



チュートリアル：SQL Anywhere を使った ASP.NET Web ページの作成

目次

はじめに.....	1
必要なソフトウェア	1
概要.....	2
SQL Anywhere ASP.NET プロバイダのインストール.....	3
SQL Anywhere ASP.NET プロバイダのスキーマの表示.....	6
ASP.NET Web サイトの作成とエンティティ・データ・モデルの追加.....	9
データを表示するための Web コントロールの作成.....	14
SQL Anywhere ASP.NET Data Provider の構成.....	17
認証済みユーザに対する個人情報の表示.....	21
Dynamic Data 機能の有効化.....	28
SQL Anywhere ASP.NET プロバイダの削除.....	31
まとめとその他のリソース.....	32

改訂履歴

バージョン 1.1 - 2010 年 7 月

はじめに

このホワイトペーパーでは、SQL Anywhere および Visual Studio 2010 を使ってデータベース駆動型の ASP.NET Web サイトを構築する方法を説明します。Web アプリケーションは、単純な企業ディレクトリです。パブリック・ページには、一般的な従業員情報が表示され、パスワードで保護されるセキュア・ページには、詳細な情報が表示されます。セキュア・ページのデータは、正しい認証情報を使ってログインしてからでないと表示されません。SQL Anywhere ASP.NET プロバイダの使用は、セキュリティ・メカニズムの実装方法として注目されています。各項では、SQL Anywhere ASP.NET プロバイダのインストール、構成、および設定方法と、SQL Anywhere がアプリケーションのセキュリティ情報をどのようにデータベースに格納するかなどについて説明します。また、EntityDataSource オブジェクトをデータバインド・サーバ・コントロールにバインドする方法と、.NET Framework 4.0 の新しい Dynamic Data 機能をサンプル Web サイトに統合する方法についても説明します。

必要なソフトウェア

- SQL Anywhere 12 以上 - 以下の場所から SQL Anywhere Developer Edition を無料で入手できます。
http://www.ianywhere.jp/dl/dl_evl.html
- Microsoft Visual Studio 2010 および .NET Framework 4.0
- オプションのソース・コード (C# および Visual Basic)

概要

このチュートリアルの内容は、以下のとおりです。

- SQL Anywhere ASP.NET プロバイダをインストールします。
- デモ・データベースに接続し、Sybase Central を使って SQL Anywhere ASP.NET プロバイダのスキーマを表示します。
- ASP.NET Web サイトを作成します。この項では Web サイトを作成し、SQL Anywhere データベースから Web サイトに EDM (Entity Data Model : エンティティ・データ・モデル) を追加します。
- GridView Web コントロールを作成し、EntityDataSource オブジェクトにバインドします。
GridView コントロールは EDM からデータを取得し、パブリック・ページにそのデータを表示します。
- SQL Anywhere ASP.NET プロバイダを構成します。この項では、web.config ファイルにおけるプロバイダの登録方法と ASP.NET 管理ツールの使用方法について説明します。
- 会員専用ページを作成します。このページの情報は、認証されたユーザのみアクセス可能です。
- ASP.NET 4.0 の Dynamic Data 機能を使って、データの表示をフォーマットします。

SQL Anywhere ASP.NET プロバイダのインストール

SQL Anywhere ASP.NET プロバイダを使用すれば、SQL Anywhere データベースをバックエンド記憶領域として使用するセキュリティ・メカニズムを構築できます。ユーザの機密情報の格納および管理のために、必要なスキーマを所定のデータベースに追加するためには、SQL Anywhere ASP.NET プロバイダをインストールする必要があります。

SQL Anywhere には、以下の 5 種類のプロバイダが含まれています。

- **Membership Provider** : 認証および承認サービスを提供します。
- **Roles Provider** : 役割 (ロール) の作成、役割へのユーザの追加、役割の削除に対応するメソッドを提供します。グループへのユーザの割り当てやパーミッションの管理を行います。
- **Profiles Provider** : ユーザの基本設定など、ユーザ情報の読み取り、格納、および取得に対応するメソッドを提供します。
- **Web Parts Personalization Provider** : カスタマイズしたコンテンツや Web ページのレイアウトのロードと格納に対応するメソッドを提供します。
- **Health Monitoring Provider** : 配備済み Web アプリケーションのステータス監視に対応するメソッドを提供します。

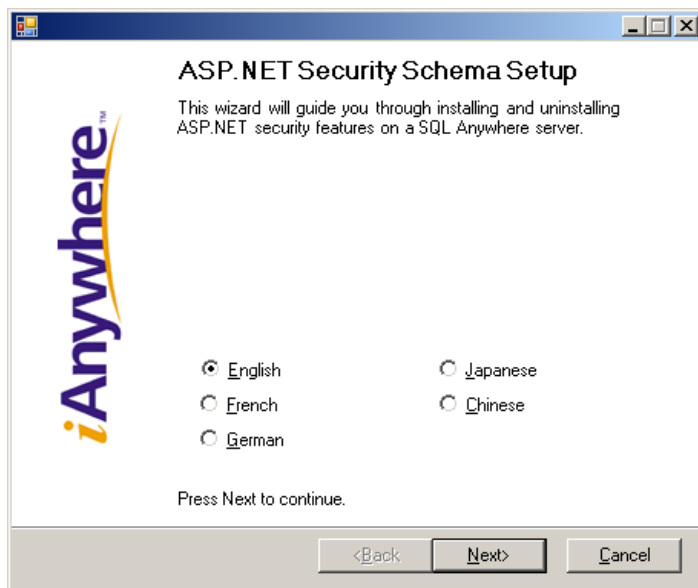
SQL Anywhere ASP.NET プロバイダの詳細については、以下のオンライン・マニュアルを参照してください。

<http://dcx.sybase.com/index.html#1201/ja/dbprogramming/aspdotnet-providers.html>

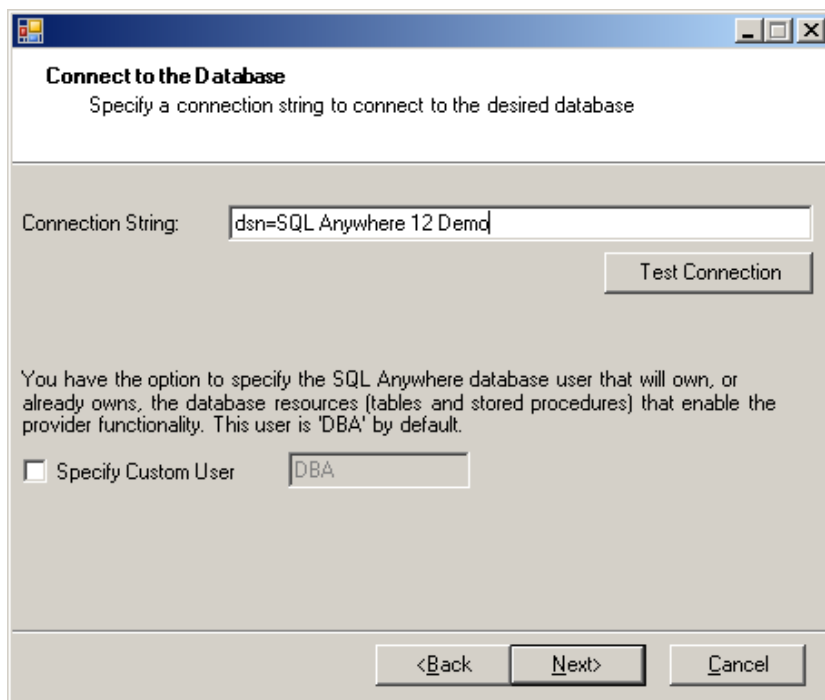
SQL Anywhere には、必要なセキュリティ・スキーマをデータベースに追加するためのセットアップ・ウィザードも用意されています。SQL Anywhere ASP.NET プロバイダは、新しいデータベースにも、既存のデータベースにも追加できます。わかりやすくするために、以下の手順では SQL Anywhere デモ・データベース (<http://dcx.sybase.com/index.html#1201/ja/saintro/fg-sademo.html>) を使用します。

手順

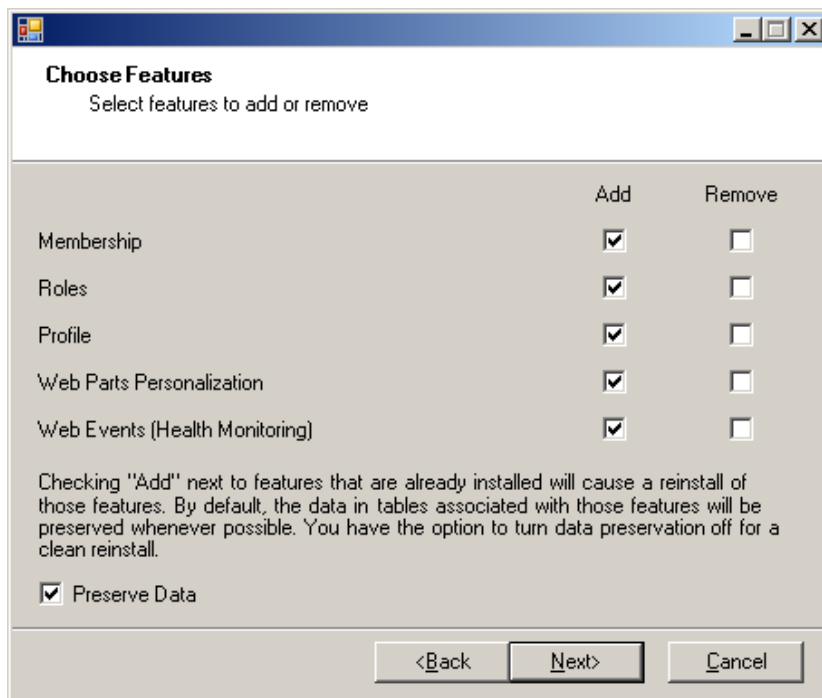
1. Windows Explorer を使用して、SQL Anywhere がインストールされているインストール・フォルダを参照し (以下はデフォルトのインストール・パス)、以下のフォルダを探します。
C:\Program Files\SQL Anywhere 12\Assembly
2. SASetupAspNet.exe をダブルクリックし、[ASP.NET Security Schema Setup] ウィザードを実行します。または、コマンド・ライン・プロンプトからでも SASetupAspNet.exe を実行できます。コマンド・ラインを使用して SASetupAspNet.exe にアクセスする場合は、疑問符 (?) 引数を使用すると、データベースの構成に関する詳細なヘルプが表示されます。



