JavaScript アプリケーションから Pervasive PSQL データへアクセスする

JavaScript は、インテリジェント Web サイトを作成するために使用する主要なスクリプト言語です。この JavaScript を使用すれば、(Pervasive PSQL の標準的な API クライアント ライブラリを使用できない)モバイル デバイスなどのさまざまなプラットフォーム上で実行可能なアプリケーションを作成することができます。このようなアプリケーションであれば、モバイル デバイスからでも Pervasive PSQL をデータ リポジトリとして使用することができます。

JavaScript を使用して Pervasive PSQL データベースのデータへアクセスする Web ベースのアプリケーションを作成したいのであれば、サーバーとクライアントの両方の要素が必要です。この両方の要素を作成することは難しいことではありません。Pervasive PSQL のサンプル データベースである DEMODATA からデータを取得する基本的な JavaScript アプリケーションはすぐに作成できました。このドキュメントでは、そのアプリケーション作成に使用した手順の概要を以下のセクションに沿って説明します。

Open Data Protocol および JayData について

私は OData(Open Data Protocol)を活用するためにサーバーを実装しました。JavaScript アプリケーション用には、OData をサポートする JayData というオープン ソース パッケージを使用しました。JayData の詳細については、「JavaScript アプリケーション(OData コンシューマー)を作成する」セクションを参照してください。まずは、データ アクセスを可能にするメカニズムである OData について見てみましょう。

OData は、Microsoft Open Specification Promise に従ってリリースされた Web ベースのプロトコルです。これは、標準化された方法でさまざまなソースのデータにアクセスするよう設計されています。OData は、HTTP、Atompub(Atom Publishing Protocol)および JSON(JavaScript Object Notation)など既存の標準 Web テクノロジに基づいて構築されています。OData クライアント ライブラリは、JavaScript などのさまざまなアクセス方法をサポートするデバイス向けに存在しています。このクライアント ライブラリは OData の詳細を抽象化するので、JavaScript アプリケーションを作成することができます(OData プロトコルやライブラリなどの詳細については、OData Web サイト(http://www.odata.org/)を参照してください)。

OData ソリューションは従来のクライアント/サーバー アーキテクチャに似ています。OData の用語で、サーバーは "プロデューサー"、クライアントは "カスタマー" と言います。つまり、JavaScript ソリューションにはプロデューサーとコンシューマーの両方が必要となります。

環境を設定する

JavaScript ソリューションを実装するために 2 つのマシンを使用しました。

- JavaScript アプリケーションを作成する開発マシン。私の場合は、Windows 7 で Visual Studio 2010 を実行しているシステムを使用しました。Visual Studio 2012 を使用してもよいでしょう。このシステムには Pervasive PSQL Client のインストールも必要です。
- IIS（Internet Information Services）8 または IIS 7 がインストールされているサーバー マシン（データ プロデューサー）。私の場合、Windows Server 2008 で IIS 7 および Windows Server 2012 で IIS 8 の両環境を試しました。このマシンでは Pervasive PSQL Vx Server も実行していました。

OData プロデューサーを作成する

これについては別のドキュメントで説明しています。このプロデューサーを作成する手順については、「WCF と IIS を使用して OData プロデューサーを作成する」ドキュメントを参照してください。あるいは、最初にそのドキュメントにおける最終生成物を見ることもできます。これはダウンロード アーカイブ ファイル **ODataIISProducer.zip** を参照してください。ご自分の環境の Pervasive PSQL Vx Server に対応する IP アドレスまたはホスト名を提供するためにプロデューサーを編集する必要があることに注意してください。JavaScript アプリケーションの作成に関する次のセクションは、ODataIISProducer.zip の Visual Studio プロデューサーを使用していることを前提としています。

JavaScript アプリケーション（OData コンシューマー）を作成する

以下に示すのは、シンプルな JavaScript アプリケーションを作成する手順です。OData サポートを提供する JayData Library を使用しました（JayData の詳細については、Web サイト www.jaydata.org を参照してください）。

以下の手順を省略して、まずはこのドキュメントにおける最終生成物を確認することもできます。最終生成物はダウンロード アーカイブ ファイル **ODataJavaScript.zip** で入手可能です。ご自分の環境に合わせていくつか変更する必要があるでしょう。たとえば、データベース接続情報を指定する場合は手順 7 を見てください。

