WCF と IIS を使用して OData プロデューサーを作成する

IIS（Internet Information Services）向けの OData プロデューサーを作成することは非常に簡単です。私は ADO.NET や IIS についてほとんど知識がなくても、Visual Studio で WCF（Windows Communication Foundation）サービス アプリケーションを作成できました。IIS にその WCF サービス アプリケーションを配置すると、そのアプリケーションは Pervasive PSQL データベースと OData コンシューマー間の仲介役として機能します。

OData プロトコルの詳しい情報については、OData Web サイトをご覧ください。

ここでは、以下の項目について説明します。

環境を設定する

使用した設定は次のとおりです。サンプル プロジェクト ファイル **ODataIISProducer.zip** をダウンロードすれば作業にも役立ちます。

- （IIS で実行する）WCF サービス アプリケーションを作成する開発マシン。データ プロデューサーと同じマシンを使用することもできます。私は Windows 7 デスクトップ システムを使用しました。このシステムには Visual Studio 2010 がインストール済みで、最新の Pervasive PSQL ADO.NET Data Provider をインストールしました。この Data Provider ランタイム コンポーネントは、Pervasive PSQL Client のインストール時にデフォルトでインストールされます。
- データ プロデューサーとして機能するマシン。これは Windows 2012 Server（IIS 8 搭載）で、Pervasive PSQL Vx Server v11 SP3 がインストールされています（Pervasive PSQL Vx Server v11 のどのエディションでも使用できます）。Windows 2012 Server をお持ちでない場合は、IIS 7 または IIS 8 のいずれかをサポートするサーバー プラットフォームを使用することができます。Windows 2008 Server（IIS 7 搭載）上にプロデューサーを配置することも可能でした。

WCF サービス アプリケーションを作成する

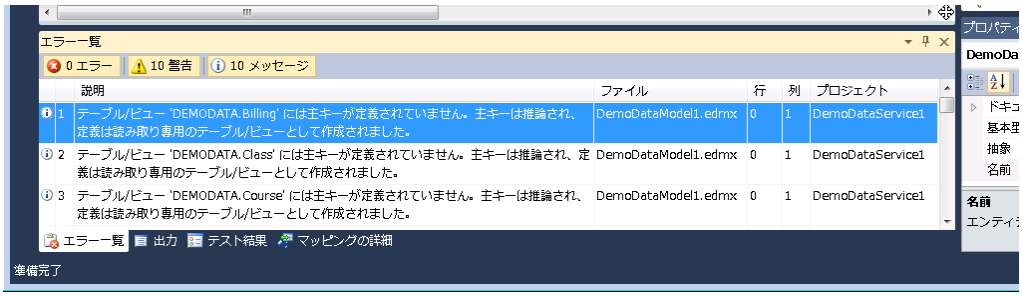
Microsoft では、SQL Server データベースに対する WCF サービス アプリケーションの作成手順を既に作成しています。私は、Microsoft の Web ページ http://msdn.microsoft.com/en-us/data/gg192995.aspx（英語）に記載されている手順に従って作業しました。

ただし、Microsoft の手順をそのまま実行するのではなく、SQL Server の代わりに Pervasive PSQL を使用するよう、手順に注釈を付けました。また、私がアクセスしているデータベースを参照する名前を使用しました。

1. Microsoft の手順と同じ。
2. Microsoft の手順と同じ。
3. Microsoft の手順と同じ。
4. サンプル プロジェクトには、**DemoDataService** という名前を付けました。
5. Microsoft の手順と同じ。
6. モデルの名前を指定します。**DemoDataModel** という名前にしました。
7. Microsoft の手順と同じ。
8. 接続が存在しない場合、[新しい接続]をクリックしてサーバーの情報を入力してください。この接続情報は Web.config ファイルに保存し、後でテキスト エディターで編集することができます。また、ダイアログ下部のテキスト ボックスに表示される名前を書き留めておいてください。この名前は後で使用します。サンプル プロジェクトの名前には **DemoDataEntities** を使用しました。

   この方法で DEMODATA にアクセスできない場合は、お使いの開発マシンに Pervasive PSQL Client がインストールされているかどうかを確認してください。
9. [テーブル]オプションのチェックをオンにしてください。DEMODATA データベースにはビューもストアドプロシージャもありません。