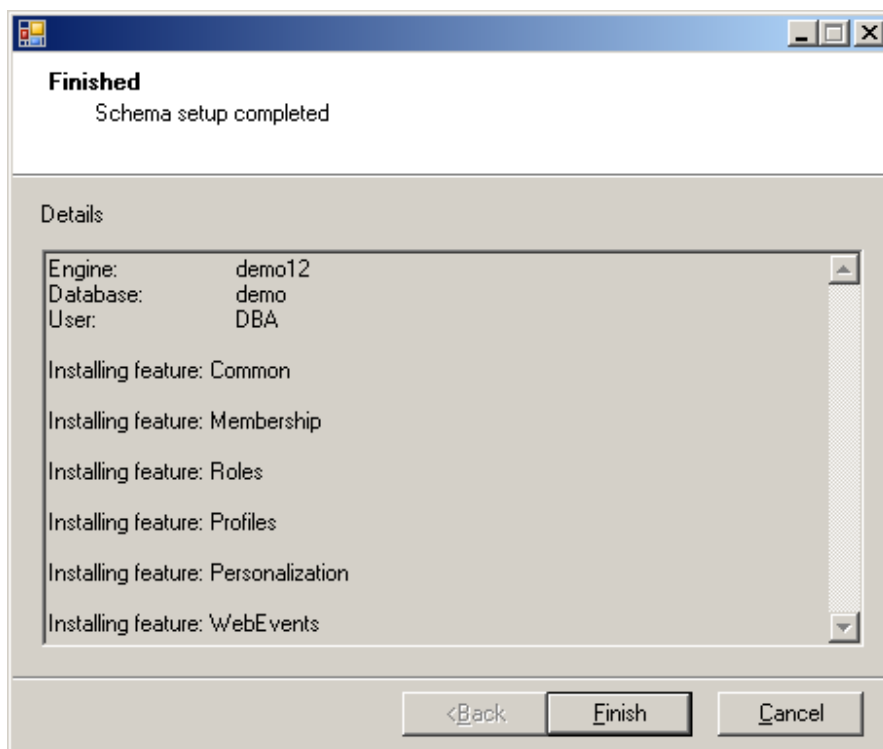
3. 希望の言語を選択し、[Next] をクリックします。
4. [Connection String] に “dsn=SQL Anywhere 12 Demo” と入力し、[Next] をクリックします。ここで、データベース接続のテストも行えます。



5. 必要な機能を追加し(この場合はすべて)、[Next] をクリックします。すでにインストール済みの機能を削除することもできます。



6. インストール内容を確認し、[Finish] をクリックします。

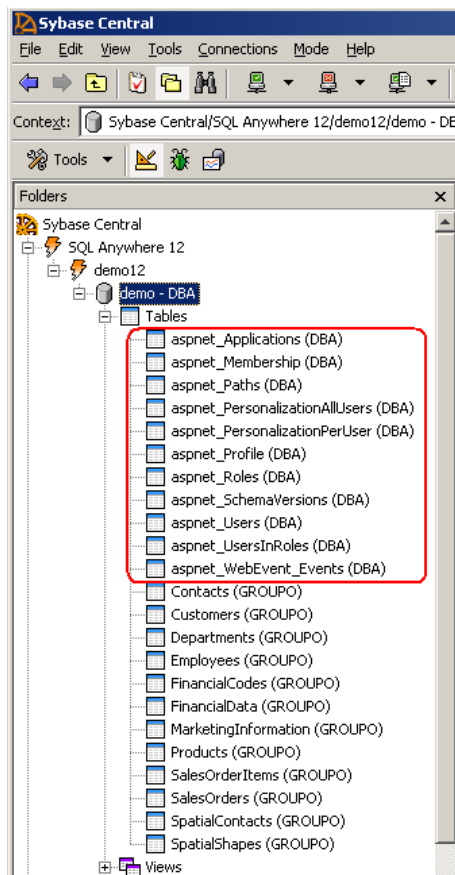


SQL Anywhere ASP.NET プロバイダのスキーマの表示

前の項のウィザードにより、必要なテーブルおよびストアド・プロシージャがデモ・データベースに追加されました。いずれも“aspnet_”というプレフィックスが付いています。この項では、変更する点を示しながら、ログインに関する機密情報とアプリケーションのセキュリティ情報が、SQL Anywhere でどのように格納されるかについて説明します。

手順

1. Sybase Central を開き、デモ・データベースに接続します。[Folders] ビューで [Tables] を展開し、ウィザードによって作成されたテーブルを表示します。以下の図のように、“aspnet_”というプレフィックスが付いた新しいテーブルが、デモ・データベースに追加されています。これらのテーブルには、Web サイトで使用されるセキュリティ・データが格納されます。



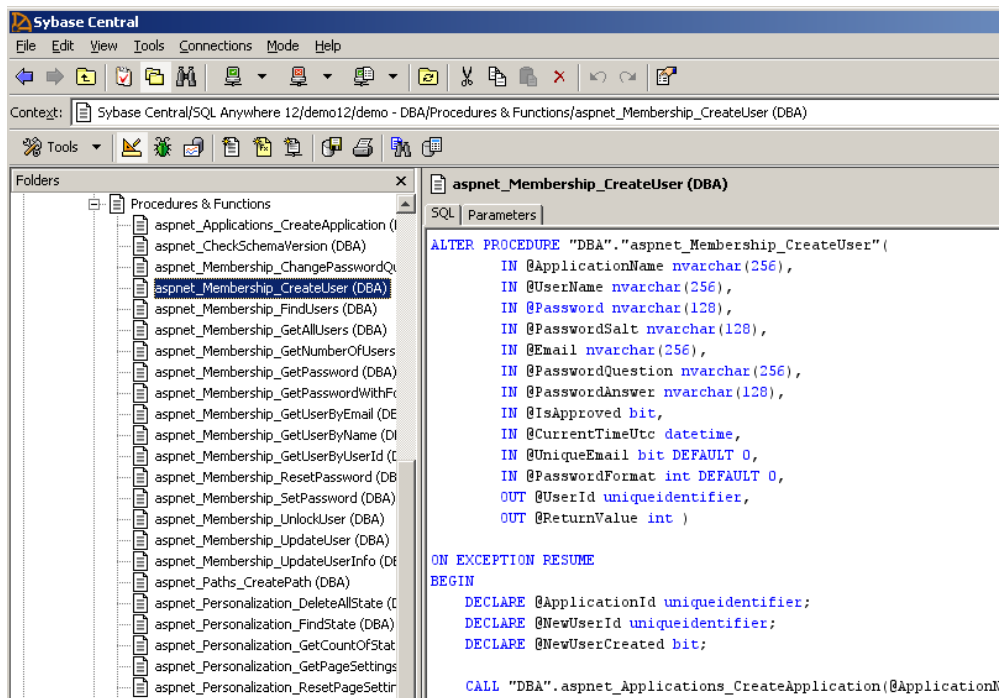
2. テーブル aspnet_Membership をクリックすると、テーブルのカラムが表示されます。このテーブルには、ユーザのメンバシップ情報が格納されます。たとえば、ユーザ ID (プライマリ・キー)、パスワード、セキュリティのための質問、最終ログイン日付など、Web サイトの認証に必要なすべての情報が格納されます。デフォルトでは、データベースに格納された情報が確実に保護されるように、パスワード形式がハッシュされます。

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

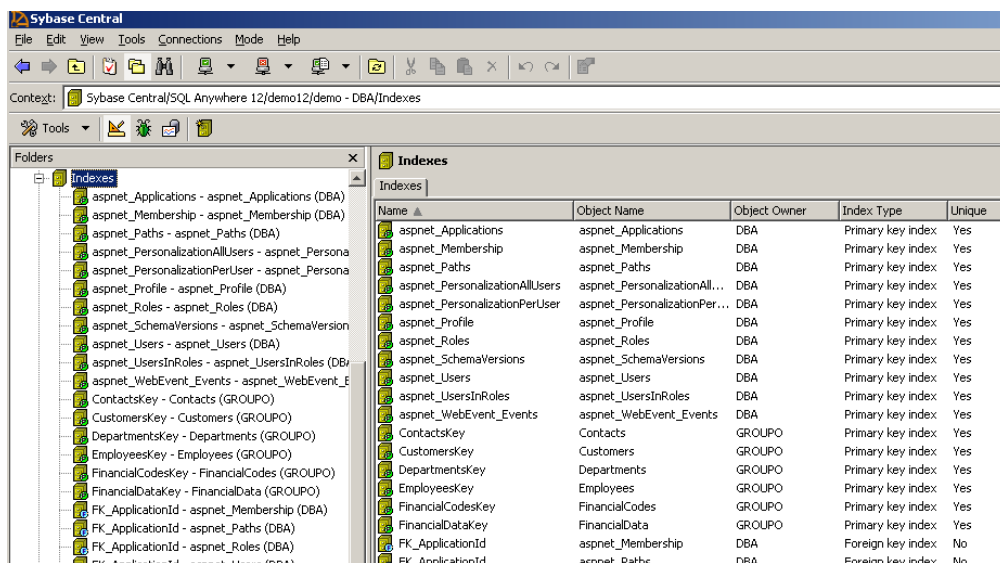
- [Constraints] タブをクリックします。ウィザードにより、カラムに対して適切なキー制約が設定されていることがわかります。

aspnet_Membership (DBA)	
[Columns] [Constraints] [Referencing Constraints] [Indexes] [Text Indexes] [Triggers] [Dependent Views] [Data]	
Name	Constraint Type
ASA206	Primary key constraint
FK_ApplicationId	Foreign key constraint
FK_UserId	Foreign key constraint

- [Folders] ビューで [Procedures and Functions] を展開します。ウィザードにより、各プロバイダで使用可能な操作のプロシージャが追加されています。aspnet_Membership_CreateUser プロシージャをクリックすると、SQL コードが表示されます。これは、データベースに新しいユーザ・レコードが挿入されたときに Membership Provider が使用するプロシージャです。



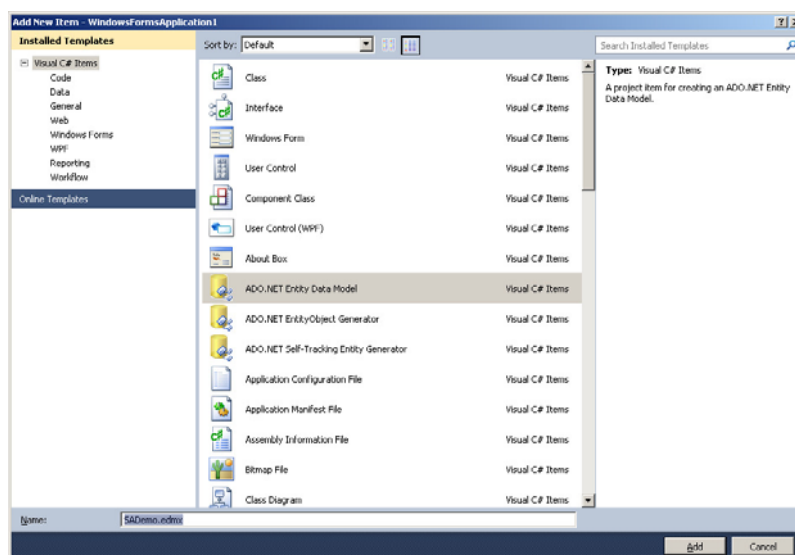
5. [Indexes] を展開します。パフォーマンスを向上するために、追加されたテーブル用のインデックスも自動的に作成されています。