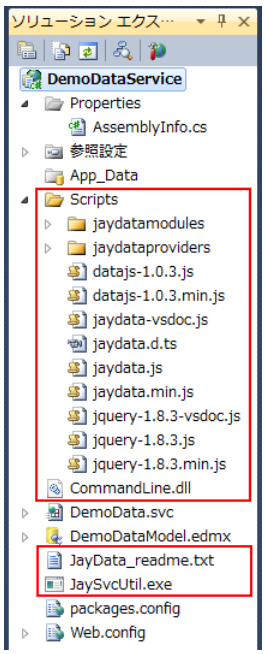
1. アーカイブ ダウンロード ファイル **ODataIISProducer.zip** に含まれている Visual Studio ソリューションを開きます。
2. Visual Studio の NuGet パッケージ マネージャーを使用して JayData パッケージをダウンロードします。
   a) パッケージ マネージャー コンソールを開きます。メニュー バーから［ツール］>［ライブラリ パッケージ マネージャー］>［パッケージ マネージャー コンソール］の順に選択します。
   b) `PM>` プロンプトで `Install-Package JayData` というコマンドを入力します。

   jQuery、datajs および JayData のインストールに関するメッセージが表示されます。私の環境では、コマンドの出力は次のようになりました。

パッケージ マネージャーをこれまで使用したことがない場合は、最初にそのアップデートを促されるかもしれません。その場合はアップデートしてください。

パッケージ マネージャーは最新バージョンをインストールするので、私がダウンロードしたものとは異なるバージョンがダウンロードされる可能性もあります。

ソリューションに追加される新しい項目に注目してください。次のスクリーン ショットで、赤い四角で囲んでいるのがその項目です。

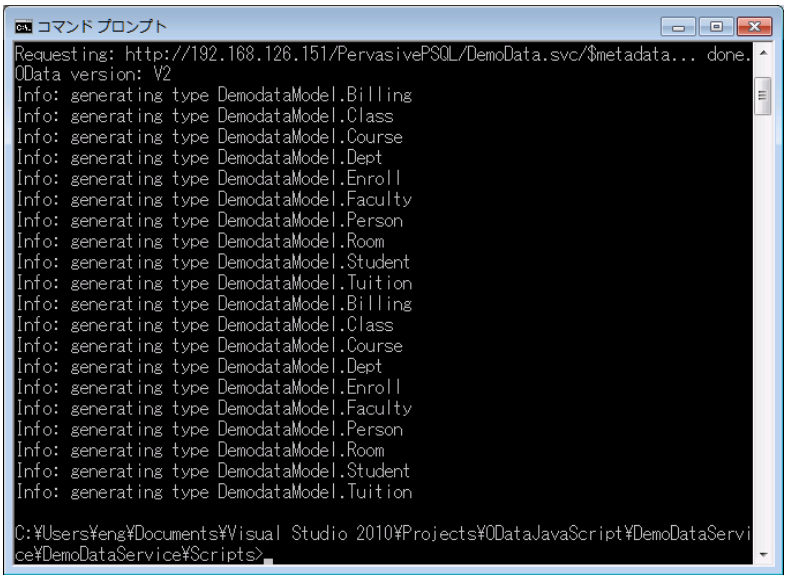


3. ここからが重要です。JayData JaySvcUtil.exe を使用して Pervasive PSQL DEMODATA データベースへのアクセスを可能にする機能を組み込みます。コマンド プロンプトを開き、プロジェクトの Scripts サブディレクトリへ移動して、次のように入力します。

```
..¥JaySvcUtil.exe -m http://192.168.126.151/PervasivePSQL/Demo
Data.svc/$metadata -n DemoData -o demodata.js
```

サーバー名または IP アドレス部分は、ご自分の環境で OData プロデューサーを実行しているマシンのサーバー名または IP アドレスに置き換えてください(「OData プロデューサーを作成する」を参照)。

