation path is shown here) and locate the following folder:

**C:\Program Files\SQL Anywhere 11\Assembly\v2**

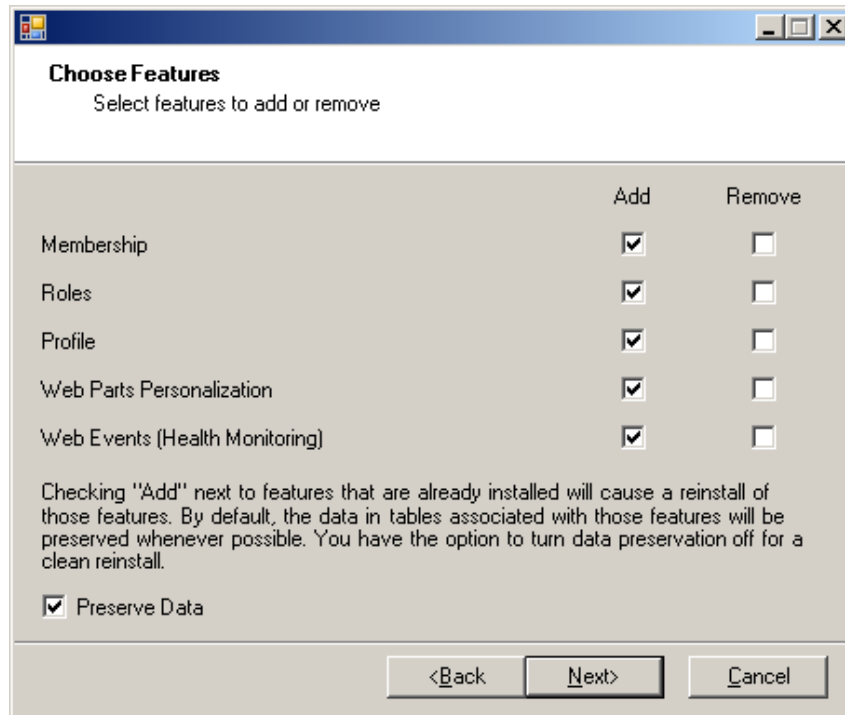
2. Double-click **SASetupAspNet.exe** to run the ASP.NET Security Schema Setup Wizard. Alternatively, you can run SASetupAspNet.exe from a command line prompt. When using the command line to access the SASetupAspNet.exe, use the question mark (-?) argument to display detailed help for configuring the database.



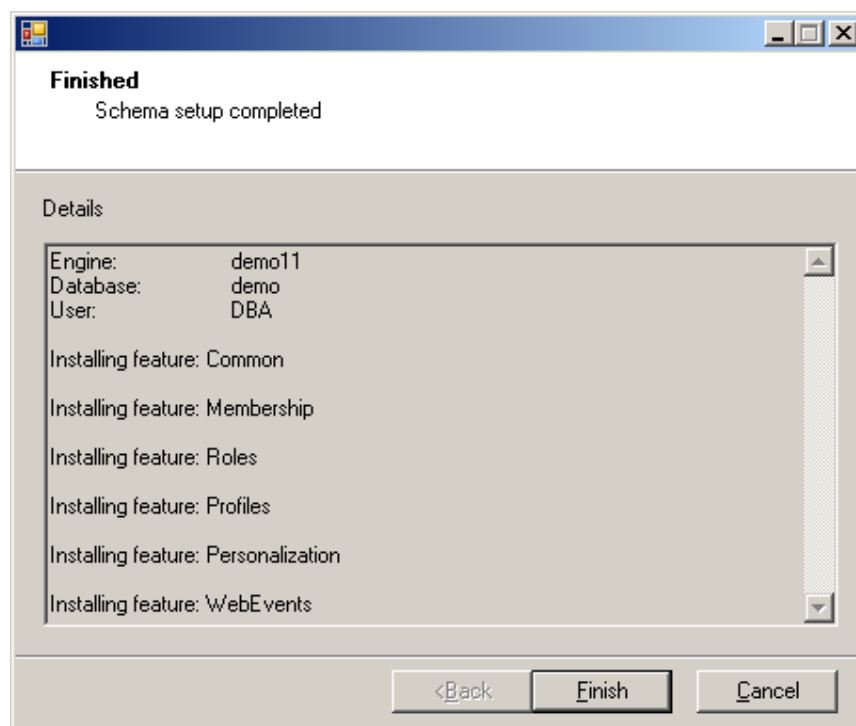
3. Select the desired language and click **Next**.
4. Enter "dsn=SQL Anywhere 11 Demo" for the Connection String and click **Next**. You can also test the database connection at this point.



5. Add the desired features (all in this case) and click **Next**. Notice that you can also remove features already installed.



6. Verify the installing details and click **Finish**.

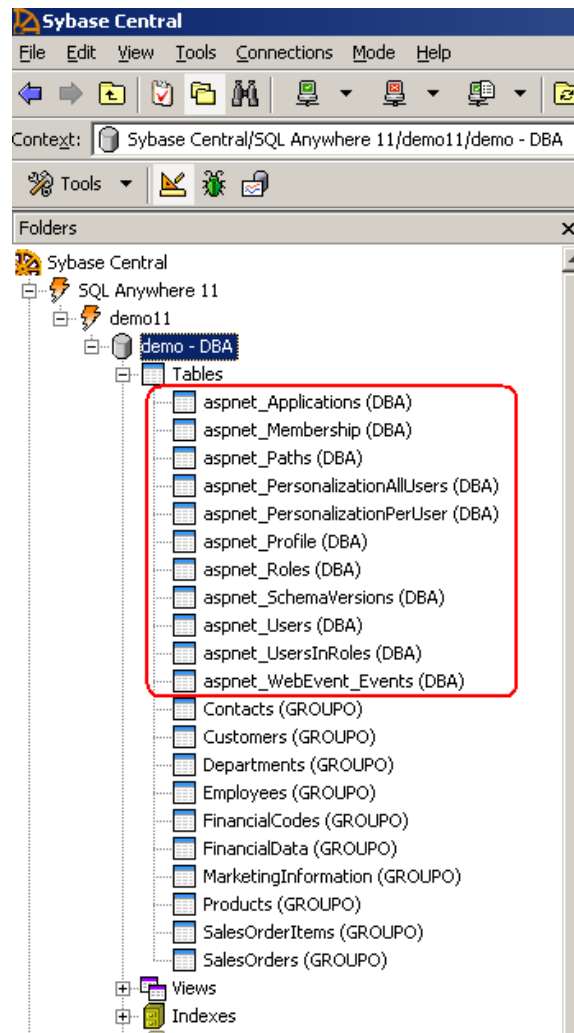


## Viewing the SQL Anywhere ASP.NET Providers Schema

The necessary tables and stored procedures are added to the demo database by the wizard in the previous section, all with the prefix “aspnet\_”. This section shows the changes made to provide an overview of how SQL Anywhere stores confidential login information and application security details.

### Steps

1. Open Sybase Central and connect to the demo database. Using the Folders View, expand **Tables** to see the tables created by the wizard. As shown below, notice that new tables prefixed with “aspnet\_” are added to the demo database. These are the tables that hold the security data used in the web site.



2. Click the table **aspnet\_Membership** to view its columns. This table stores the membership information of the users, including userId (primary key), password, security

question, lastLoginDate and all the necessary information to perform web site authentication. Password format is hashed by default, ensuring that the information saved in the database is secure.

aspnet_Membership (DBA)										
	PKey	Name	ID	Object ID	Data Type	Size	Scale	Cmp.	Nulls	Unique
1	<input type="checkbox"/>	ApplicationId	1	3523	uniqueidentifier			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	<input checked="" type="checkbox"/>	UserId	2	3524	uniqueidentifier			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	Password	3	3525	nvarchar	128		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	<input type="checkbox"/>	PasswordFormat	4	3526	integer			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	<input type="checkbox"/>	PasswordSalt	5	3527	nvarchar	128		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	<input type="checkbox"/>	MobilePIN	6	3528	nvarchar	16		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
7	<input type="checkbox"/>	Email	7	3529	nvarchar	256		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
8	<input type="checkbox"/>	LoweredEmail	8	3530	nvarchar	256		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
9	<input type="checkbox"/>	PasswordQuestion	9	3531	nvarchar	256		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
10	<input type="checkbox"/>	PasswordAnswer	10	3532	nvarchar	128		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
11	<input type="checkbox"/>	IsApproved	11	3533	bit			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
12	<input type="checkbox"/>	IsLockedOut	12	3534	bit			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
13	<input type="checkbox"/>	CreateDate	13	3535	datetime			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
14	<input type="checkbox"/>	LastLoginDate	14	3536	datetime			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
15	<input type="checkbox"/>	LastPasswordCha...	15	3537	datetime			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
16	<input type="checkbox"/>	LastLockoutDate	16	3538	datetime			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
17	<input type="checkbox"/>	FailedPasswordAt...	17	3539	integer			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
18	<input type="checkbox"/>	FailedPasswordAt...	18	3540	datetime			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
19	<input type="checkbox"/>	FailedPasswordA...	19	3541	integer			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
20	<input type="checkbox"/>	FailedPasswordA...	20	3542	datetime			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
21	<input type="checkbox"/>	Comment	21	3543	long nvarchar			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

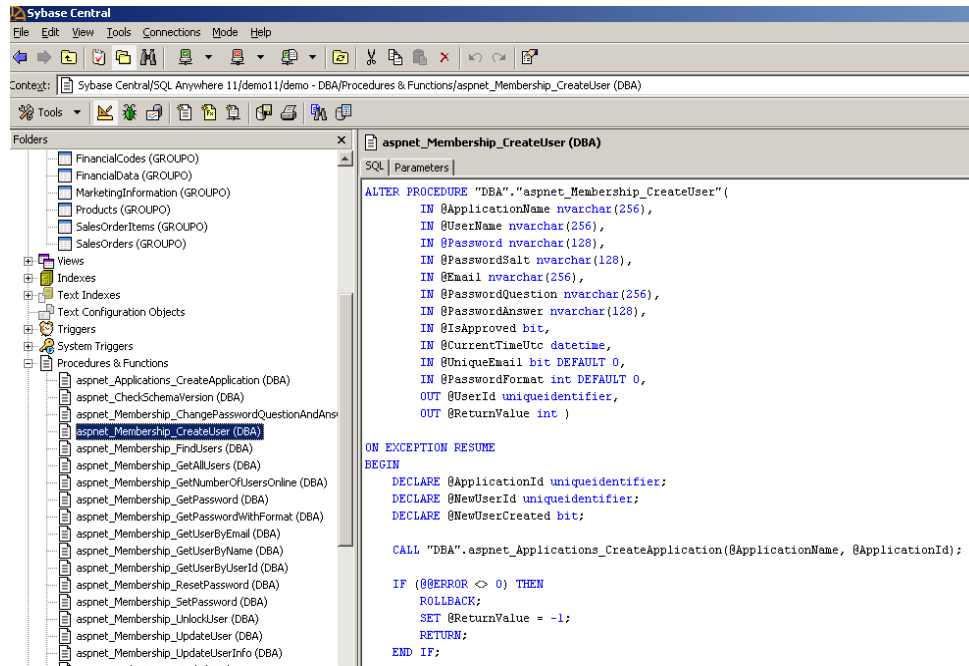
- Click the **Constraints** tab and notice that the proper key constraints on the columns are also set up by the wizard.