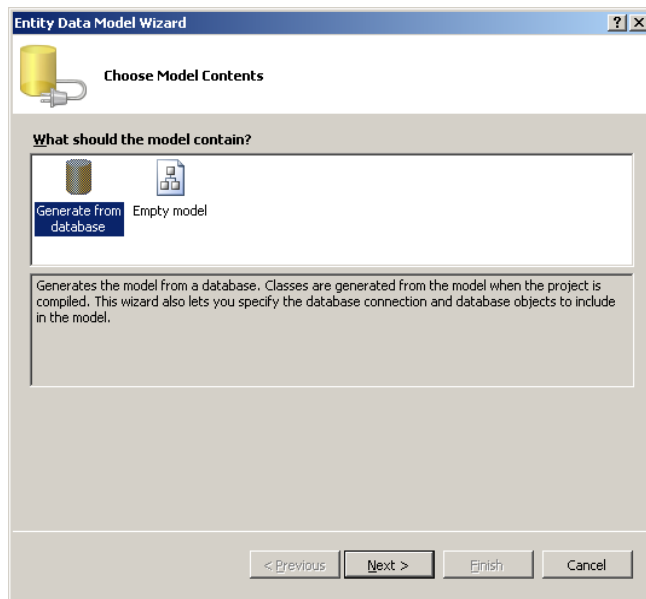
ASP.NET Web サイトの作成とエンティティ・データ・モデルの追加

Visual Studio で定義された任意の SQL Anywhere データベース・プロファイルを使用して、新しいエンティティ・データ・モデル (EDM) を作成できます。以下の手順に従って、ASP.NET Web サイトを作成し、このサイトに SQL Anywhere デモ・データベースを EDM として追加します。

1. Visual Studio 2010 を起動します。
2. [File] > [New] > [Web Site] を選択します。
3. [New Web Site] ダイアログで [ASP.NET Web Site] を選択します。
4. 自分で選んだ既知の場所に、この Web サイトを 'Sample_asp.net' という名前で保存します。
5. Solution Explorer で Web サイト・プロジェクトを右クリックし、ポップアップ・メニューから [Add New Item] > [ADO.NET Entity Data Model] をクリックします。
6. [Name] フィールドに SADemo.edmx と入力します。[Add] をクリックします。
7. [Yes] をクリックすると、ADO.NET Entity Model がフォルダ App_Code に追加されます。



8. [Entity Data Model] ウィザードが表示されます。[Generate from database] を選択し、[Next] をクリックします。



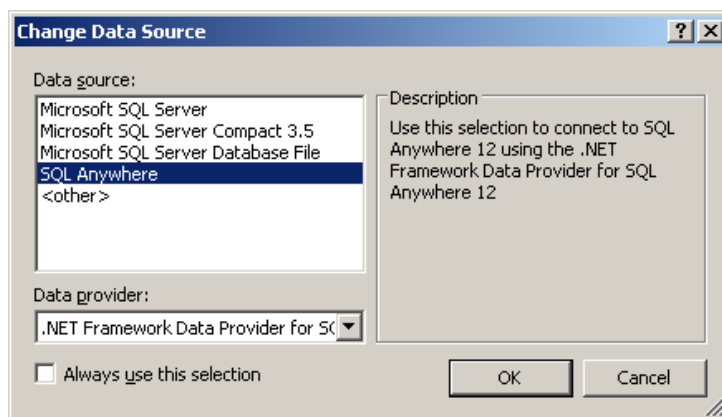
9. [New Connection] をクリックします。[Data Source] の横の [Change] ボタンをクリックします。[Data source] リストで [SQL Anywhere] を選択し、[OK] をクリックします。

[Data source] リストに [SQL Anywhere] が表示されていない場合、Visual Studio 用の SQL Anywhere 統合コンポーネントが正しくインストールされていることを確認してください。統合コンポーネントをインストールするには、以下の手順を実行します。

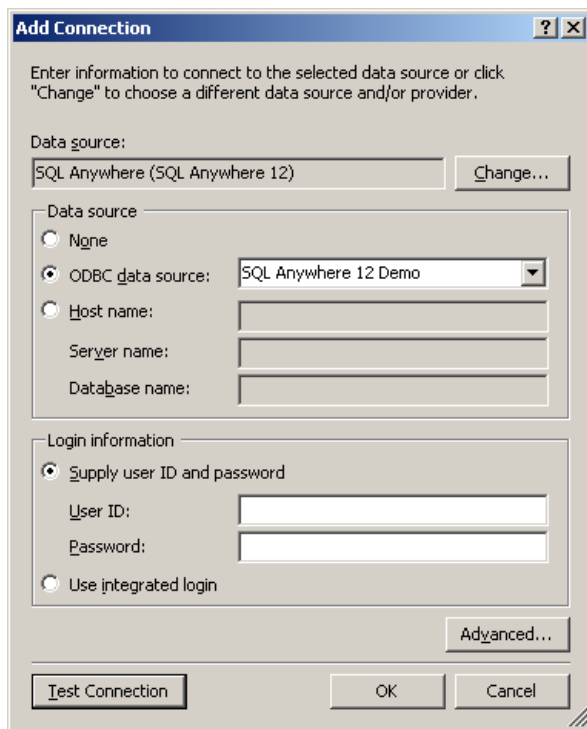
1. Visual Studio 2010 を閉じます。
2. コマンド・プロンプトを開き、以下のディレクトリに移動します。

C:\Program Files\SQL Anywhere 12\Assembly\v2

3. 次に、以下のコマンドを実行します： SetupVSPackage.exe -i



10. [ODBC Data Source] の名前をクリックし、[SQL Anywhere 12 Demo] を選択します。[OK] をクリックします。



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:

Data source

☐ None

☒ ODBC data source:

☐ Host name:

Server name:

Database name:

Login information

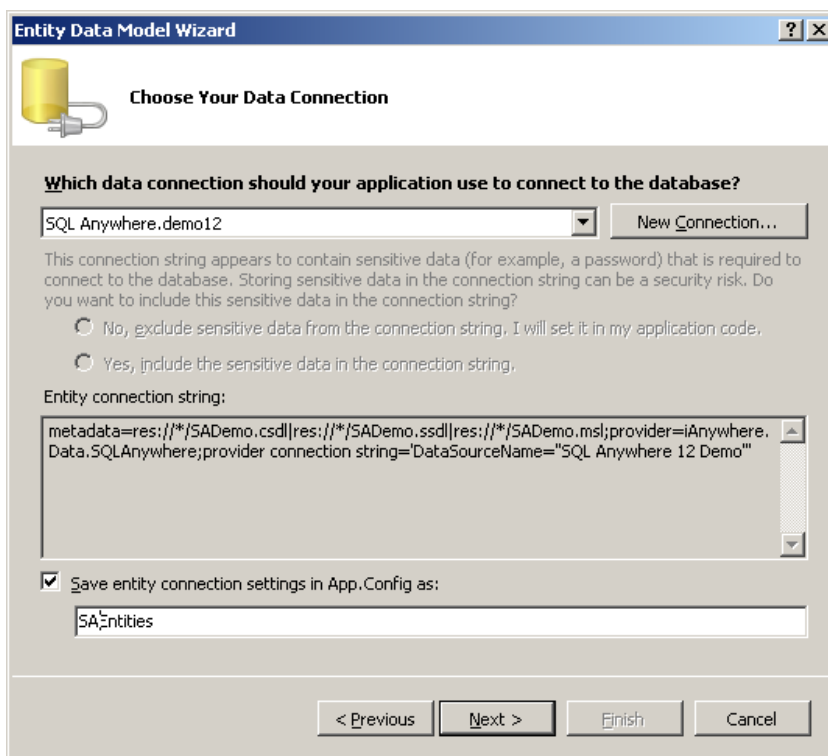
☒ Supply user ID and password

User ID:

Password:

☐ Use integrated login

11. エンティティ接続文字列の名前を“SAEntities”に変更し、[Next] をクリックします。



Entity Data Model Wizard

Choose Your Data Connection

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

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?

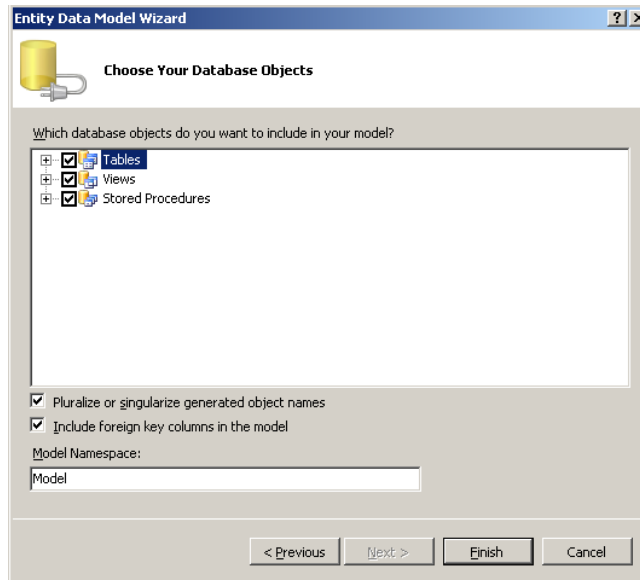
☐ No, exclude sensitive data from the connection string. I will set it in my application code.