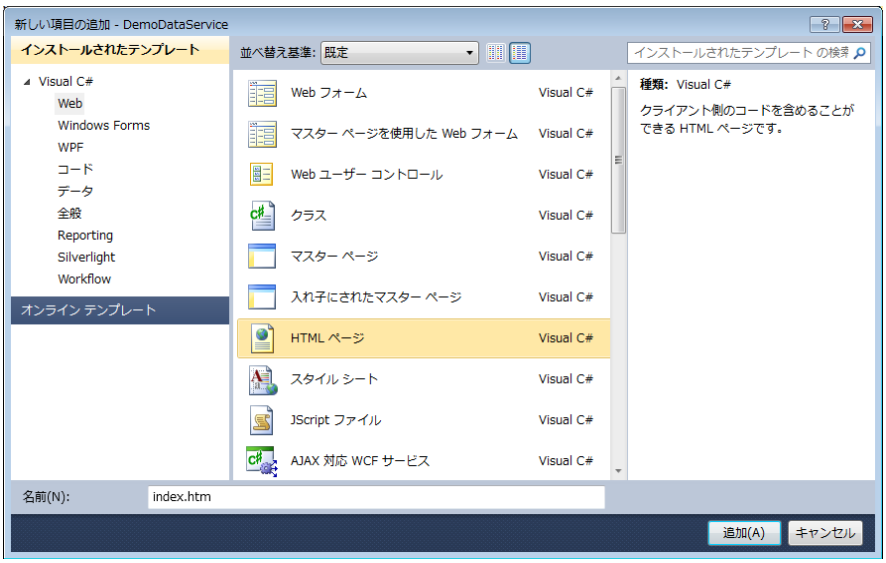
私の環境ではコマンドの実行後、次のような画面になりました。



JayData Web サイトで、この手順は "クライアント コンテキストの作成" と呼んでいます。

このコマンドはプロジェクトの Scripts サブディレクトリに demodata.js ファイルも作成します。私はそのファイルをプロジェクト ダウンロード ファイル **ODataJavaScript.zip** に含めました。

4. これで JavaScript を記述する準備が整いました。ソリューション エクスプローラー ペインで、プロジェクト名を右クリックし、[追加] > [新しい項目]の順に選択します。[新しい項目の追加]ダイアログで、テンプレート タイプに "Web" の "HTML ページ" を選択します。私はこのページに index.htm という名前を付けました。