Sybase Central					
File Edit View Tools Connections Mode Help					
Context: Sybase Central/SQL Anywhere 11/demo11/demo - DBA/Tables/aspnet_Membership (DBA)					
aspnet_Membership (DBA)					
	Name	Constraint Type	Unique	Columns	Definition
	ASA178	Primary key constraint	Yes	UserId	
	FK_Application...	Foreign key constraint	No	ApplicationId	
	FK_UserId	Foreign key constraint	No	UserId	

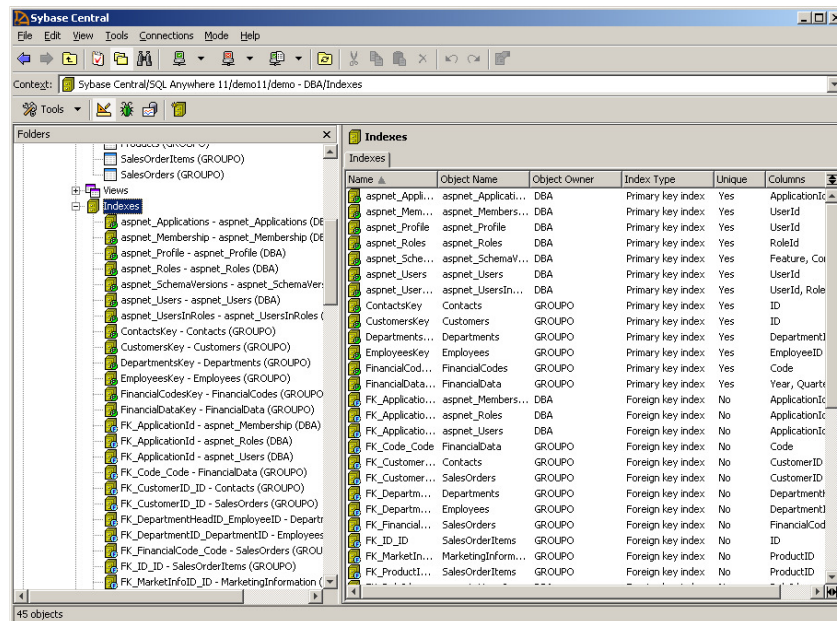
- Expand **Procedures and Functions** in the folder view. Notice the wizard added



procedures for the operations available in each provider. Click the **aspnet\_Membership\_CreateUser** procedure to view the SQL code. This is the procedure used by the Membership Provider when a new user record is inserted into the database.



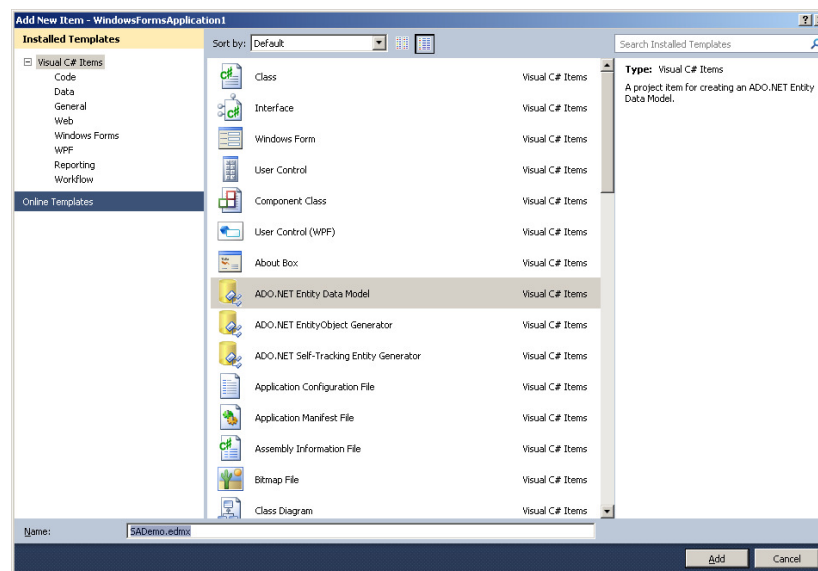
- Expand **Indexes** and notice that indices are also automatically created for the added tables to improve performance.



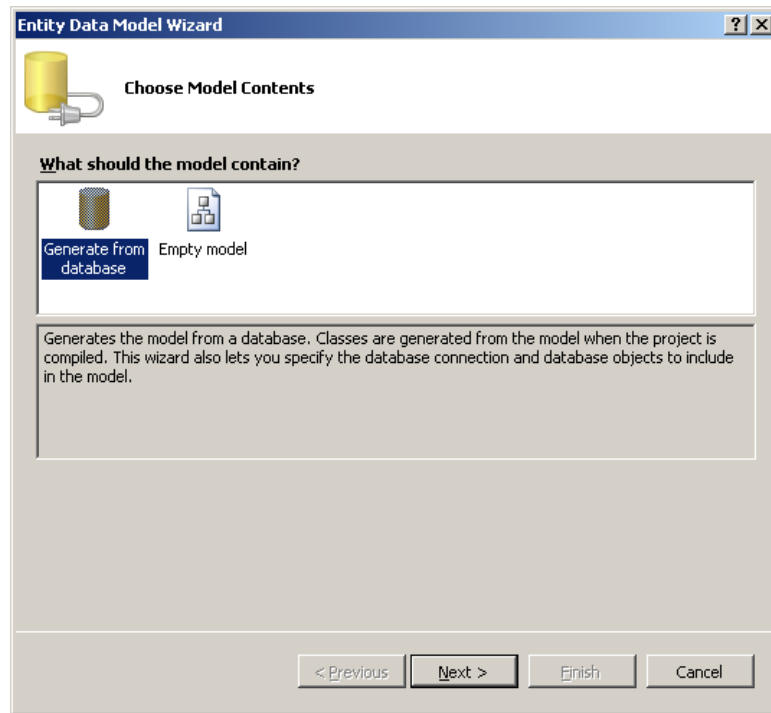
## Create an ASP.NET Web Site and Add an Entity Data Model

Any SQL Anywhere database profile defined in Visual Studio can be used to create a new entity data model (EDM). Follow the steps below to create an ASP.NET web site and add the SQL Anywhere demo database as an EDM to it.

1. Start Visual Studio 2010.
2. Select **File > New > Web Site**.
3. From the 'New Web Site' dialog, choose **ASP.NET Web Site**.
4. Save the web site as 'Sample\_asp.net' in a known location of your choice.
5. Right-click the web site project in the Solution Explorer, click **Add New Item > ADO.NET Entity Data Model** from the popup menu.
6. In the **Name** field, type **SADemo.edmx**. Click **Add**.
7. Click **Yes** to add the ADO.NET Entity Model to the folder App\_Code as recommended.



8. The Entity Data Model Wizard appears. Select **Generate from database** and click **Next**.



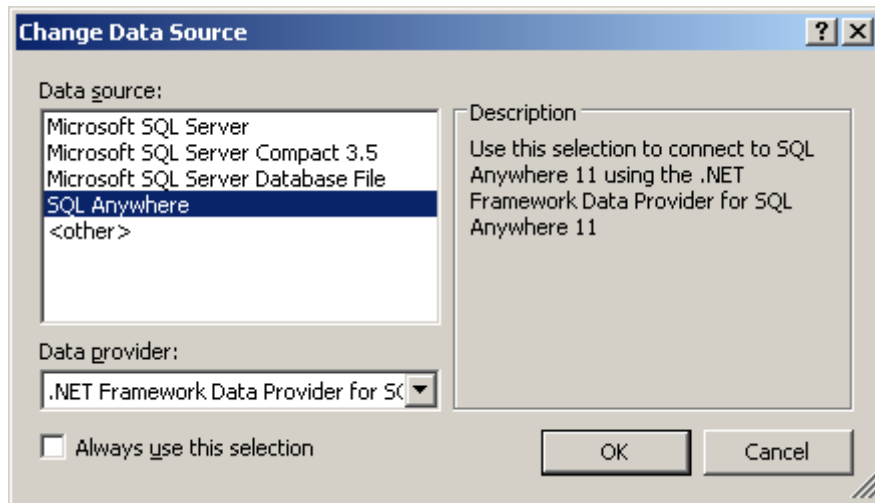
9. Click **New Connection**. Click the **Change** button beside Data Source. In the **Data source** list, select **SQL Anywhere** then click **OK**.

➤ If SQL Anywhere does not appear in the **Data source** list, please ensure that the SQL Anywhere integration components for Visual Studio are properly installed. To install the integration components:

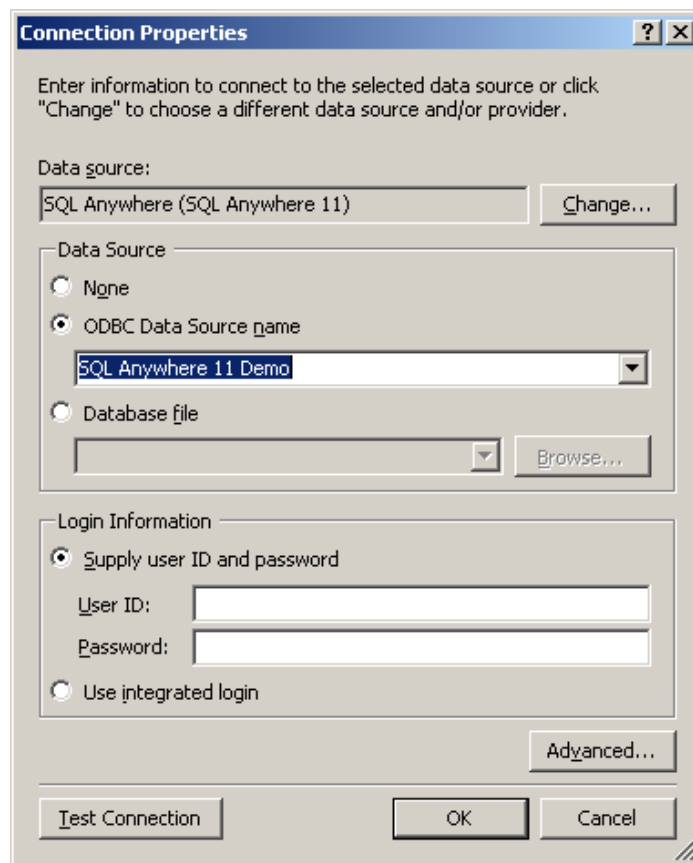
1. Close Visual Studio 2010.
2. Open a Command Prompt and change to this directory:

**C:\Program Files\SQL Anywhere 11\Assembly\v2**

3. Then execute the following command: **SetupVSPackage.exe -i**




10. Click **ODBC Data Source name** and select **SQL Anywhere 11 Demo**. Click **OK**.



11. Change the name of the entity connection string to “SAEntities” and click **Next**.

**Entity Data Model Wizard**

 **Choose Your Data Connection**

**Which data connection should your application use to connect to the database?**

SQL Anywhere.demo11 New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Entity connection string:

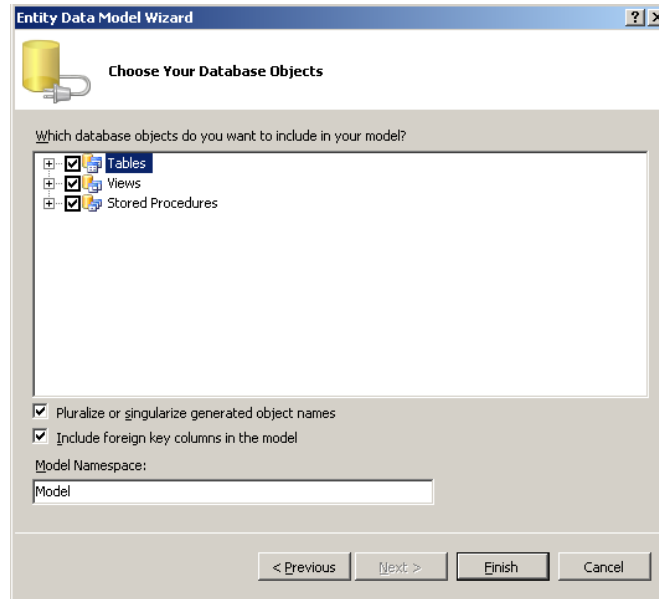
```
metadata=res://*/App_Code.Model.csdl|res://*/App_Code.Model.ssdl|res://*/App_Code.Model
.msl;provider=iAnywhere.Data.SQLAnywhere;provider connection
string='DataSourceName="SQL Anywhere 11 Demo"'
```

☒ Save entity connection settings in Web.Config as:

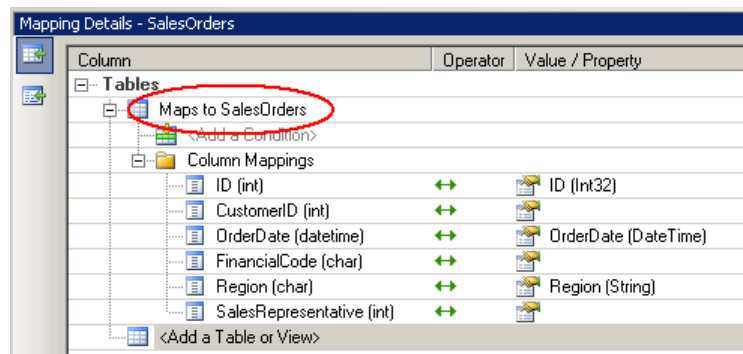
SAEntities

< Previous Next > Finish Cancel

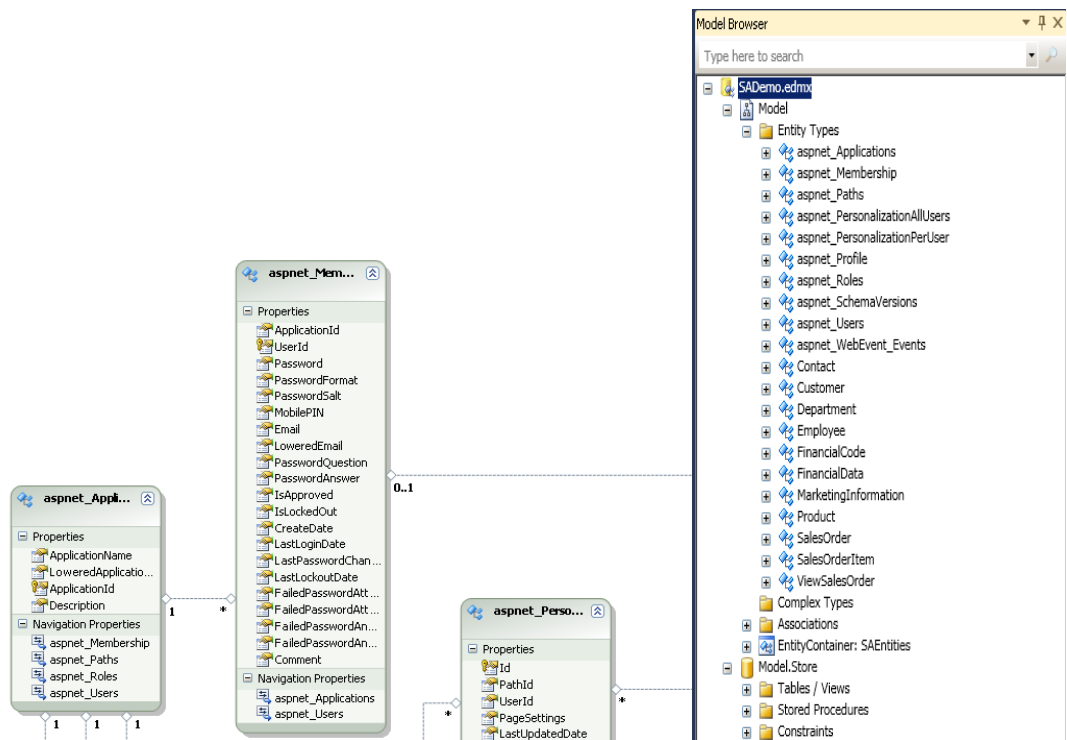
12. Include all database objects in the model and click **Finish**. In practice, your EDM would only include the objects required by your application. For simplicity, we're adding all the objects in this tutorial.



13. Open **SADemo.edmx** file, a visual representation of the model appears in the Entity Designer. Right-click the SalesOrders entity and select **Table Mapping** to view the mapping details. This diagram illustrates that the entity is properly mapped to the database schema.



14. You can also use the Entity Designer to view the tables that are added to the demo database by the ASP.NET providers. Right-click the empty space in the Entity Designer and select **Model Browser** to obtain an overview of all the entity types in the Model namespace.

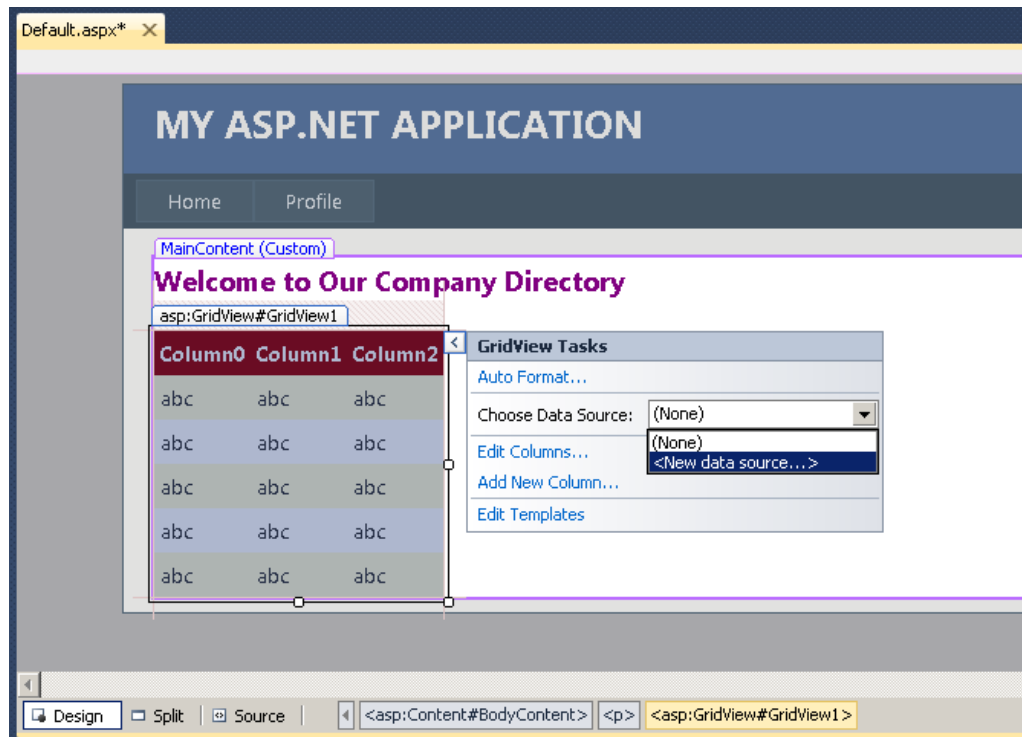


## Creating Web Controls to Display Data

Now that we have set up the ASP.NET providers and the Entity Model used in the web site, we will add a GridView control to the default page and bind it to an EntityDataSource. This page will be the public area of the web site and all users are allowed to view the information without any authentication.

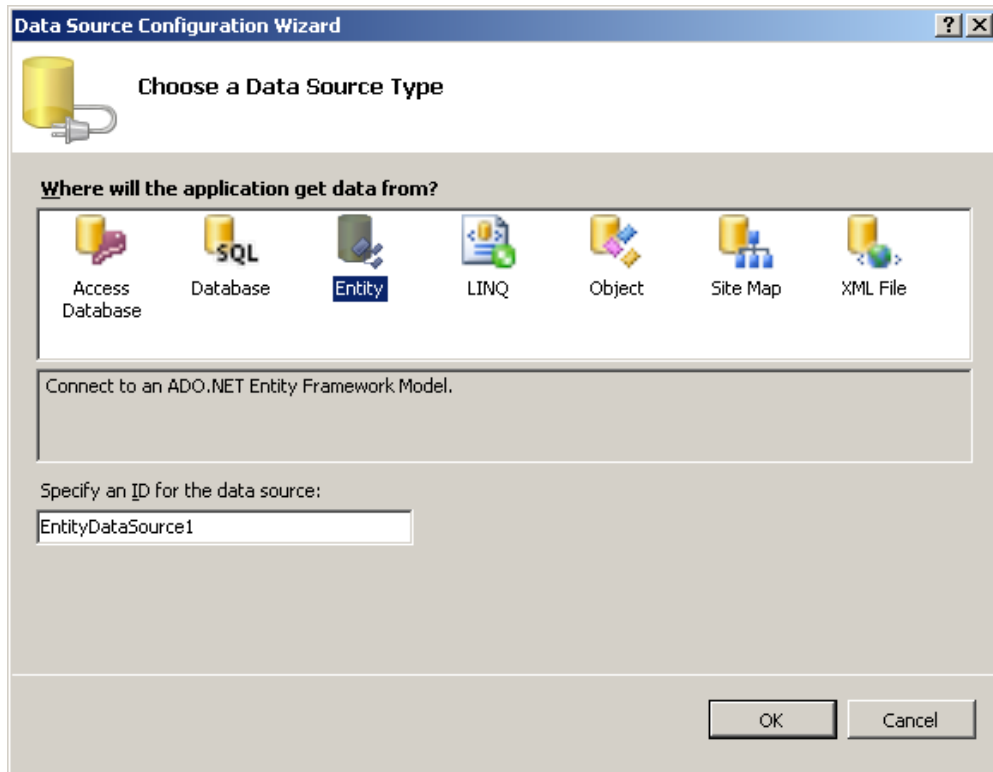
### Steps

1. Open Default.aspx, switch to Design view and drag a **GridView** control from the toolbox to replace the default body content.
2. In the **GridView Tasks** menu, open the drop down list **Choose Data Source** and select **<New data source...>**