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

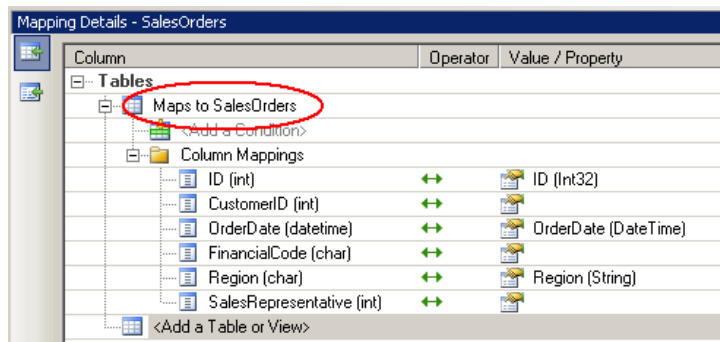
Entity connection string:

☒ Save entity connection settings in App.Config as:

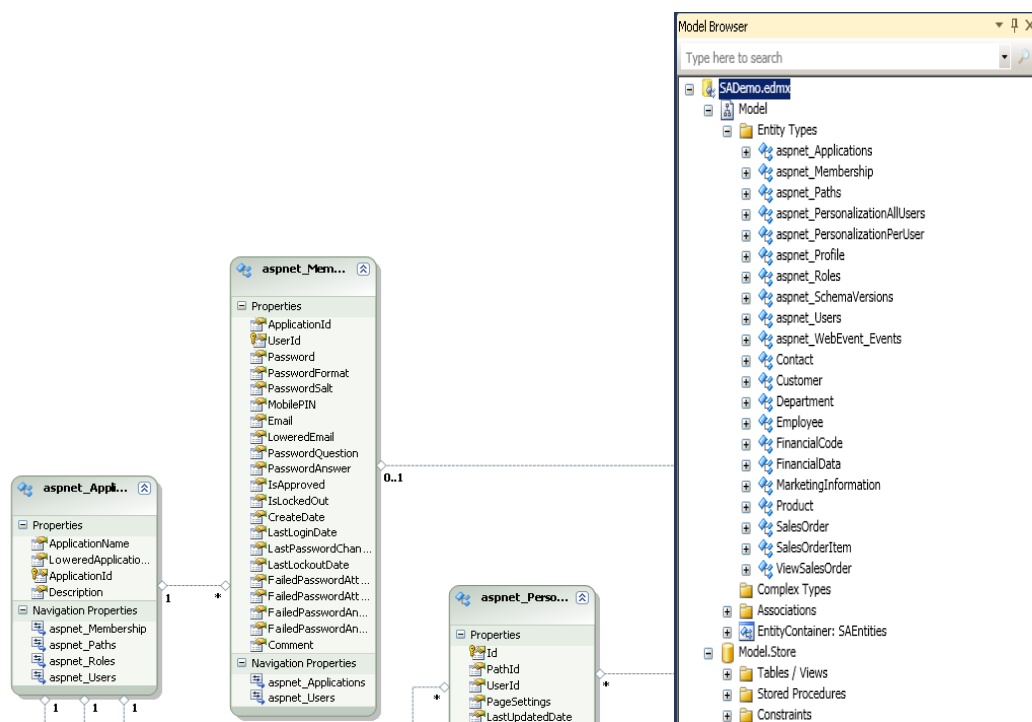
12. モデルに含めるデータベース・オブジェクトとしてすべてのデータベース・オブジェクトを選択し、[Finish] をクリックします。実際には、EDM はアプリケーションに必要なオブジェクトだけを含めます。このチュートリアルでは、簡単にするために、すべてのオブジェクトを追加します。



13. SADemo.edmx ファイルを開くと、Entity Designer にモデルが視覚的に表示されます。
SalesOrders エンティティを右クリックし、[Table Mapping] を選択すると、マッピングの詳細情報が表示されます。以下の図は、データベース・スキーマに対してエンティティが適切にマッピングされていることを示しています。



14. Entity Designer を使用して、ASP.NET プロバイダによってデモ・データベースに追加されたテーブルを表示することもできます。Entity Designer で何もない部分を右クリックし、[Model Browser] をクリックすると、Model ネームスペース内のすべてのエンティティ型の概要が表示されます。

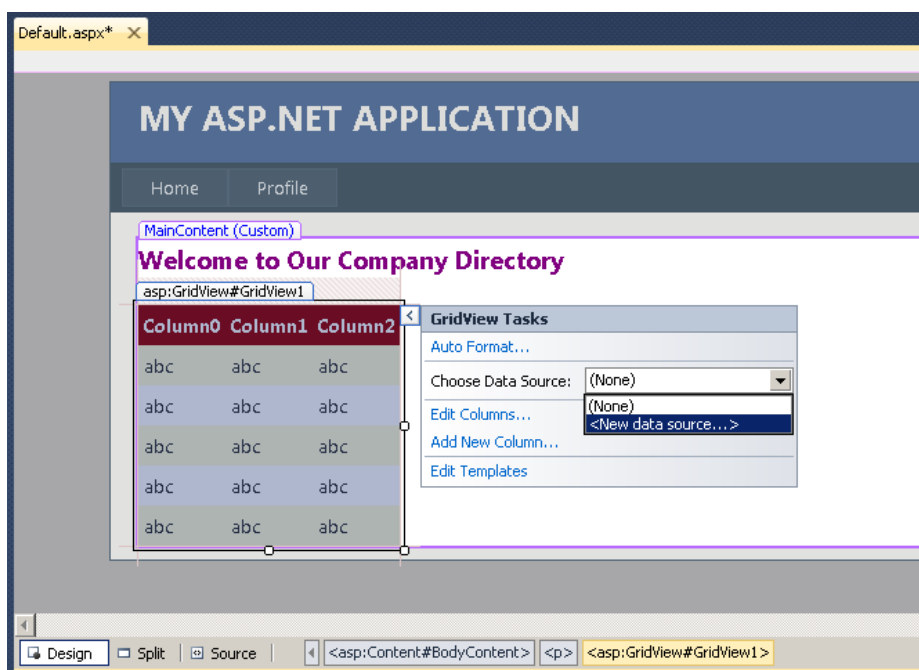


データを表示するための Web コントロールの作成

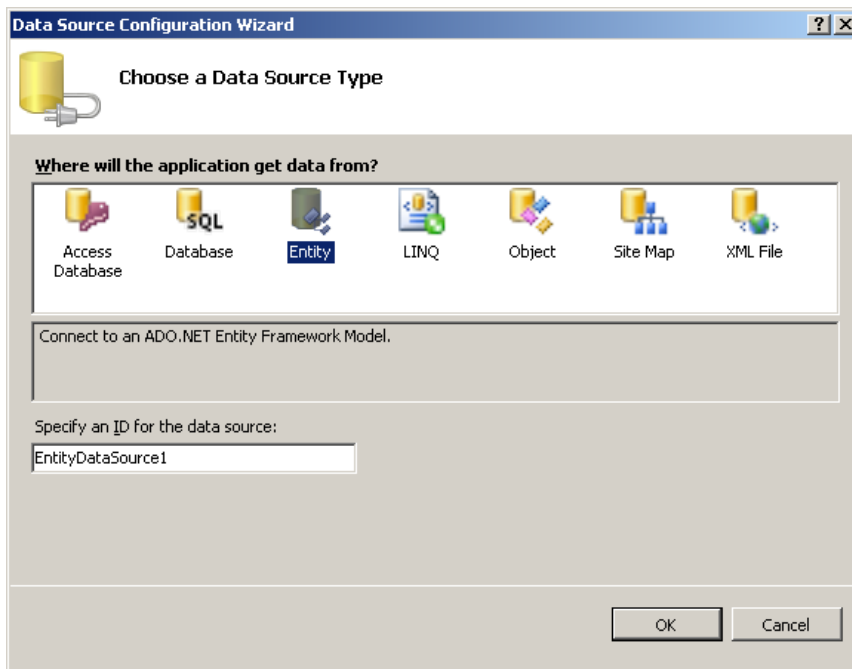
ASP.NET プロバイダと Web サイトで使用する Entity Model の設定が完了したので、デフォルト・ページに GridView コントロールを追加し、EntityDataSource にバインドします。このページは、Web サイトのパブリックな領域になります。このページでは、すべてのユーザが認証の必要なく情報を参照できます。

手順

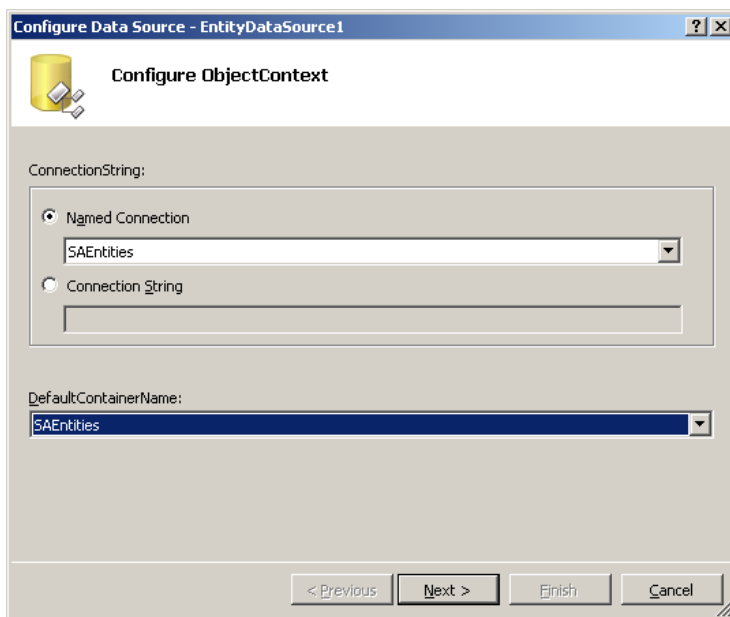
1. Default.aspx を開いてデザイン・ビューに切り替えます。ツールボックスから GridView コントロールをドラッグし、デフォルトのボディ・コンテンツと置き換えます。
2. [GridView Tasks] メニューで、[Choose Data Source] ドロップダウン・リストを表示し、[<New data source...>] を選択します。