5. 空の index.htm ドキュメントに JavaScript コードを記述します。<title> </title> タグの前に以下のコードを記述しました。

```
    <script src="Scripts/datajs-1.0.3.js" type="text/javascript"></script>
     <script src="Scripts/jquery-1.8.3.js" type="text/javascript"></script>
     <script src="Scripts/JayData.js" type="text/javascript"></script>
     <script src="Scripts/jaydataproviders/odataprovider.js"
type="text/javascript"></script>
     <script src="Scripts/ demodata.js" type="text/javascript"></script>

     <script>
         OData.defaultHttpClient.enableJsonpCallback = true;

// The next block of code creates the connection to the OData provider,
which allows access to the DEMODATA database
         var demodata = new DemoDataService.DemoDataEntities({
             name: 'oData',
// FIXME!  Change the IP address below to the IP Address or hostname of
your IIS server!
             oDataServiceHost: 'http://
192.168.126.151/PervasivePSQL/DemoData.svc'
         });

// The GetAllRooms() function retrieves all of the rooms from the DEMODATA
Rooms table.  It is the equivalent of SELECT * FROM Rooms
         function GetAllRooms() {

// The next statement places all of the rooms into an array
             var roomsList = demodata.Rooms.toArray();
             var roomsInfo;

             $.when(roomsList, $).then(function (roomsList) {

                 var tableHeader = "<table>  <thead> <tr> <th
style=¥"width:150px;text-align:center¥">Building</th> <th
style=¥"width:75px;text-align:center¥">Number</th> <th
style=¥"width:75px;text-align:center¥">Type</th> <th
style=¥"width:75px;text-align:center¥">Capacity</th> </tr> </thead>
<tbody>";
                 var tableFooter = "</tbody> </table>";
                 var roomsInfo = tableHeader;

// The next block of code iterates through the list of rooms retrieved from
the provider and constructs the HTML to display the list on the application
Web page
                 roomsList.forEach(function (classroom) {
                     var item = "<tr><td style=¥"width:150px;text-
align:center¥">@building</td> <td style=¥"width:75px;text-
align:center¥">@roomnum</td> <td style=¥"width:75px;text-
align:center¥">@roomtype</td> <td style=¥"width:75px;text-
align:center¥">@capacity</td> </tr>"
                         .replace("@building", classroom.Building_Name)
                         .replace("@roomnum", classroom.Number)
                         .replace("@roomtype", classroom.Type)
                         .replace("@capacity", classroom.Capacity);

                     roomsInfo = roomsInfo + item;
                 });

                 roomsInfo = roomsInfo + tableFooter;

                 document.getElementById("pvswheading").innerHTML = "Rooms
at Pervasive University";
```

```
                document.getElementById("pvswrooms").innerHTML = roomsInfo;
            });
        };


// The GetRoomsCapacity() function retrieves a list of rooms with a minimum
capacity as specified on the application Web page
        function GetRoomsCapacity() {


            input = document.getElementById("fieldmincapacity").value;

            if (isNaN(input)) {
                alert(input + " is not a valid number");
                return;
            }

            mincapacity = parseInt(input);

// The next block of code is equivalent to SELECT * FROM Rooms WHERE
capacity >= <user-supplied value> ORDER BY capacity
            var roomsList = demodata.Rooms
                            .filter(function (room) { return room.Capacity
>= mincapacity })
                            .orderBy(function (room) {return
room.Capacity})
                            .toArray();

            $.when(roomsList, $).then(function (roomsList) {

                var tableHeader = "<table>  <thead> <tr> <th
style=¥"width:150px;text-align:center¥">Building</th> <th
style=¥"width:75px;text-align:center¥">Number</th> <th
style=¥"width:75px;text-align:center¥">Type</th> <th
style=¥"width:75px;text-align:center¥">Capacity</th> </tr> </thead>
<tbody>";
                var tableFooter = "</tbody> </table>";
                var roomsInfo = tableHeader;

                roomsList.forEach(function (classroom) {
                    var item = "<tr><td style=¥"width:150px;text-
align:center¥">@building</td> <td style=¥"width:75px;text-
align:center¥">@roomnum</td> <td style=¥"width:75px;text-
align:center¥">@roomtype</td> <td style=¥"width:75px;text-
align:center¥">@capacity</td> </tr>"
                        .replace("@building", classroom.Building_Name)
                        .replace("@roomnum", classroom.Number)
                        .replace("@roomtype", classroom.Type)
                        .replace("@capacity", classroom.Capacity);

                    roomsInfo = roomsInfo + item;
                });
                roomsInfo = roomsInfo + tableFooter;

                document.getElementById("pvswheading").innerHTML = "Rooms
at Pervasive University with a minimum capacity of " + mincapacity;

                document.getElementById("pvswrooms").innerHTML = roomsInfo;
            });
        };
    </script>
```

6. index.htm ファイルの最下部近くにある <body> </body> タグの間には次のコードを記述しました。このコードは Javascript アプリケーションのユーザー インターフェイスで、手順 5 で記述した関数を呼び出します。

```
    <h1>Sample Javascript Rooms Application </h1>
        Show All Rooms: <button onclick="GetAllRooms()">All
Rooms</button><br><br />
        Rooms with a Minimum Capacity: <input type="text"
id="fieldmincapacity" value=10>
        <button onclick="GetRoomsCapacity()">Get</button><br />

        <h1 id="pvswheading"> </h1>
            <ul id="pvswrooms"> </ul>
```

7. ソリューションを配置する前に、そのソリューションで使用される IP アドレスまたはホスト名が正しい場所を指定していることを確認してください。これは、手順に従って作業するのではく、Visual Studio ソリューションをダウンロードしていた場合には特に重要です。

   該当する場所の 1 つは index.htm ファイルです。次の抜粋コードを見てください。

```
        var demodata = new DemoDataService.DemoDataEntities({

                    name: 'oData',
// FIXME!  Change the IP address below to the IP Address or hostname
of your IIS server!
                oDataServiceHost: 'http://
192.168.126.151/PervasivePSQL/DemoData.svc'
    });
```

   IP アドレス部分を、ご自分の環境にある IIS サーバーの IP アドレスまたはホスト名に変更してください。

   該当するもう 1 つの場所は web.config ファイルです。<connectionStrings> タグ内で、Pervasive PSQL Client ADO.NET 接続用の接続文字列があります。Host= プロパティを探して、ホスト名または IP アドレス部分をご自分の環境にある Pervasive PSQL Vx Server のホスト名または IP アドレスに変更してください。