   エンティティが作成されると、[エラー一覧]で主キーに関する一連のメッセージが表示されました。これは次のように見えます。

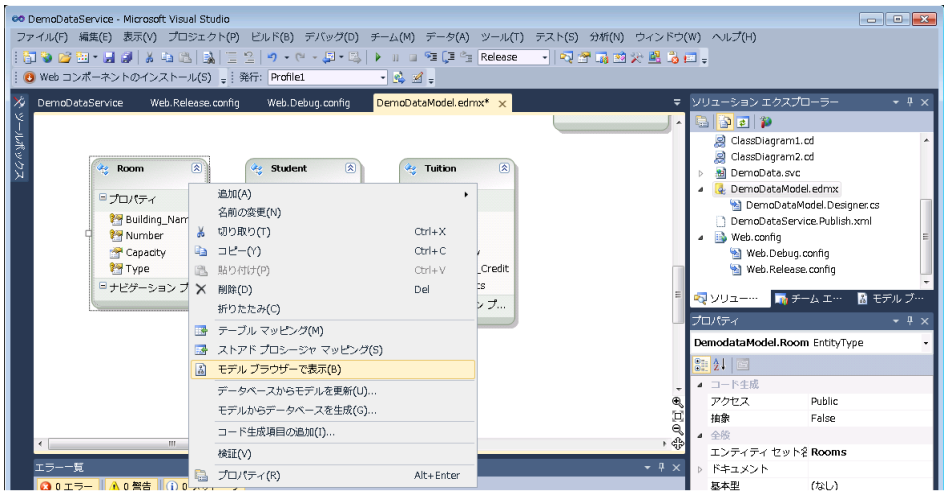


   これらのメッセージは無視しました。このサービスは読み取り専用となるため、DEMODATA のテーブルに対して主キーを定義する必要はありません。

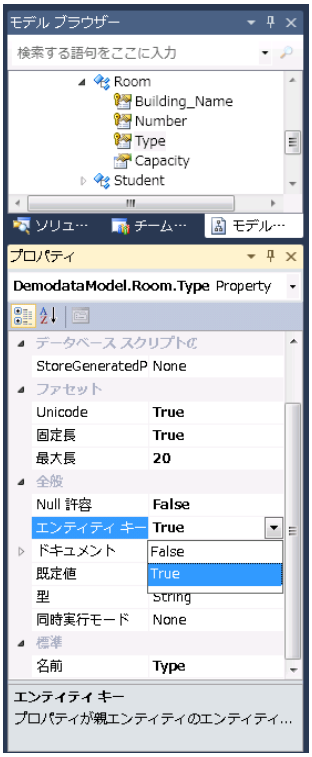
   ただし、問題がある場合は、推測されたキーを調べて修正することができます。たとえば、Room テーブルに対して推測されたキーで問題がある場合、次のように修正します。

   a) ソリューション エクスプローラーで **DemoDataModel.edmx** をダブルクリックします。

   b) Room エンティティを探し、そのエンティティを右クリックして[**モデル ブラウザーで表示**]を選択します。



c) モデル ブラウザーで、Room の下にある **Type** を選択します。[プロパティ]ペインで[**エンティティ キー**]の値をドロップダウンし、"**False**" に変更します。



d) 同じく Room の下にある **Building_Name** および **Number** フィールドに対し、[プロパティ]ペインで[**エンティティ キー**]の値を "**True**" に変更します。これで、キーが Building_Name と Number に設定されました。

10. Microsoft の手順と同じ。

11. [**名前**]フィールドの値を変更します。**DemoData.svc** という名前にしました。

12. Microsoft の手順に記載されているコードをコピーし、Visual Studio へ貼り付けて編集できるようにしました（Microsoft の手順に記載のコードをコピーして貼り付けただけでは、コンパイルしません）。Microsoft の手順とは異なる名前があるので、Visual Studio によって生成されたファイルで次の 2 箇所を変更する必要があります。

    - `/* TODO: put your data source class name here */` というコメントがあるので、この部分を **DemoDataEntities** に置き換えます。
    - **InitializeService** ルーチンに次の 1 行を追加します。

      ```
      config.SetServiceOperationAccessRule("*", ServiceOperationRights.AllRead);
      ```