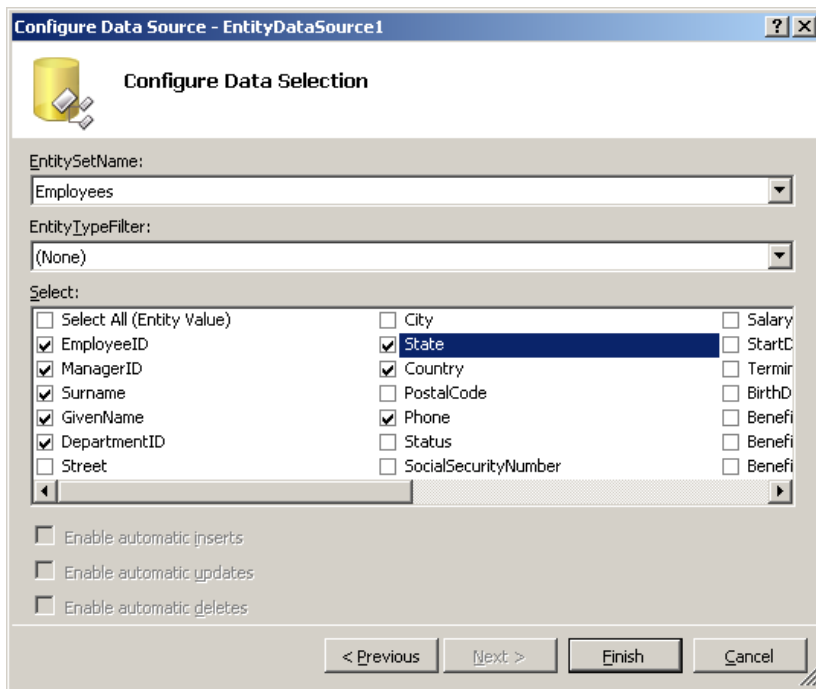
3. [Entity] を選択し、[OK] をクリックします。



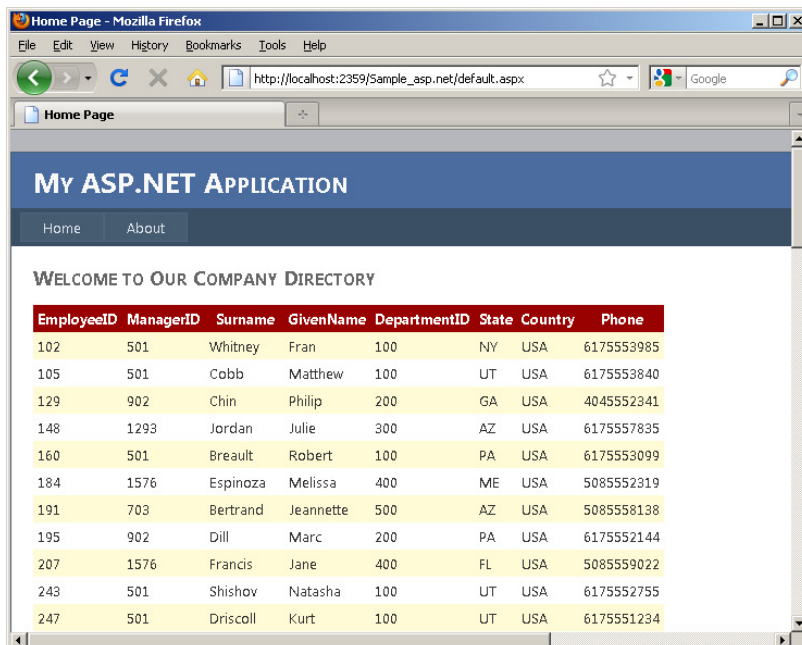
4. [Configure Data Source] ウィザードが表示されます。[Named Connection] ドロップダウン・リストから [SAEntities] を選択し、[Next] をクリックします。



5. [EntitySetName] ドロップダウン・リストから [Employees] を選択します。誰でも参照できるパブリック情報として、[EmployeeID]、[ManagerID]、[Surname]、[GivenName]、[DepartmentID]、[State]、[Country]、[Phone] を選択します。[Finish] をクリックします。



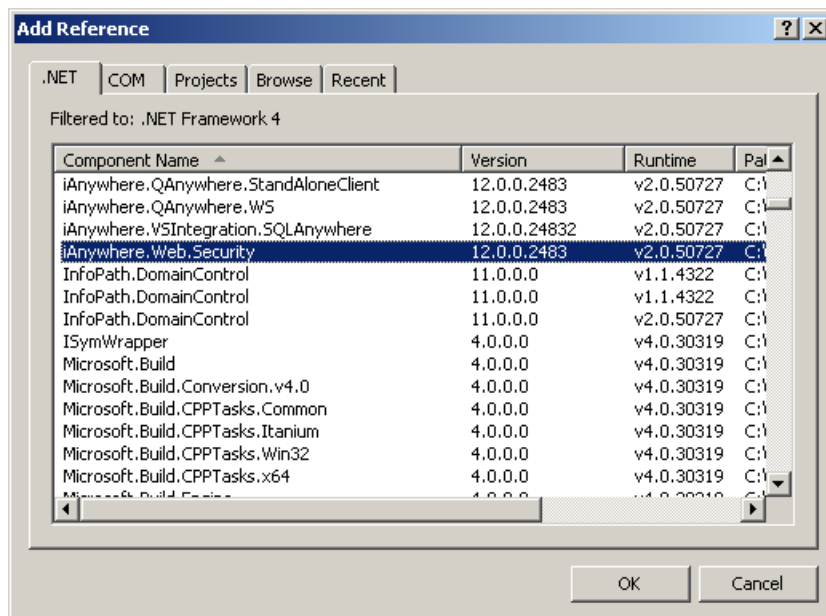
6. [F5] を押してアプリケーションを実行／デバッグします。必要に応じて、Web.config ファイル内でデバッグを有効にします (有効にするかどうかを尋ねるダイアログが表示される場合があります)。デフォルト・ページに、デモ・データベースの Employee テーブルから選択したカラムが表示されます。この情報は、このページにアクセスできるユーザであれば誰でも参照できます。ページを閉じてアプリケーションを終了します。



SQL Anywhere ASP.NET Data Provider の構成

次に、セキュア・ページを Web サイトに追加し、適切なセキュリティ設定を有効にします。また、SQL Anywhere ASP.NET プロバイダを使用するように、アプリケーションを構成する必要があります。以下の手順は、SQL Anywhere ASP.NET プロバイダの登録方法と、ASP.NET 管理ツールを使ったユーザ情報の管理方法を示しています。

1. Solution Explorer でプロジェクトを右クリックし、[Add Reference] を選択します。[.NET] タブを選択し、iAnywhere.Web.Security アセンブリへの参照を Web サイトに追加するために、リストから [iAnywhere.Web.Security] を選択します。



2. web.config ファイルを開き、<connectionStrings> 要素に接続文字列を追加します (ハイライト部分が変更箇所)。

```
<connectionStrings>
  <!-- 接続文字列の登録 -->
  <add name="MyConnectionString"
        connectionString="dsn=SQL Anywhere 12 Demo"
        providerName="iAnywhere.Data.SQLAnywhere" />
  <add name="SAEntities"
        connectionString="metadata=res://*/App_Code.SADemo.csdl|res://*/App_Code.SADemo.ssd1|res://*/App_Code.SADemo.msl;provider=iAnywhere.Data.SQLAnywhere;provider connection string='UserID=dba;Password=sql;DataSourceName=&quot;SQL Anywhere 12 Demo&quot;;'"
        providerName="System.Data.EntityClient" />
</connectionStrings>
```

3. プロバイダごとにエントリを追加し、<machineKey validation="SHA1"> を <system.web> 要素に追加します。SQL Anywhere ASP.NET プロバイダの名前をアプリケーションの "defaultProvider" 属性に追加します (ハイライト部分が変更箇所)。

```

<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="/"

```

```

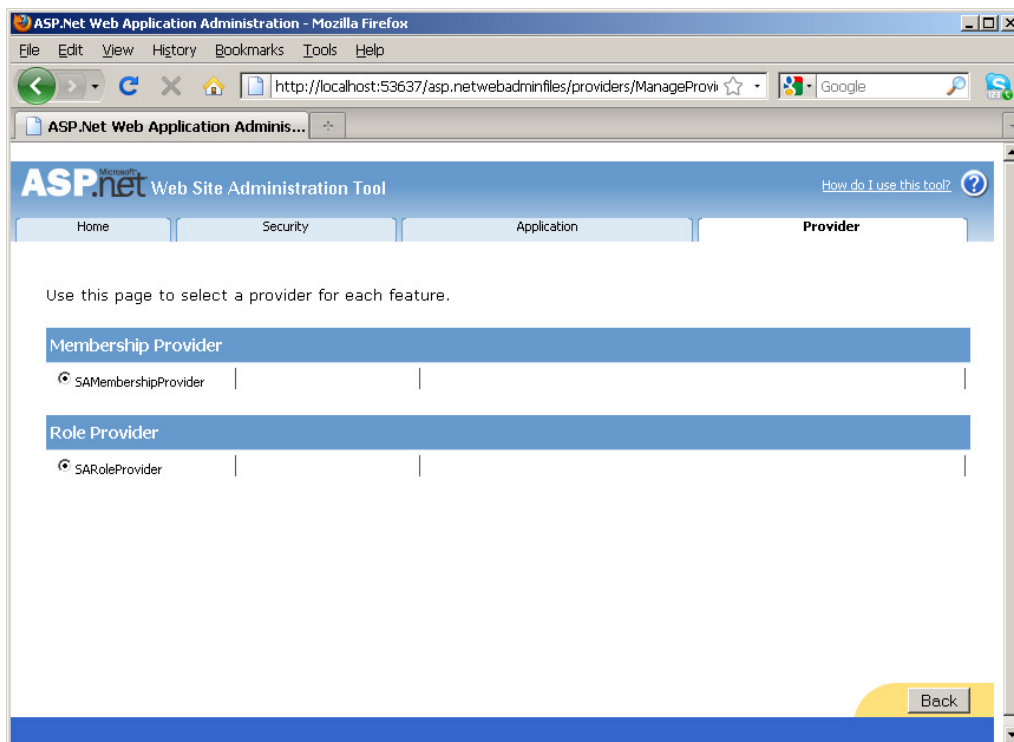
        commandTimeout="30" name="SARoleProvider"
        type="iAnywhere.Web.Security.SARoleProvider" />
    </providers>
</roleManager>
</system.web>

```

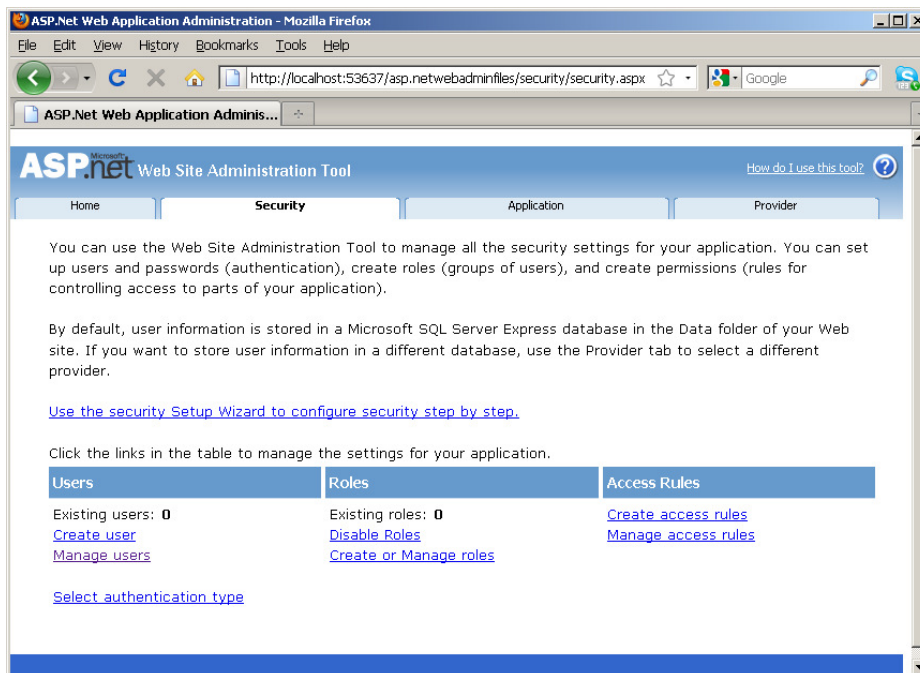
SQL Anywhere ASP.NET プロバイダの構成方法の詳細については、以下の API のオンライン・マニュアルを参照してください。