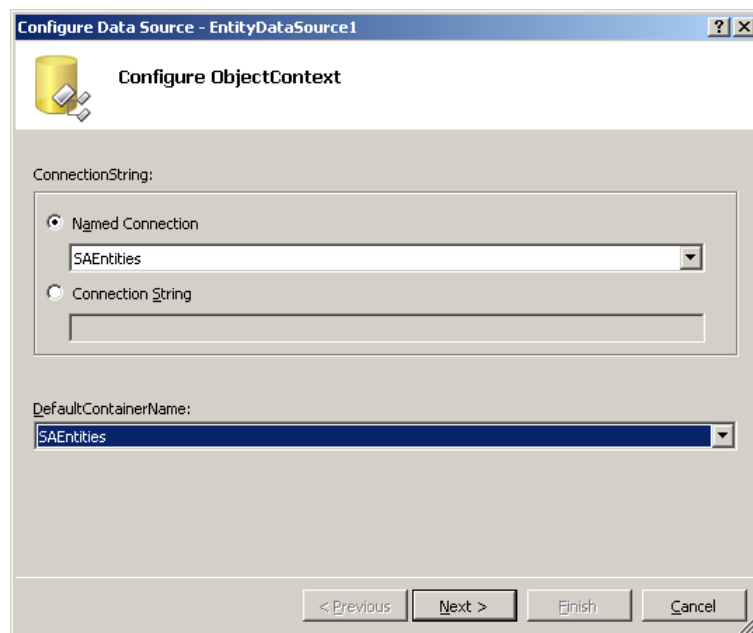


3. Choose **Entity** and click **OK**.





4. The Configure Data Source wizard appears. In the **Named Connection** drop down list, select **SAEntities** and click **Next**.



5. Select **Employees** from the EntitySetName drop down list. Select EmployeeID, ManagerID, Surname, GivenName, DepartmentID, State, Country and Phone as the public information to be viewed by anyone. Click **Finish**.

**Configure Data Source - EntityDataSource1**

**Configure Data Selection**

EntitySetName: Employees

EntityTypeFilter: (None)

Select:

<input type="checkbox"/> Select All (Entity Value)	<input type="checkbox"/> City	<input type="checkbox"/> Salary
<input checked="" type="checkbox"/> EmployeeID	<input checked="" type="checkbox"/> State	<input type="checkbox"/> StartDate
<input checked="" type="checkbox"/> ManagerID	<input checked="" type="checkbox"/> Country	<input type="checkbox"/> TerminationDate
<input checked="" type="checkbox"/> Surname	<input type="checkbox"/> PostalCode	<input type="checkbox"/> BirthDate
<input checked="" type="checkbox"/> GivenName	<input checked="" type="checkbox"/> Phone	<input type="checkbox"/> BeneficialInterest
<input checked="" type="checkbox"/> DepartmentID	<input type="checkbox"/> Status	<input type="checkbox"/> BeneficialInterest
<input type="checkbox"/> Street	<input type="checkbox"/> SocialSecurityNumber	<input type="checkbox"/> BeneficialInterest

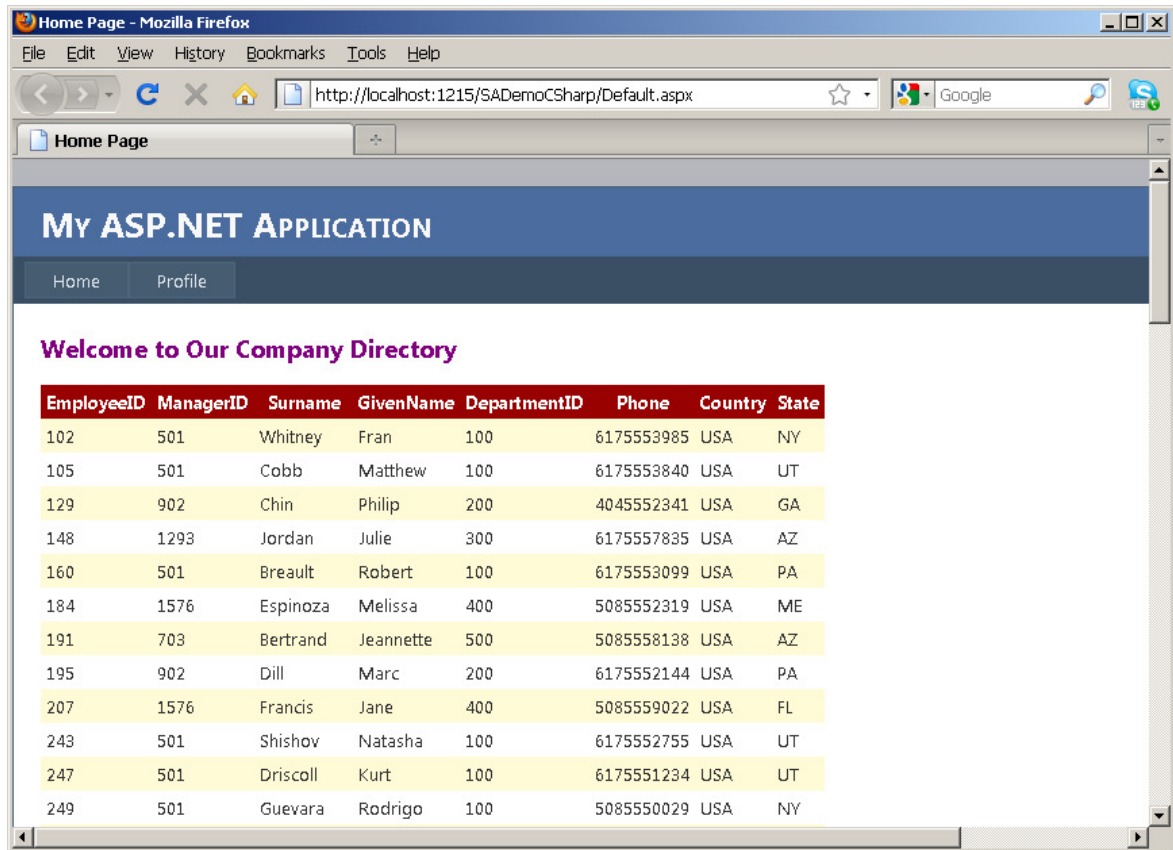
☐ Enable automatic inserts

☐ Enable automatic updates

☐ Enable automatic deletes

< Previous   Next >   Finish   Cancel

6. Hit **F5** to run/debug the application. Enable debugging in the Web.config file if desired (you may see a dialog asking for that). The default page displays the selected columns from the Employee table in the demo database. Note that this information is available to anyone who accesses the page. Close the page and stop the application.



Home Page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:1215/SADemoCSsharp/Default.aspx

Home Page

## My ASP.NET APPLICATION

Home Profile

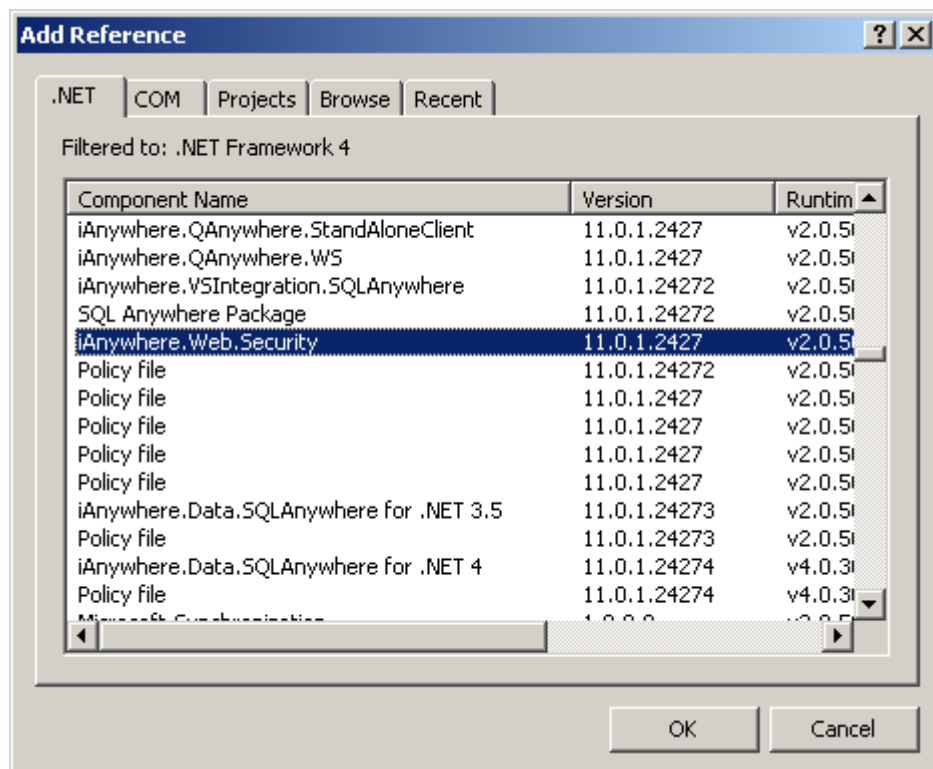
### Welcome to Our Company Directory

EmployeeID	ManagerID	Surname	GivenName	DepartmentID	Phone	Country	State
102	501	Whitney	Fran	100	6175553985	USA	NY
105	501	Cobb	Matthew	100	6175553840	USA	UT
129	902	Chin	Philip	200	4045552341	USA	GA
148	1293	Jordan	Julie	300	6175557835	USA	AZ
160	501	Breault	Robert	100	6175553099	USA	PA
184	1576	Espinoza	Melissa	400	5085552319	USA	ME
191	703	Bertrand	Jeannette	500	5085558138	USA	AZ
195	902	Dill	Marc	200	6175552144	USA	PA
207	1576	Francis	Jane	400	5085559022	USA	FL
243	501	Shishov	Natasha	100	6175552755	USA	UT
247	501	Driscoll	Kurt	100	6175551234	USA	UT
249	501	Guevara	Rodrigo	100	5085550029	USA	NY

## Configuring SQL Anywhere ASP.NET Data Providers

Now we want to add the secure page to the web site and enable the appropriate security settings. The application also must be configured to use the SQL Anywhere ASP.NET providers. The following steps show you how to register the SQL Anywhere ASP.NET providers and how to use the ASP.NET Administration Tool to manage user information.