    私が記述したコードは次のようになりました。

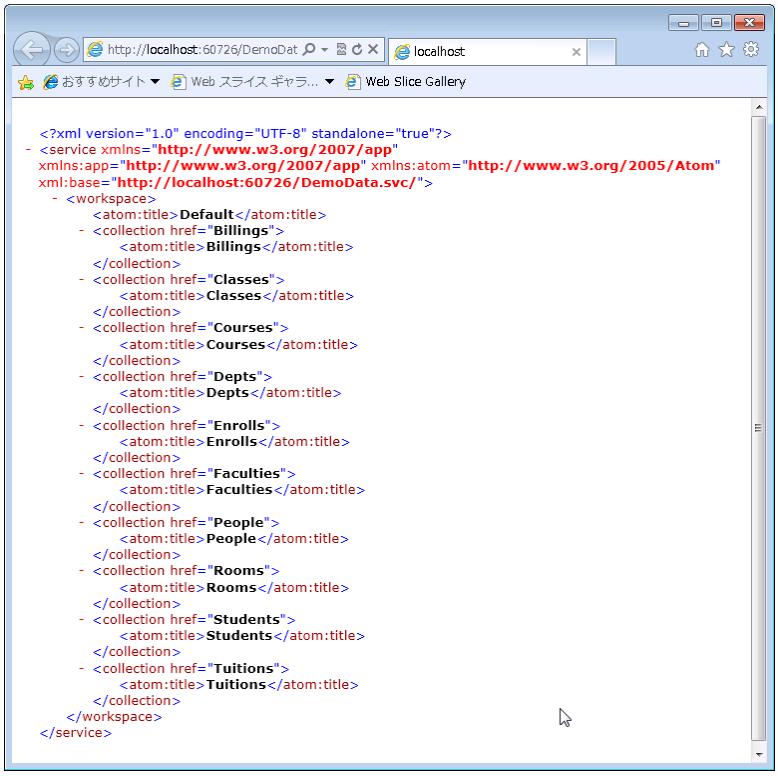
```
using System;
using System.Collections.Generic;
using System.Data.Services;
using System.Data.Services.Common;
using System.Linq;
using System.ServiceModel.Web;
using System.Web;

namespace DemoDataService
{
    public class DemoData :DataService<DemoDataEntities>
    {
            // This method is called only once to initialize service-wide
            policies.
            public static void InitializeService(DataServiceConfiguration config)
            {
                // TODO: set rules to indicate which entity sets and
                // service operations are visible, updatable, etc.
                // Examples:
                // config.SetEntitySetAccessRule("MyEntityset",
                // EntitySetRights.AllRead);
                // config.SetServiceOperationAccessRule
                //("MyServiceOperation", ServiceOperationRights.All);
                config.DataServiceBehavior.MaxProtocolVersion                    =
                DataServiceProtocolVersion.V2;
                config.SetEntitySetAccessRule("*", EntitySetRights.AllRead);

            }
    }
}
```

13. OData が既存の Web プロトコルをベースにして構築されていることを思い出してください。この手順と次の手順では、Web ブラウザーを使用してプロデューサーが動作していることを確認し、何を返すかを示します。

    DemoData.svc を右クリックし、[ブラウザーで表示]を選択します。次のような XML が表示されるはずです。



この XML が見えない場合は、Microsoft の手順に記載されている「Note」を参照してください。

14. 特定のテーブルの全レコードを対象にクエリを実行するには、そのテーブルのエンティティ名を使用します。これらは上の図で記載されています。たとえば、Rooms エンティティの全レコードを対象にクエリを実行するには、次の書式を使用します。

    ```
    http://localhost:<yourPort>/Demodata.svc/Rooms
    ```

    <yourPort> 部分は、手順 13 の Demodata.svc ページを表示するよう選択したときに開いた ASP.NET 開発サーバーのポートに置き換えます。

    単一レコードに対するクエリを最適化するには、まず DEMODATA のテーブルで主キーを定義する必要があります。

配置

http://msdn.microsoft.com/en-us/data/gg192995.aspx で上記以降の手順では、IIS サーバーへのソリューションを配置する別の方法について説明しています。アプリケーションの配置や IIS 構成は非常に複雑であるため、そのトピックに関する解説や提言は行いません。

サーバーへソリューションを配置したら、Web ブラウザーからクエリを実行してみて DEMODATA に対するデータアクセスが動作していたことを確認しました。クエリには、先の手順で示した名前を使用しました。

```
http://<IIS Host Name>/DemoDataService/Demodata.svc/Rooms
```

商用アプリケーションの計画

次に挙げる提案事項は、CRUD（Create、Read、Update、Delete）操作すべてを実行する必要がある、より複雑なデータベース ソリューションを設計する際に役立ちます。

- UPDATE（更新）/INSERT（挿入）/DELETE（削除）操作を必要とするテーブルにはすべて主キーが定義されていること。Windws ストア アプリが行を変更するには、各行を一意に識別できるようにする必要があります。この要件を満たすため、テーブル定義に IDENTITY 列を追加する必要があるかもしれません。
- 固定長の文字フィールドで、その固定長の長さすべてを使用していないデータが含まれる場合、クエリが適切に動作しない可能性があります。たとえば、Rooms テーブルで **Building_Name = 'Young Building'** のエントリをすべて検索するアプリケーションを使用すると、0 行が返ります。Building_Name 列の長さは 25 バイトです。このため、行の検索には **Building_Name = 'Young␣Building␣␣␣␣␣␣␣␣␣␣'** というクエリが必要です。␣ は空白の文字を示します。これは Pervasive PSQL データベースに用いられるその他のプログラミング API とは異なります。

最後に

OData コンシューマーが Pervasive PSQL にアクセスできる OData プロデューサーのセットアップに必要な手続きをお見せしました。IIS 下で実行する WCF サービス アプリケーションで、Pervasive PSQL をデータ リポジトリとしてアクセスすることは、ほかのデータベース アクセスと同様に簡単なプロセスです。必要なものは、Visual Studio、OData および ADO.NET などの業界標準コンポーネントだけです。

関連ドキュメント「Windows ストア アプリから Pervasive PSQL データへアクセスする」も参照してください。

Pervasive Software ソフトウェア開発技術者 Jan D.