<http://dcx.sybase.com/index.html#1201/ja/dbprogramming/aspdotnet-providers.html>

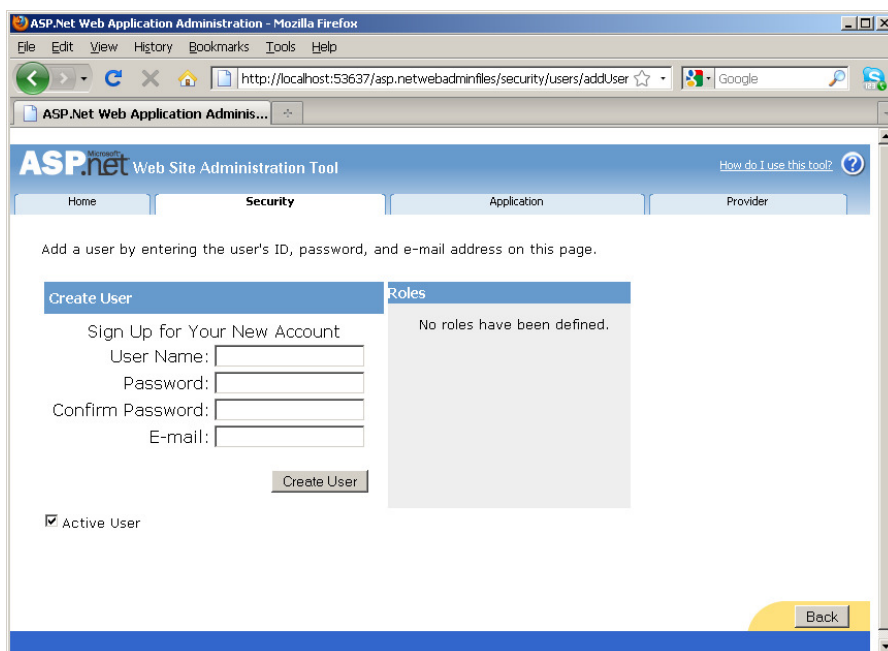
4. これで、Web Site Administration Tool を使用して新しいユーザを作成できます。このツールを使用するには、変更を保存してから、Visual Studio 2010 で [Website] メニューの [ASP.NET Configuration] をクリックします。[Provider Configuration] > [Select a different provider for each feature (advanced)] をクリックすると、SAMembershipProvider と SARoleProvider が使用されていることがわかります。



5. [Security] タブには、ロールの有効化やユーザごとのアクセス・ルールを作成など、Web サイトの管理タスクを実行するためのさまざまなツールが表示されます。



6. [Security] > [Create user] をクリックすると、Web フォームを入力して新しいユーザ・アカウントを作成できます。

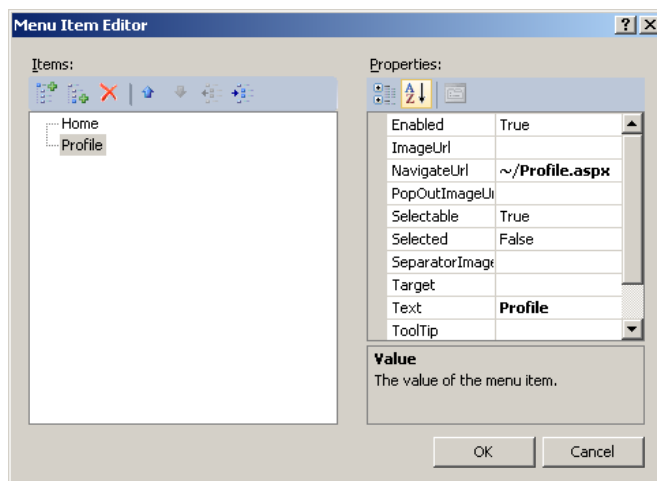


7. [ASP.NET Web Application Administration] ページを閉じます。

認証済みユーザに対する個人情報の表示

これで、ログインしている従業員のみ利用可能な、従業員のプロファイル情報が表示される 2 番目のページを作成できます。簡単にするために、Employee テーブルにすでに情報が存在するユーザを登録します (ユーザ名として名を使用)。組み込みの About.aspx ファイルの名前を変更し、個人情報を表示する従業員プロフィール・ページになるようカスタマイズします。Visual Studio には、強化された ASP.NET Web サイト・テンプレートも用意されています。このテンプレートには、基本的なメンバシップ機能がすでに構成済みの組み込みアカウント・フォルダが含まれています。以下の手順では、これらのテンプレートを利用して、サンプル・アプリケーションを設定します。

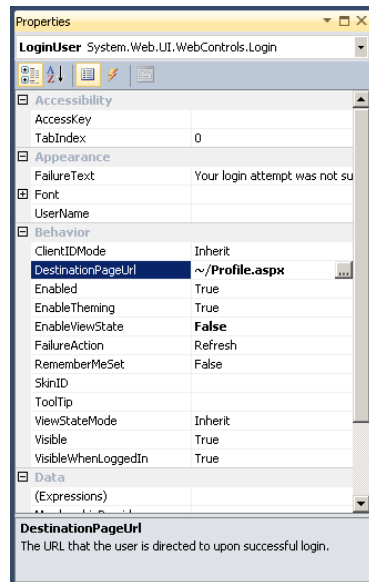
1. Solution Explorer で About.aspx ファイルを右クリックし、[Rename] をクリックします。ファイル名を Profile.aspx に変更します。対応するコード・ファイルも自動的に名前が変更されます。
2. メニュー項目名がファイル名と一致するように、マスタ・ページのメニュー・バーを編集する必要があります。site.master ファイルを開いてデザイン・ビューに切り替えます。ナビゲーション・メニューを探し、その横の [>] ボタンをクリックします。[Edit Menu Items] を選択し、Menu Item Editor で 2 番目のメニュー項目の URL を “~/Profile.aspx” に置き換え、Text プロパティを [Profile] に変更します。[OK] をクリックします。



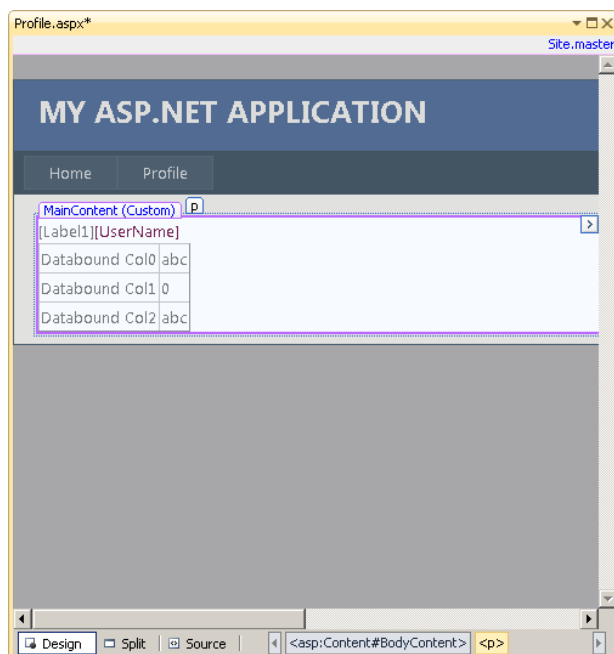
ソース・ビューに切り替えると、ナビゲーション・メニューのマークアップが以下のように表示されるはずです。

```
<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>
```

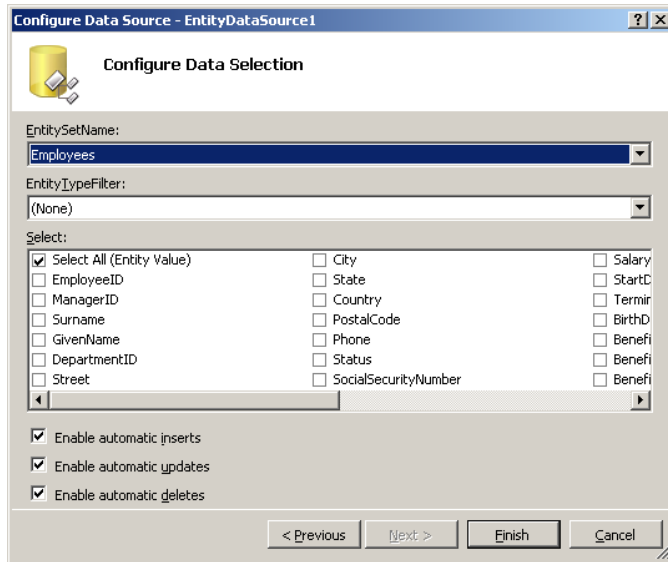
3. ログイン成功後に、このプロフィール・ページにユーザが自動的にリダイレクトされるように、デフォルトのログイン・テンプレートを編集します。Account/Login.aspx を開き、デザイン・ビューで Login コントロールを右クリックし、[Properties] をクリックします。DestinationPageURL プロパティに対して Profile.aspx を設定します。