1. Right-click the project in the Solution Explorer and select **Add Reference**. Select the .NET tab and choose **iAnywhere.Web.Security** from the list to add a reference to the iAnywhere.Web.Security assembly to your web site.



2. Open the **web.config** file and add the connection string to the **<connectionStrings>** element (changes are highlighted):

```
<connectionStrings>
  <!--Register the connection string-->
  <add name="MyConnectionString"
    connectionString="dsn=SQL Anywhere 11 Demo"
    providerName="iAnywhere.Data.SQLAnywhere" />
  <add name="SAEntities"
    connectionString="metadata=res://*/App_Code.SADemo.csdl|res://*/App_Code.SADemo.ssdl|res://*/App_Code.SADemo.msl;provider=iAnywhere.Data.SQLAnywhere;provider connection
    string='UserID=dba;Password=sql;DataSourceName="SQL Anywhere 11 Demo";'
    providerName="System.Data.EntityClient" />
</connectionStrings>
```

3. Add an entry for each provider and add **<machineKey validation="SHA1">** to the **<system.web>** element. Add the name of the SQL Anywhere ASP.NET provider to the **"defaultProvider"** attribute in the application (changes are highlighted):

```
<system.web>
  <machineKey validation="SHA1" />
  <compilation debug="true" strict="false" explicit="true" targetFramework="4.0">
    <assemblies>
      <add assembly="System.Security, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A" />
      <add assembly="System.Data.Entity, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089" />
      <add assembly="System.Data.Entity.Design, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089" />
    </assemblies>
    <buildProviders>
      <add extension=".edmx"
        type="System.Data.Entity.Design.AspNet.EntityDesignerBuildProvider" />
    </buildProviders>
  </compilation>
  <authentication mode="Forms">
    <forms loginUrl="~/Account/Login.aspx" timeout="2880" />
  </authentication>
```

```
<membership defaultProvider="SAMembershipProvider">
  <providers>
    <clear />
    <add name="SAMembershipProvider"
      type="iAnywhere.Web.Security.SAMembershipProvider"
      connectionStringName="MyConnectionString"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="false"
      requiresUniqueEmail="false"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="6"
      minRequiredNonalphanumericCharacters="0"
      passwordAttemptWindow="10"
      applicationName="/"
      passwordFormat="hashed" />
  </providers>
</membership>
```

```
<profile defaultProvider="SAProfileProvider">
  <providers>
    <clear />
    <add name="SAProfileProvider"
      type="iAnywhere.Web.Security.SAProfileProvider"
      connectionStringName="MyConnectionString"
      applicationName="/"
      commandTimeout="30" />
  </providers>
</profile>
```

```
<roleManager enabled="true" defaultProvider="SARoleProvider">
  <providers>
    <clear />
```

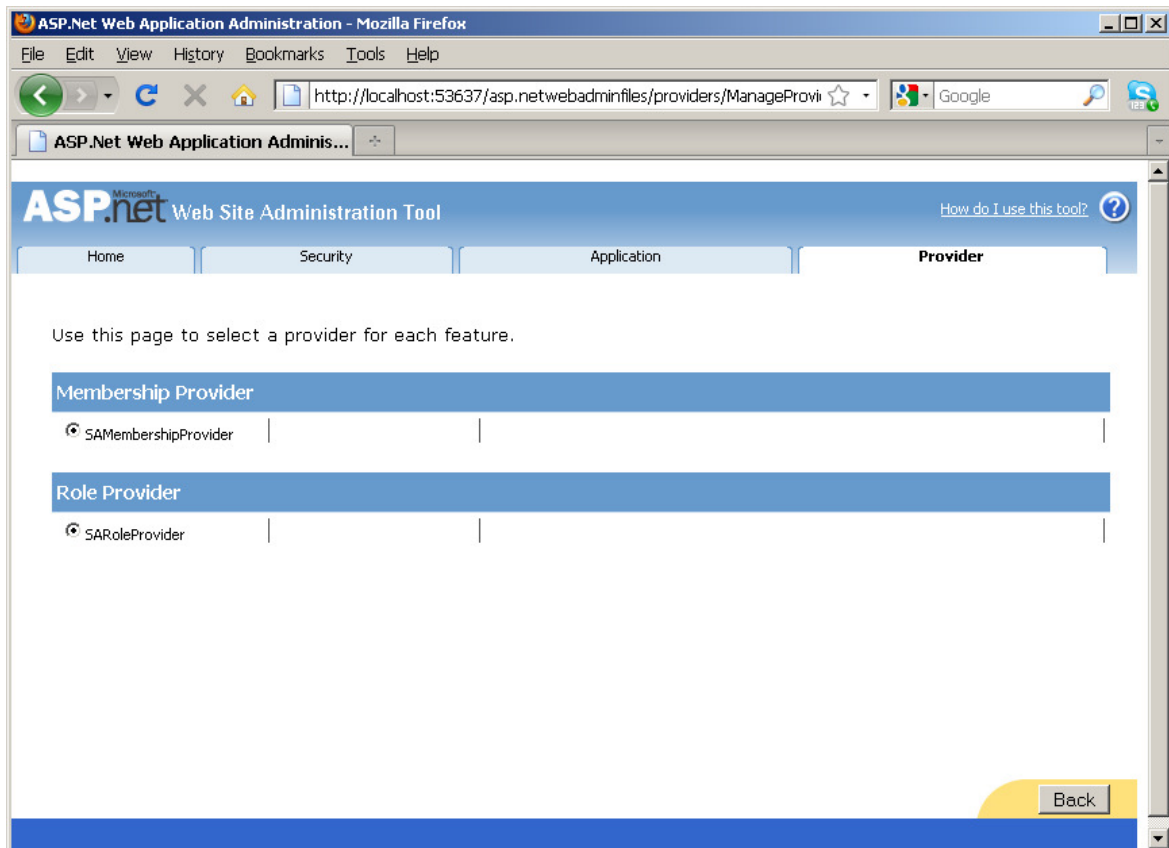
```

    <add connectionStringName="MyConnectionString"
        applicationName="/"
        commandTimeout="30"
        name="SARoleProvider"
        type="iAnywhere.Web.Security.SARoleProvider" />
  </providers>
</roleManager>
</system.web>

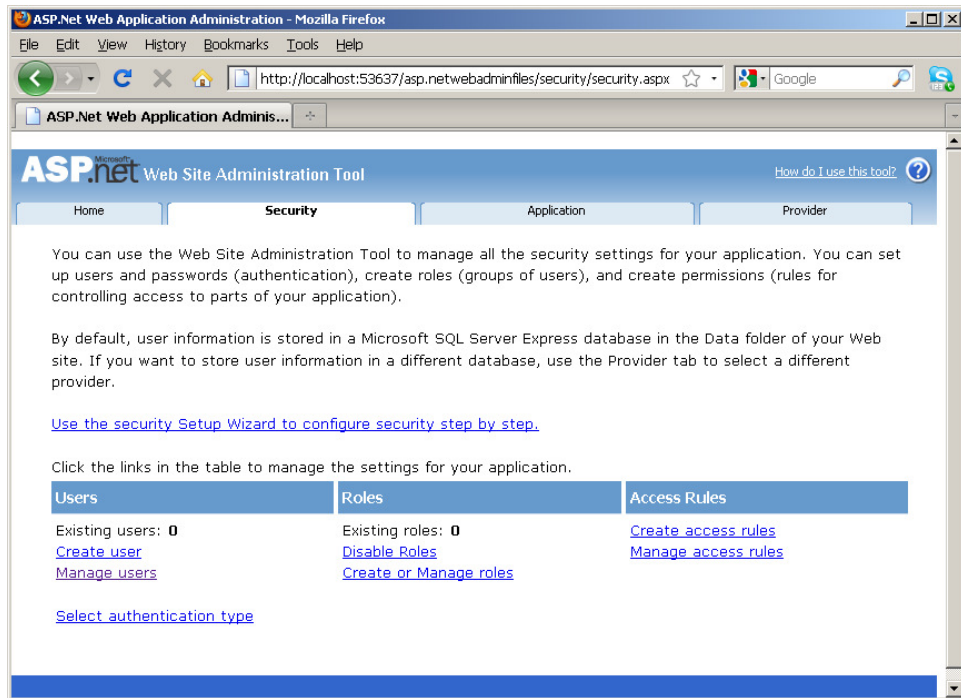
```

For more information on how to configure the SQL Anywhere ASP.NET Providers, please refer to the online API documentation: [http://dcx.sybase.com/1101en/dbprogramming\\_en11/pg-aspdotnet-providers-s-3878926.html](http://dcx.sybase.com/1101en/dbprogramming_en11/pg-aspdotnet-providers-s-3878926.html).

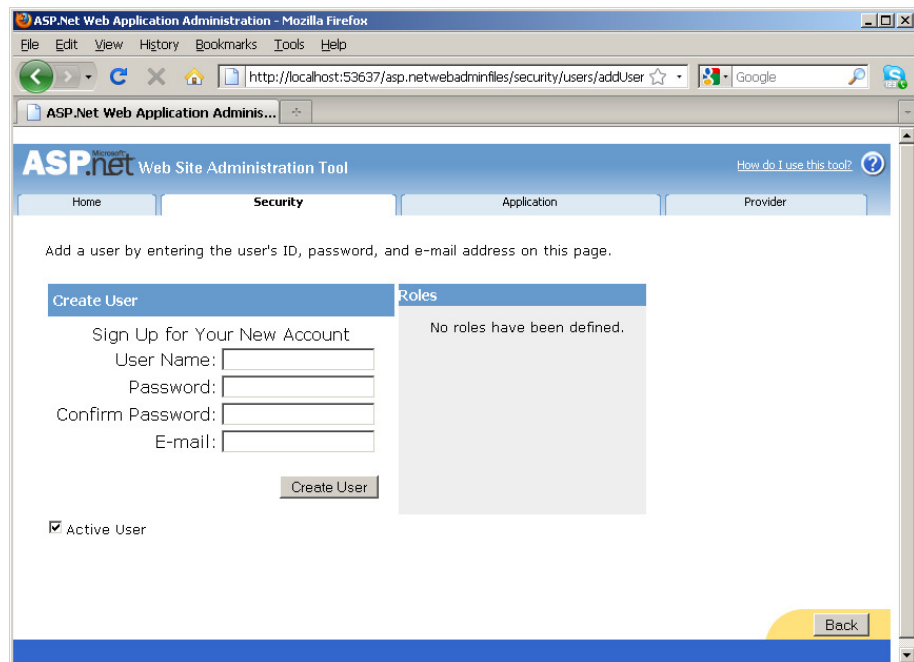
- Now you can use the Web Site Administration Tool to create new users. To use this tool, save the changes and then click **ASP.NET Configuration** on the **Website** menu in Visual Studio 2010. Click **Provider Configuration > Select a different provider for each feature (advanced)**, you'll see that SAMembershipProvider and SARoleProvider are used.