8. これでソリューションを IIS に配置することができます。この手順の詳細については説明しません。環境に応じてカスタマイズされる IIS での設定は無数にあるため、それらすべてをこのドキュメントで述べることはできません。また、その環境によってソリューションの配置にもさまざまな方法があります。

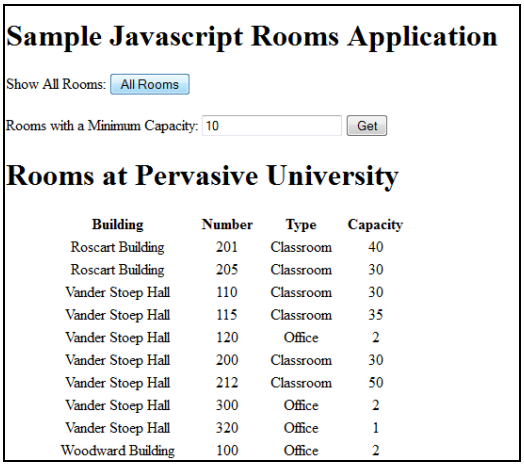
   しかしながら、次のような手順が一般的です。

   - ファイルを IIS へ発行する
   - ソリューションをアプリケーションに変換する
   - ソリューションに、適切に設定されたファイル権限、ユーザー権限、構成されたユーザーがあることを確認する

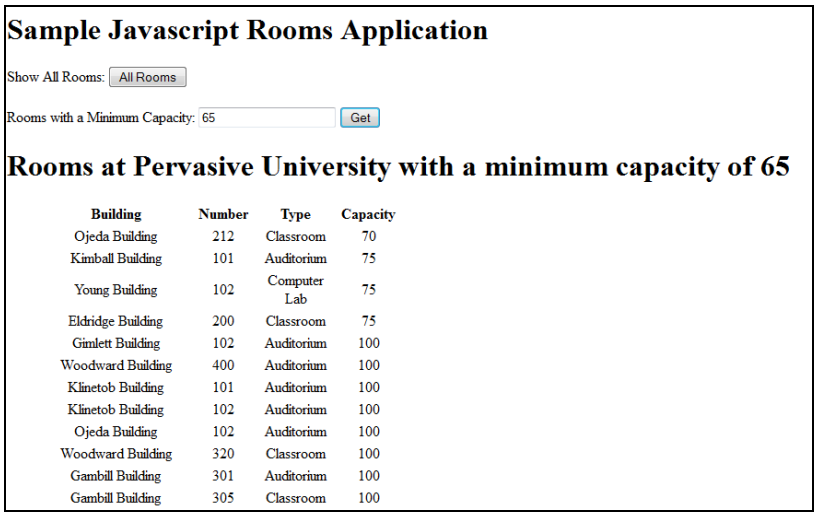
9. Web ブラウザーでアプリケーションを実行します。私の環境では URL として http://<hostname>/PervasivePSQL/ を指定しました。

ご自分の環境でこの JavaScript アプリケーションを実行すると次のような結果になります。

ご覧のとおり、2 つのオプションがある非常にシンプルなアプリケーションです。最初のオプションは、DEMODATA Room テーブルのすべての部屋を一覧表示します。もう 1 つのオプションは、最少収容人数に対応する部屋のみを表示します。[All Rooms]をクリックしたときに出力されるビューの一部は次のとおりです。



最少収容人数 65 を入力して[Get]をクリックすると、出力の先頭は次のように表示されます。



最後に

ここでは JavaScript アプリケーションが Pervasive PSQL Vx Server データベースへアクセスできるようにする方法の一例を示しました。これにより、Pervasive PSQL の標準的な API クライアント ライブラリを使用できないモバイル デバイスなど、多様なプラットフォーム上で実行可能なアプリケーションを作成できる可能性が生まれました。JayData についてはさらに詳しく調べてみてください。なぜなら、JayData には他にもデータの照会やキャッシュ、およびいくつかのユーザー インターフェイス ライブラリとの統合をより簡単に行うための組み込み機能があるからです。

Pervasive Software ソフトウェア開発技術者 Jan D.