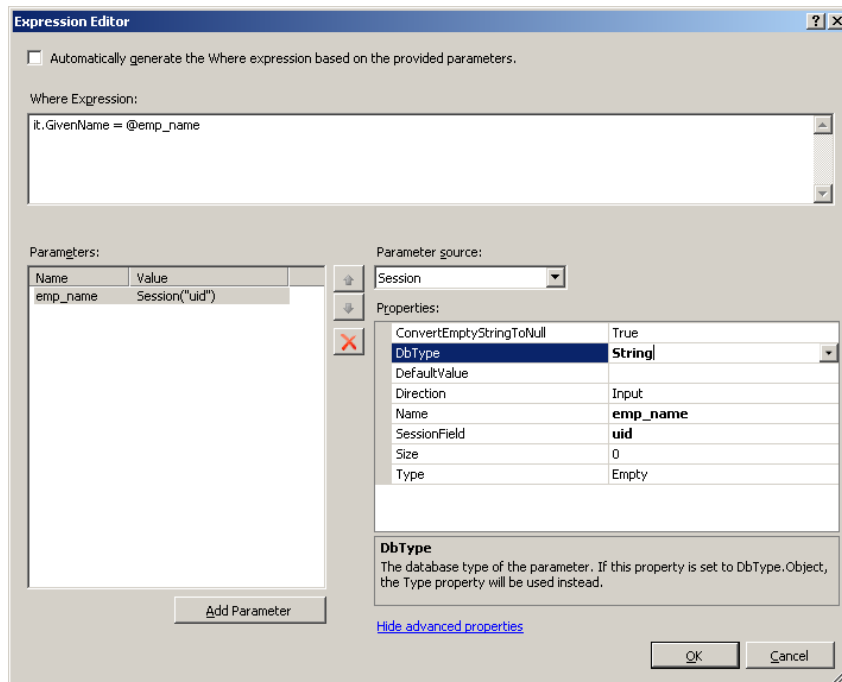
4. Profile.aspx を開き、MainContent プレースホルダのデフォルト行を削除します。次に、Label コントロール、LoginName コントロール、および DetailsView コントロールを MainContent プレースホルダにドラッグ・アンド・ドロップします。



5. [DetailsView Tasks] メニューで、[Choose Data Source] ドロップダウン・リストを表示し、[<New data source...>] を選択します。パブリック・ページの場合と同様に、新しい EntityDataSource を追加します。
6. [Entity] を選択し、[OK] をクリックします。
7. [Named Connection] ドロップダウン・リストから [SAEntities] を選択し、[Next] をクリックします。
8. [EntitySetName] ドロップダウン・リストから [Employees] を選択します。今回は、テーブルのすべてのカラムを選択します。[Finish] をクリックします。



9. DetailsView コントロールの下に新しい EntityDataSource が表示されます。これを右クリックし、[Properties] を選択します。
10. [Properties] メニューの横の [Where] ボタンをクリックして式エディタを開きます。[Add Parameter] をクリックし、[name] フィールドに “emp_name” と入力します。[Parameter Source] ドロップダウン・リストから [Session] を選択し、[Session] フィールドに “uid” と入力します。[Where Expression] フィールドに “it.GivenName=@emp_name” と入力します。
11. [Show advanced properties] をクリックし、[DbType] ドロップダウン・リストから [String] を選択します。[OK] をクリックして式エディタを閉じます。



12. Profile.aspx ページの Page_Load イベントに以下のコードを挿入します。

[C#]

```
//ユーザがログインしたかどうか確認
if (User.Identity.IsAuthenticated)
{
    //ユーザ名を取得してセッション変数として保存
    var user = User.Identity;
    Session["uid"] = user.Name;

    //ログイン情報を表示
    Label1.Text = "You are logged in as ";
}
else {
    Session["uid"] = "";
    Label1.Text = "Hello Guest! Please sign in to view your profile.";
}
```

[VB]

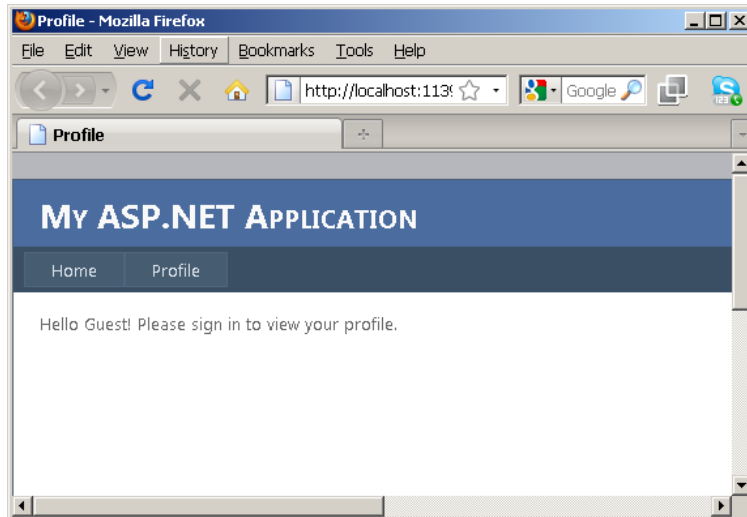
```
‘ユーザがログインしたかどうか確認
If User.Identity.IsAuthenticated Then
    ‘ユーザ名を取得してセッション変数として保存
    Dim user As MembershipUser = Membership.GetUser()
    Session("uid") = user.UserName.ToString
    Label1.Text = "You are logged in as "
Else
```

```

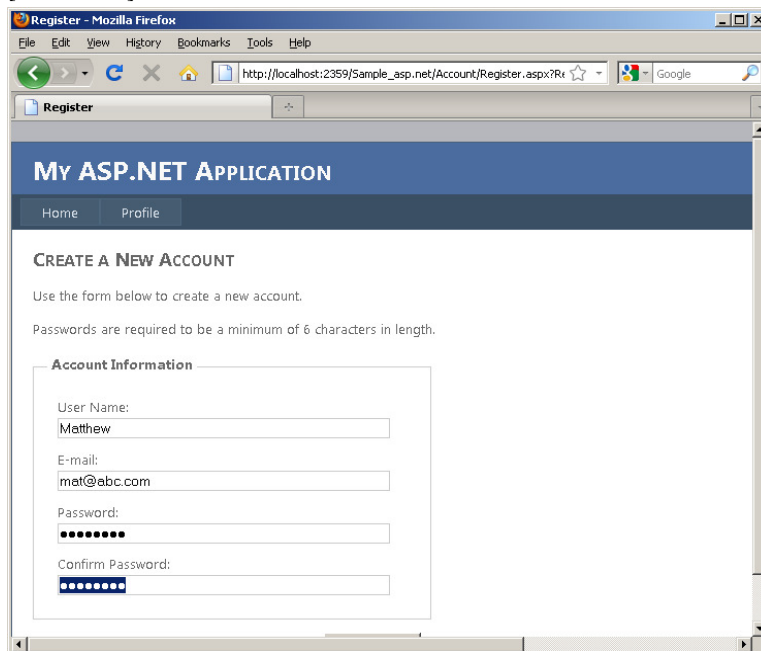
Session("uid") = ""
Label11.Text = "Hello Guest! Please sign in to view your profile." End
If

```

13. [F5] を押すとページが表示されます。ユーザがサインインしていないことを示すメッセージがラベルに表示されます。

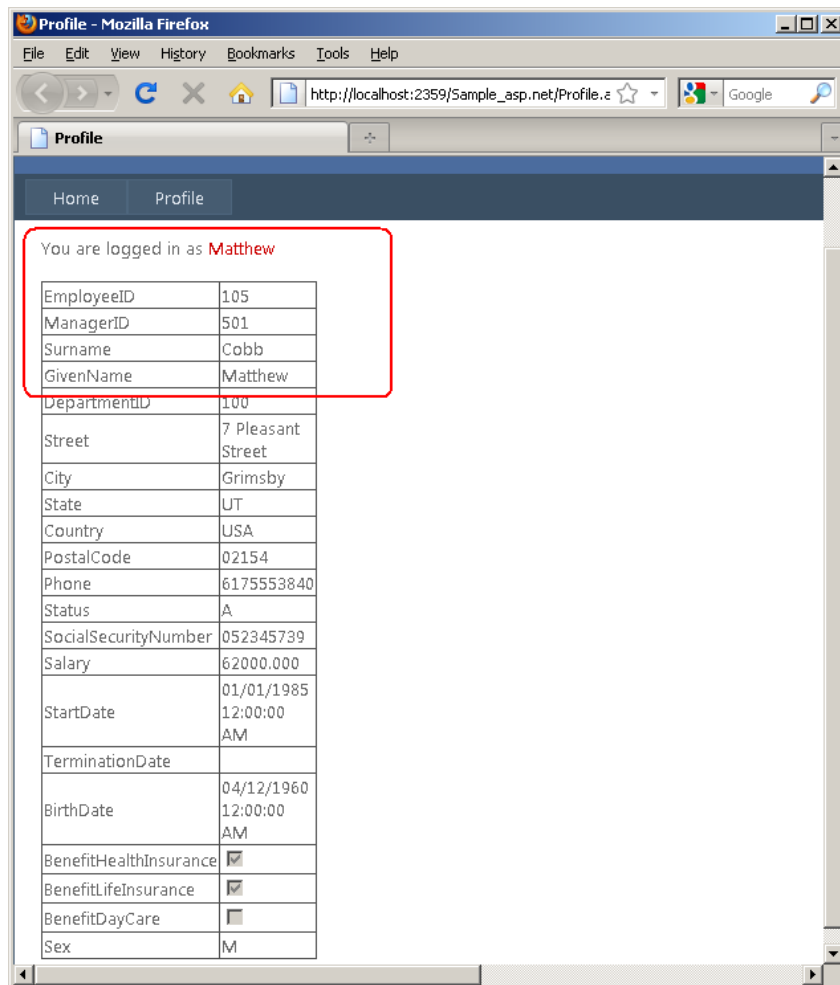


14. ページの右上隅の [Log in] をクリックします。[Register] をクリックし、ユーザ名として“Matthew” (レコードがデモ・データベースにすでに存在する従業員) を入力します。[password] にパスワードを設定します。[Create User] をクリックします。