- Under the **Security** tab, there are a number of tools provided for you to perform web site administration tasks including enabling roles and creating access rules for different users.



6. Click **Security>Create user**, you will then be able to create new user accounts by completing the web form.

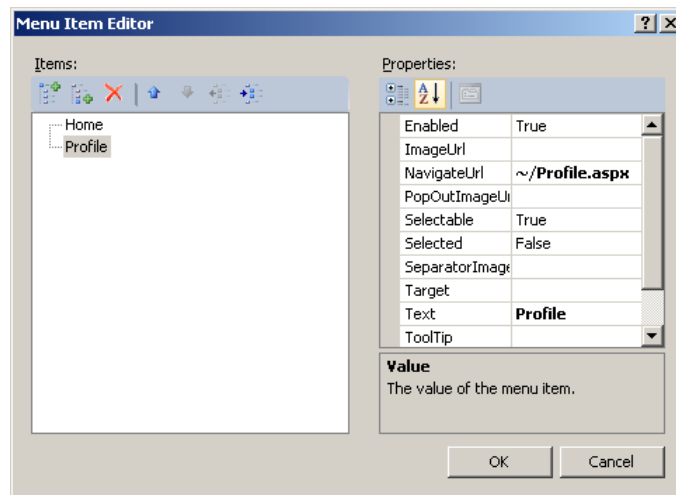


7. Close the ASP.NET Web Application Administration page.

## Displaying Private Information for Authenticated Users

Now we can create a second page that displays a detailed employee profile that's only available to an employee who's logged in. For simplicity, we will register a user whose information already exists in the Employee table (we'll use the first name as the user name). We will rename the built-in About.aspx file and customize it to be an employee profile page that displays private information. Visual Studio also provides an enhanced ASP.NET web site template that includes a built-in account folder already configured with basic membership functionality. We will take advantage of these provided templates to set up the sample application in the following steps.

1. Right-click file **About.aspx** in solution explore and click **Rename**. Rename the file to **Profile.aspx**. Notice that the corresponding code file is renamed automatically as well.
2. Now we want to edit the menu bar in the master page so that the menu item name is consistent with the file name. Open the file site.master and switch to Design view. Locate the navigation menu and click the > button beside it. Select **Edit Menu Items**, and Replace the URL of the second menu item to “~/Profile.aspx” in the Menu Item Editor and change the Text property to “Profile”. Click **OK**.

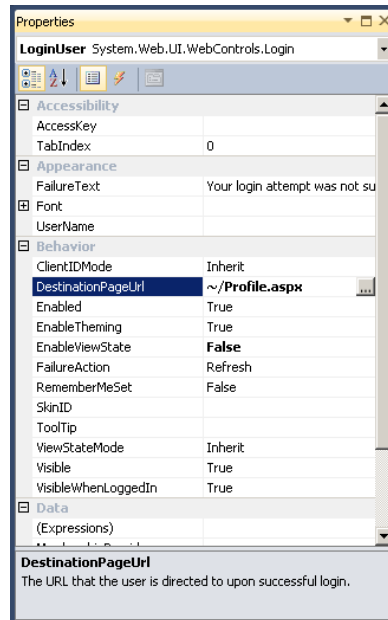


If you switch to Source view, the markup for the navigation menu should look like this:

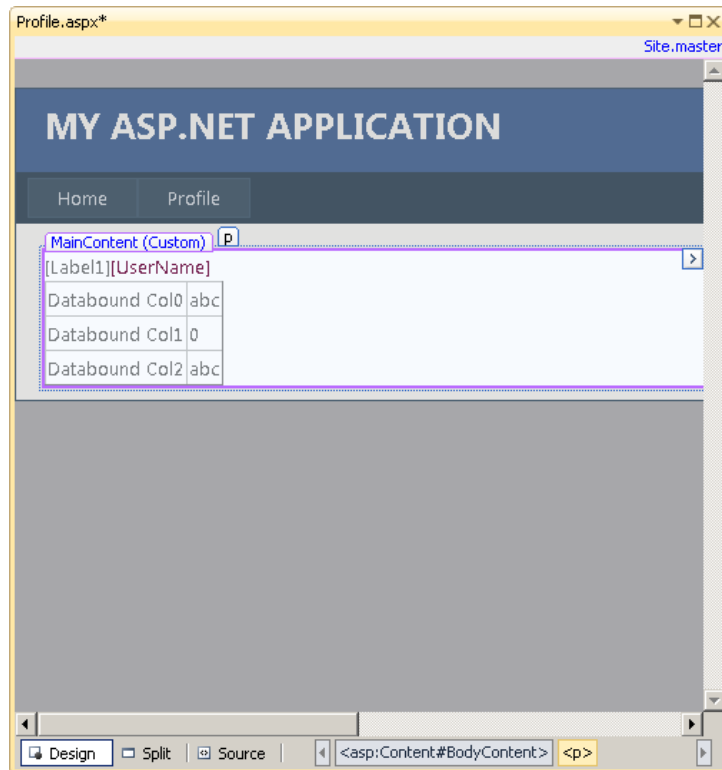
```
<asp:Menu ID="NavigationMenu"
    runat="server"
    CssClass="menu"
    EnableViewState="false"
    IncludeStyleBlock="false"
    Orientation="Horizontal">
  <Items>
    <asp:MenuItem NavigateUrl="~/Default.aspx" Text="Home"/>
    <asp:MenuItem NavigateUrl="~/Profile.aspx" Text="Profile"/>
  </Items>
</asp:Menu>
```



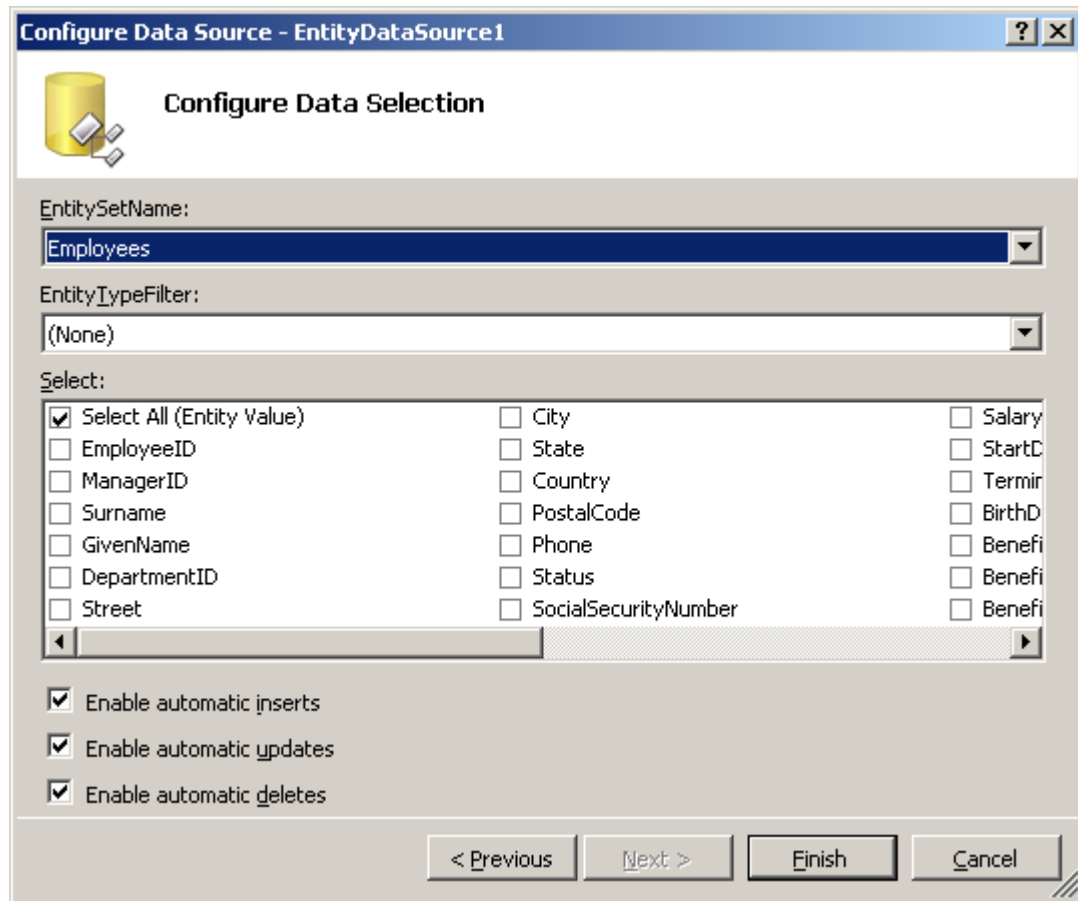
- Now we will edit the default login template so that it automatically redirects the user to this Profile page after logging in successfully. Open Account/Login.aspx, right-click the Login control in the Design view and click **Properties**. Select Profile.aspx for the DestinationPageURL property.



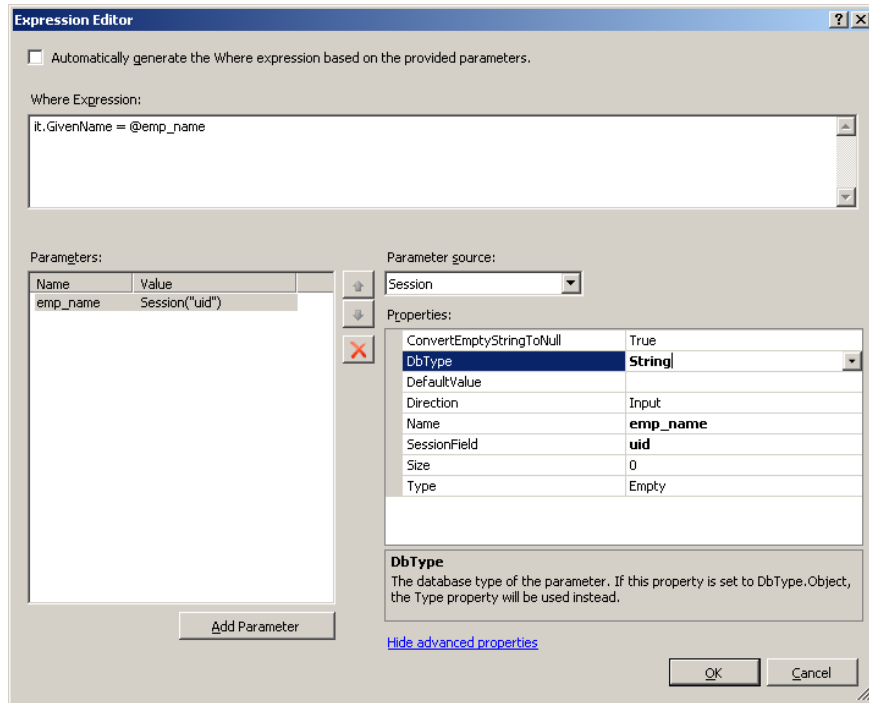
- Open Profile.aspx and delete the default lines in the MainContent place holder. Then drag a Label control, a LoginName control and a DetailsView control and drop them into the MainContent place holder.