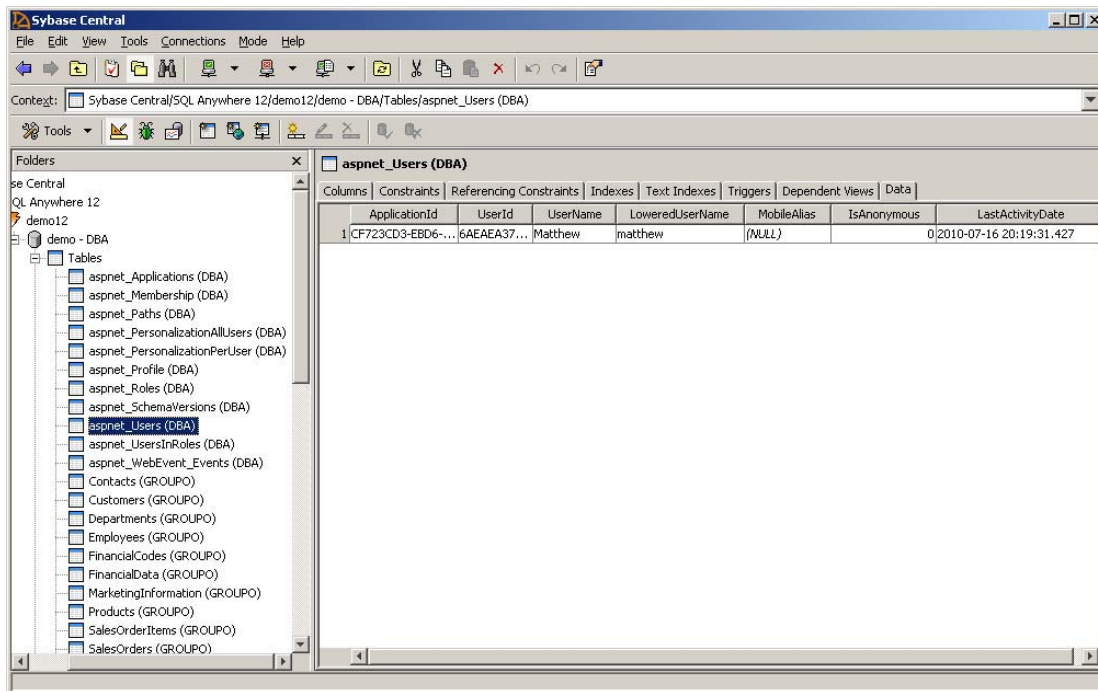


15. ホーム・ページにはパブリック情報が表示され、プロフィール・ページの DetailsView

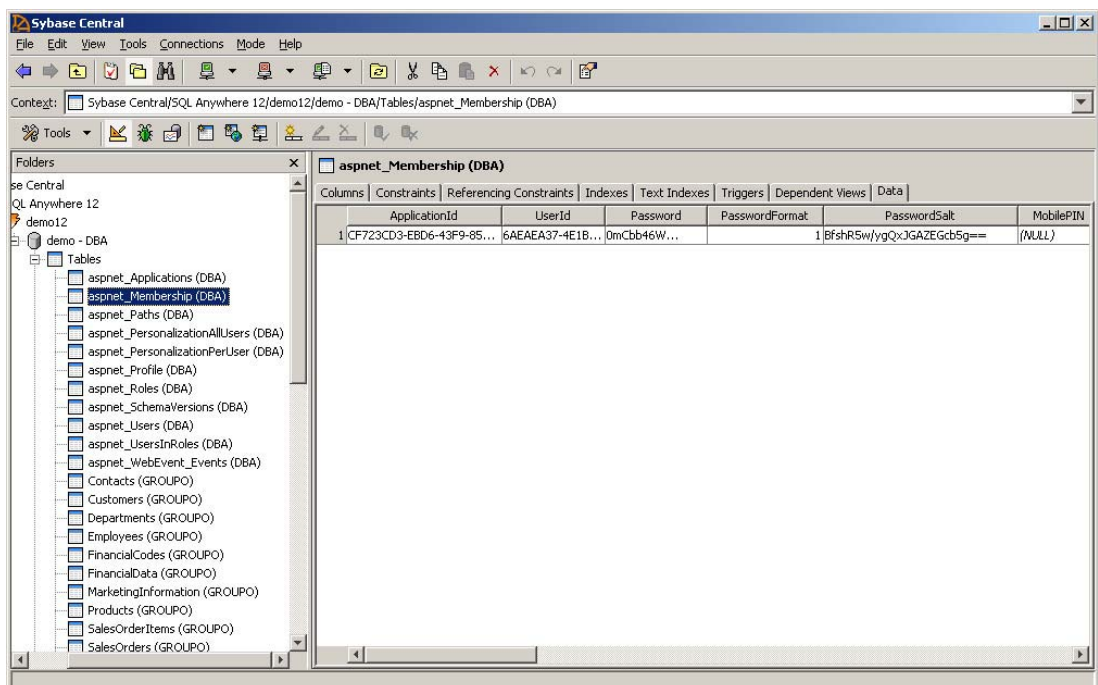
コントロールには、ユーザ Matthew の個人情報が、Employee テーブルのすべてのカラムを使って表示されます。



16. [Log out] をクリックし、再びページにアクセスします。ユーザが認証されていないため、情報が表示されなくなります。
17. Web ブラウザを閉じます。
18. Sybase Central に戻って [Tables] を展開し、テーブル aspnet_users を選択してデータを表示します。ユーザ “Matthew” の認証情報が格納された新しいローが作成されています。



19. テーブル `aspnet_Membership` を選択し、[Data] タブをクリックします。Matthew のメンバシップ情報を格納するための新しいローが追加されています。Profiles Provider など、他の ASP.NET プロバイダを使用する場合、Sybase Central を使用して対応するテーブルに格納されているデータを確認したり管理したりできます。



Dynamic Data 機能の有効化

データ駆動型 Web アプリケーションの場合、通常は、データベースから取得したデータの外觀と動作をカスタマイズする必要があります。Dynamic Data 機能を使用しない場合、希望するフォーマットでデータが表示されるように、プログラマが SQL 文を操作する必要がありますが、これはうんざりする作業であるうえ、コードの読みやすさが低下します。

ASP.NET Dynamic Data は、実行時にデータ・エンティティの外觀と動作を推測し、そこからユーザ・インタフェースの動作を導き出すことにより問題を解決します。ほんの少しの手順とわずかなコードの記述だけで、.NET 4.0 の Dynamic Data 機能を利用して、SQL Anywhere データベースに格納されているデータの表示をフォーマットできます。

手順

1. 単純な Web サイトを実行し、ユーザ Matthew としてログインします。StartDate および BirthDate カラムのデータ・フォーマットが、希望するフォーマットで表示されていません。データベースには時間情報がないため、12:00:00AM という時刻が自動的に生成されます。これら 2 つの日付が標準的な YYYY-MM-DD フォーマットで表示されるように、Employee クラスを修正します。

また、TerminationDate フィールドは空ですが、これはデータベースにレコードが存在しないということです。そこで、データが null の場合に表示されるデフォルト値を設定します。Web アプリケーションを閉じます。

2. Visual Studio に戻り、新しいクラスのファイルをフォルダ App_Code に追加し、この新しいファイルに Employee.cs または Employee.vb という名前を付けます。
3. 追加したファイルに以下のコード行を挿入します。

[C#]

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

namespace Model {

    //Employee 部分クラスを拡張
    [MetadataType(typeof(EmployeeMeta))]
    public partial class Employee {

        public class EmployeeMeta
        {

            //StartDate カラムを再フォーマット
            [DisplayFormat(DataFormatString="{0:yyyy-MM-dd}")]
            public object StartDate { get; set; }

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

            //TerminationDate カラムのデフォルト値を設定
            [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="Model1.SAEntities"
EntityTypeFilter="" Select="">
<WhereParameters>
<asp:SessionParameter DefaultValue="" Name="emp_name"
SessionField="uid" DbType="String" />
</WhereParameters>
</asp:EntityDataSource>

```

これにより、DetailsView コントロールのバインド・フィールドが削除され、Dynamic Data がデータをフォーマットするようになります。

5. Profile.aspx ページの Page_Load イベントに以下の行を追加します。

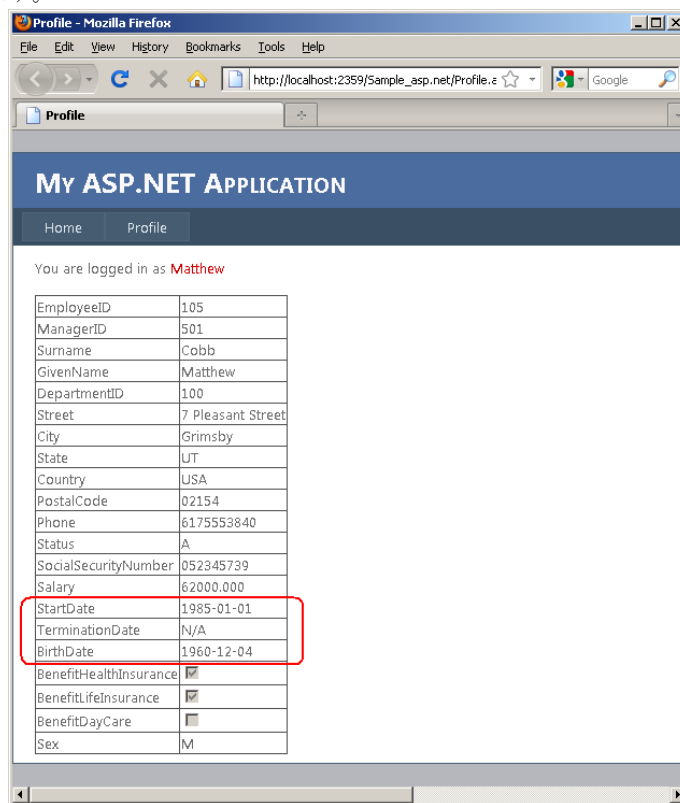
[C#]

```
DetailsView1.EnableDynamicData(typeof(Model1.Employee));
```

[VB]

```
DetailsView1.EnableDynamicData(GetType(Model1.Employee))
```

6. ここで Web サイトを再び実行し、ユーザ Matthew としてログインします。StartDate、BirthDate、および TerminationDate フィールドの表示フォーマットの変更が、Employee コード・ファイル内の注釈に従って行われています。



7. Web ブラウザを閉じます。これでチュートリアルは終了です。

SQL Anywhere ASP.NET プロバイダの削除

デモ・データベースを元の状態に戻したい場合、SQL Anywhere ASP.NET プロバイダをアンインストールする必要があります。

1. インストールの項で説明した [ASP.NET Security Schema Setup] ウィザードを起動し、デモ・データベースに接続します。
2. 削除するすべての機能を選択し、3 番目の手順で [Preserve Data] の選択を解除して、ウィザードを完了します。
3. ウィザードにより、デモ・データベースから SQL Anywhere ASP.NET プロバイダのテーブルが削除されます。

まとめとその他のリソース

SQL Anywhere ASP.NET プロバイダと ASP.NET 4.0 の強化された Web サイト・テンプレートを利用することで、セキュリティに対応した Web サイトを非常に簡単に構築できます。さらに、Visual Studio 2010 用の SQL Anywhere 統合コンポーネントとその Entity Model のサポートは、EntityDataSources をデータバインド・サーバ・コントロールにバインドすることにより、データ駆動型 Web アプリケーションの実装を可能にします。.NET 4.0 の新しい Dynamic Data 機能を使用すれば、マークアップや SQL クエリと格闘する必要なく、部分クラスを拡張するだけで、ビジネス・ロジックを簡単に実装できます。

ホワイトペーパー、チュートリアル、サンプル・コードなどその他のリソースについては、以下のサイトの SQL Anywhere .NET Development Center を参照してください。

<http://www.ianywhere.jp/developers/microsoft-net.html>