5. In the **DetailsView Tasks** menu, open the drop down list **Choose Data Source** and select **<New data source...>**. We'll add a new EntityDataSource as we did for the public page.
6. Choose **Entity** and click **OK**.
7. From the **Named Connection** drop down list, select **SAEntities** and click **Next**.
8. Select **Employees** from the EntitySetName drop down list. This time we will select all the columns in the table. Click **Finish**.



9. A new **EntityDataSource** appears below the DetailsView control. Right-click it and select **Properties**.
10. Click the button beside **Where** on the Properties menu to open the Expression Editor. Click **Add Parameter** and type “**emp\_name**” in the name field. Select **Session** from the Parameter Source drop down list and enter “**uid**” in the Session Field. In the Where Expression Field, enter “**it.GivenName=@emp\_name**”.
11. Click **Show advanced properties** and select **String** from the DbType drop down list. Click **OK** to close the Expression Editor.



12. Insert the following code to the Page\_Load event of the Profile.aspx page:

[C#]

```
//check to see if the user has logged in
if (User.Identity.IsAuthenticated)
{
    //retrieve the user name and save it as a session variable
    var user = User.Identity;
    Session["uid"] = user.Name;

    //display login information
    Label1.Text = "You are logged in as ";
}

else {
    Session["uid"] = "";
    Label1.Text = "Hello Guest! Please sign in to view your profile.";
}
```

[VB]

```
'check to see if the user has logged in
If User.Identity.IsAuthenticated Then

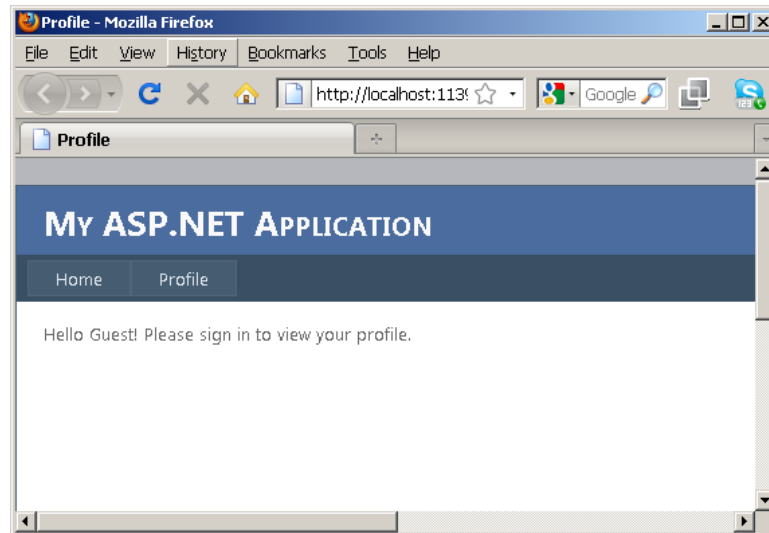
    'retrieve the user name and save it as a session variable
    Dim user As MembershipUser = Membership.GetUser()
    Session("uid") = user.UserName.ToString
    Label1.Text = "You are logged in as "
```

```

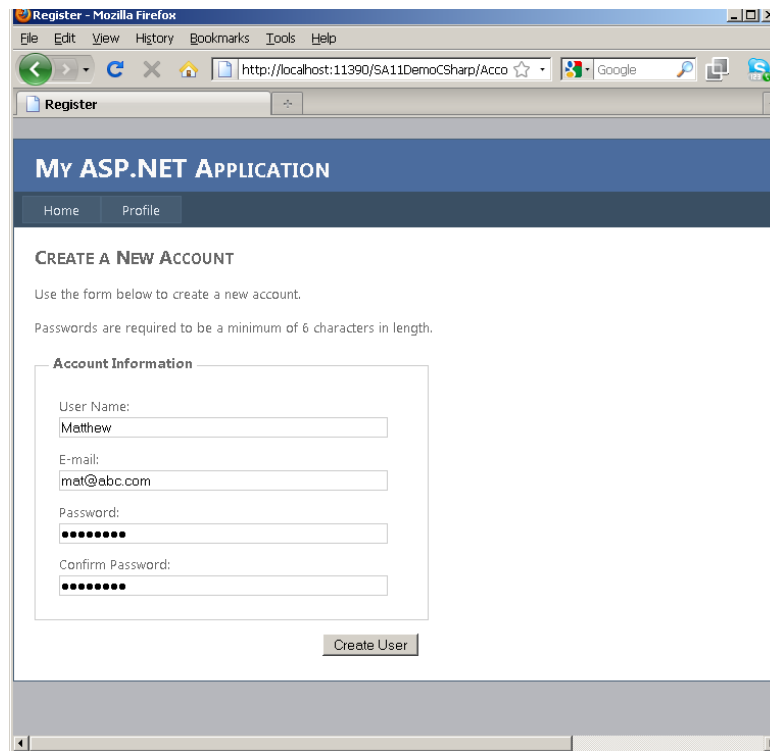
Else
    Session("uid") = ""
    Label11.Text = "Hello Guest! Please sign in to view your profile."
End If

```

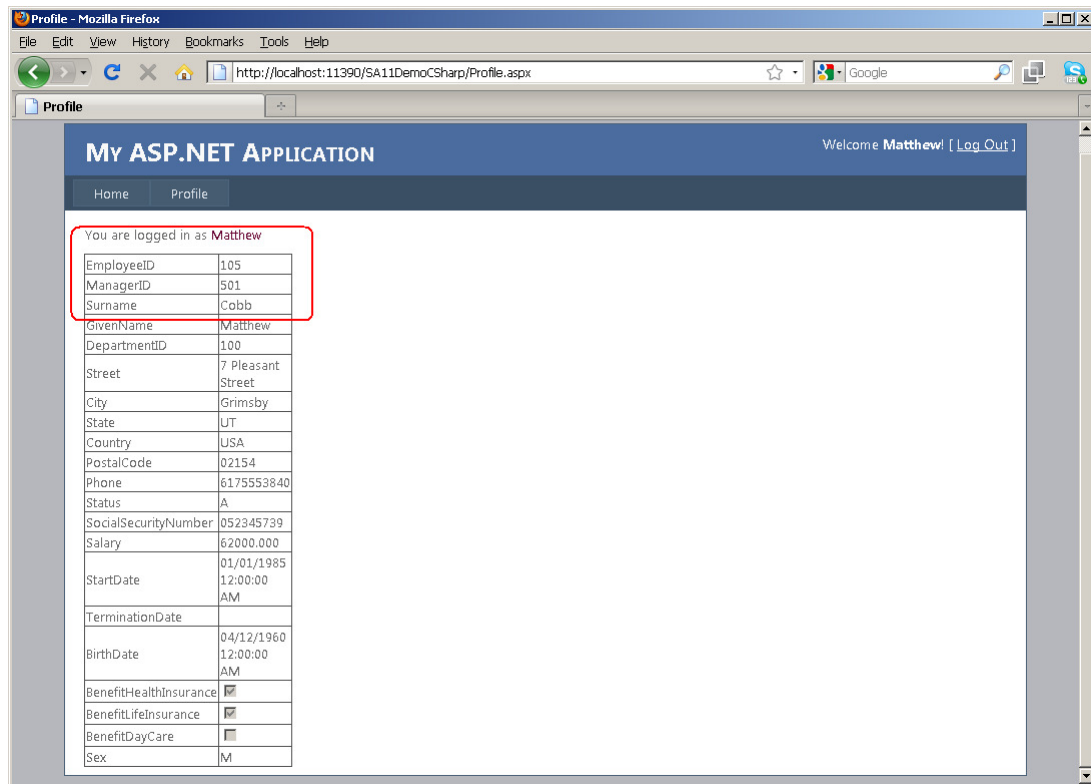
13. Now press **F5** to view the page. Notice that the label displays a message to indicate that the user has not signed in.



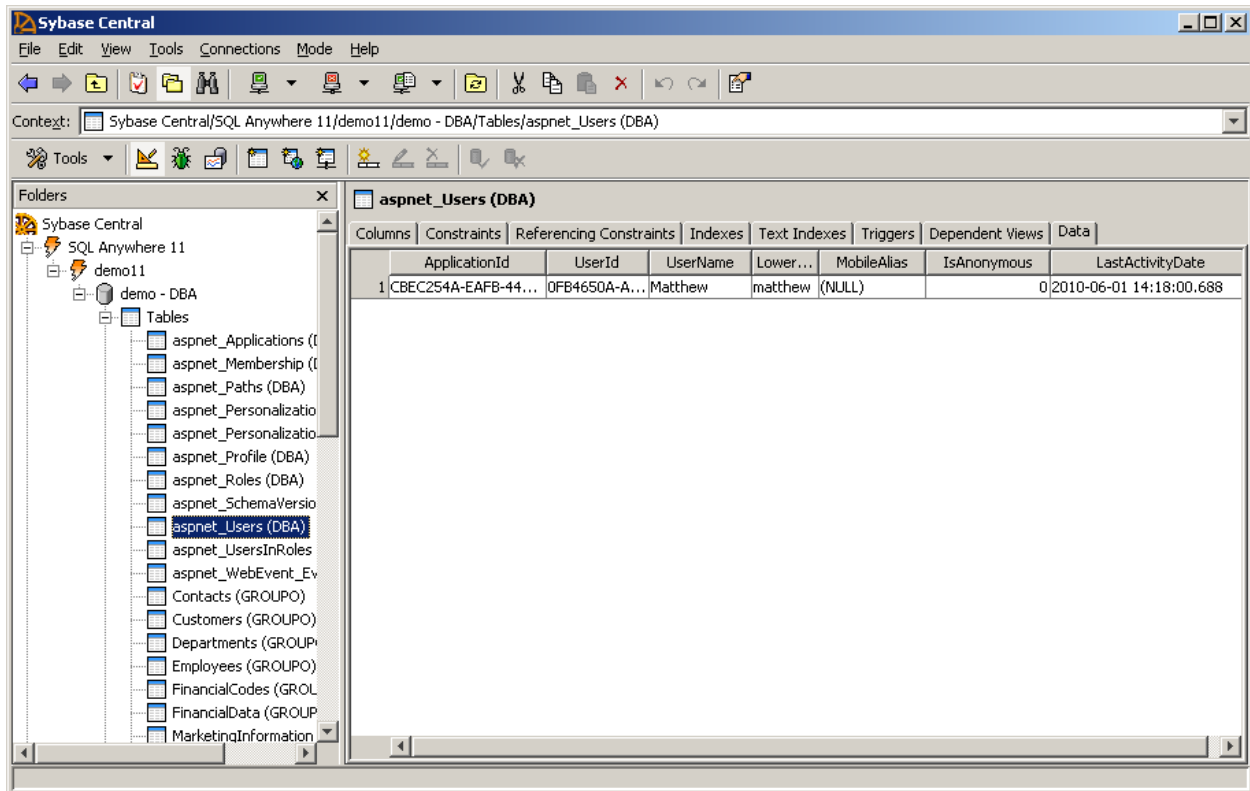
14. Click Log in at the upper right corner of the page. Click Register and enter the user name “Matthew” (a record for this employee already exists in the demo database). Set the password to “password”. Click **Create User**.



15. Notice that the Home page displays the public information, while the DetailsView control in the Profile page displays the private data for the user Matthew using all the columns from the Employee table.



16. Now click **Log out** and try to access the page again. Notice that the information is no longer available since the user is not authenticated.
17. Close the web browser.
18. Back in Sybase Central, expand **Tables** and select the table **aspnet\_users** to view its data. Notice that there's a new row created for the user "Matthew" with his authentication information stored.



19. Select the table **aspnet\_Membership** and click the **Data** tab. Notice that a new row is added to store the membership information of Matthew. When you use the other ASP.NET providers such as the Profile provider, you can check and manage the data stored in the corresponding table using Sybase Central as well.



Sybase Central

File Edit View Tools Connections Mode Help

Context: Sybase Central/SQL Anywhere 11/demo11/demo - DBA/Tables/aspnet\_Membership (DBA)

Tools

aspnet\_Membership (DBA)

Columns Constraints Referencing Constraints Indexes Text Indexes Triggers Dependent Views Data

ApplicationId	UserId	Password	Passw...	PasswordSalt	MobilePIN	Email	LoweredE...	Pass...	...	IsApproved	IsLockedOut
1/CBEC254A-EA...	DFB4650A-A917-46...	LnSz1Gx0Hb8O...		1jwAH53aPjH6Fek...	(NULL)	mat@mat.com	mat@mat....	(NULL)	(...	1	02010-

demo - DBA

- Tables
  - aspnet\_Applications (DBA)
  - aspnet\_Membership (DBA)
  - aspnet\_Paths (DBA)
  - aspnet\_PersonalizationAllu
  - aspnet\_PersonalizationPerf
  - aspnet\_Profile (DBA)
  - aspnet\_Roles (DBA)
  - aspnet\_SchemaVersions (D
  - aspnet\_Users (DBA)
  - aspnet\_UsersInRoles (DBA)
  - aspnet\_WebEvent\_Events
  - Contacts (GROUPO)
  - Customers (GROUPO)
  - Departments (GROUPO)
  - Employees (GROUPO)
  - FinancialCodes (GROUPO)
  - FinancialData (GROUPO)
  - MarketingInformation (GRC
  - Products (GROUPO)
  - SalesOrderItems (GROUPO
  - SalesOrders (GROUPO)
- Views
- Indexes
- Text Indexes
- Text Configuration Objects
- Triggers
- System Triggers
- Procedures & Functions
- Events
- Domains
- External Environments
- Users & Groups
- Login Policies
- Login Mappings

## Enabling Dynamic Data

A data-driven web application usually requires a developer to customize the appearance and behaviour of data retrieved from a database. Without the help of Dynamic Data, a programmer would need to manipulate the SQL statements in a way such that the data is displayed in a desired format -- a tedious task that reduces the readability of code. ASP.NET Dynamic Data solves the problem by inferring at run time the appearance and behavior of data entities and deriving user interface behavior from it. With a few steps and very little code, we can take advantage of the Dynamic Data feature in .NET 4.0 to format the display of the data stored in a SQL Anywhere database.

### Steps

1. Run the simple web site, and log in as user Matthew, notice that the date format in the **StartDate** and **BirthDate** columns don't show up in a desired format. The time 12:00:00AM is automatically generated since there is no time information in the database. We want to modify the Employee class to display these two dates in a standard YYYY-MM-DD format.  
In addition, notice that the **TerminationData** field is empty, meaning that there is no record in the database. We will also set a default value to be displayed when the data is null. Close the web application.
2. Back in Visual Studio, add a new class file to the folder App\_Code and name the new file **Employee.cs** or **Employee.vb**.
3. Insert the following lines of code to the added file:

[C#]

```
using System.Web.DynamicData;
using System.ComponentModel.DataAnnotations;

namespace Model {

    //Extend the Employee partial class
    [MetadataType(typeof(EmployeeMeta))]
    public partial class Employee {

        public class EmployeeMeta
        {

            //re-format the StartDate column
            [DisplayFormat(DataFormatString="{0:yyyy-MM-dd}")]
            public object StartDate { get; set; }

            [DisplayFormat(DataFormatString="{0:yyyy-MM-dd}")]
            public object BirthDate { get; set; }

            //sets default value for the TerminationDate column
            [DisplayFormat(NullDisplayText="N/A")]
            public object TerminationDate { get; set; }
        }
    }
}
```

```

    }
}
}

```

## [VB]

```

Imports Microsoft.VisualBasic
Imports System.ComponentModel.DataAnnotations
Imports System.Web.DynamicData

Namespace Model

    <MetadataType(GetType(EmployeeMeta))> _
    Partial Public Class Employee
    End Class

    Public Class EmployeeMeta

        <DisplayFormat(DataFormatString:="{0:yyyy-MM-dd}")> _
        Public Property StartDate() As Object
            Get
            End Get
            Set(ByVal value As Object)
            End Set
        End Property

        <DisplayFormat(DataFormatString:="{0:yyyy-MM-dd}")> _
        Public Property BirthDate() As Object
            Get
            End Get
            Set(ByVal value As Object)
            End Set
        End Property

        <DisplayFormat(nulldisplaytext:="N/A")> _
        Public Property TerminationDate() As Object
            Get
            End Get
            Set(ByVal value As Object)
            End Set
        End Property

    End Class
End Namespace

```

4. Open Profile.aspx and switch to Source view. Replace the markup for the GridView control and the EntityDataSource object with the following lines:

```

<asp:DetailsView ID="DetailsView1" runat="server"
    Height="50px" Width="125px"
    DataSourceID="EntityDataSource1" AutoGenerateRows="true">
</asp:DetailsView>
<asp:EntityDataSource ID="EntityDataSource1" runat="server"
    ConnectionString="name=SAEntities"

```

```

        DefaultContainerName="SAEntities"
        EnableFlattening="False"
        EntitySetName="Employees"
        Where="it.GivenName = @emp_name"
        ContextTypeName="Model.SAEntities"
        EntityTypeFilter="" Select="">
</WhereParameters>
<asp:SessionParameter DefaultValue="" Name="emp_name" SessionField="uid"
        DbType="String" />
</WhereParameters>
</asp:EntityDataSource>

```

This deletes the bound fields for the DetailsView control and lets Dynamic Data format the data.

5. Add the following line to the Page\_Load event of the Profile.aspx page:

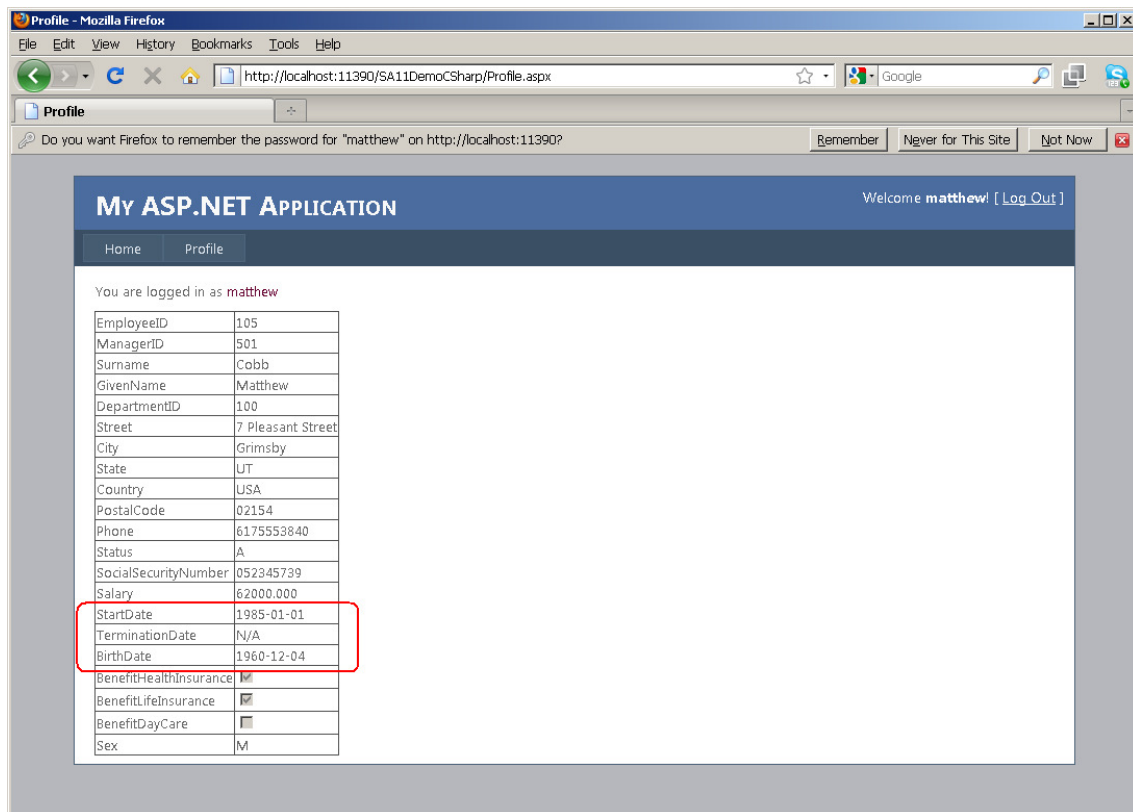
[C#]

```
DetailsView1.EnableDynamicData(typeof(Model.Employee));
```

[VB]

```
DetailsView1.EnableDynamicData(GetType(Model.Employee))
```

6. Now run the web site again and log in as user Matthew. Notice changes in the display format of the StartDate, BirthDate and TerminationDate fields are made according to the annotation in the Employee code file.



7. Close the web browser. This concludes the tutorial.

## Removing the SQL Anywhere ASP.NET Providers

If you want to revert the demo database to its original state, you must un-install the SQL Anywhere ASP.NET providers.

1. Start the ASP.NET Security Schema Setup Wizard mentioned in the installation section and connect to the demo database.
2. Select all the features for removal and un-check **Preserve Data** at the third step, then complete the wizard.
3. The wizard will remove the SQL Anywhere ASP.NET providers tables from the demo database.

## Summary

By taking advantage of the SQL Anywhere ASP.NET providers and the enhanced web site template provided by ASP.NET 4.0, developers can build a security-enabled web site with very little effort. The SQL Anywhere integration components for Visual Studio 2010 and its Entity Model support further enables developers to implement data-driven web applications by binding EntityDataSources to data-bound server controls. Using the new Dynamic Data feature of .NET 4.0, developers can easily implement business logic simply by extending a partial class instead of struggling with markup or SQL queries.