The Use of Microsoft based Technologies for the benefit of the Community

Prof. Avi Mendelson – Microsoft  
-- avim@microsoft.com

Prof. Yehuda Afek – afek@tau.ac.il

Mr. Hillel Avni - hillel.avni@gmail.com
Agenda

- Linq to SQL
General

- We would like to create a database on the phone
  - To locate it in the ISOStorage
  - To manipulate it via Linq
  - To synchronize it with a database on the WEB
An option for persisting data from your Windows Phone application is via LINQ to SQL.

LINQ to SQL is only available to applications written for Windows Phone OS 7.1, allowing them to read and write to a relational data store. Whilst behind the scenes data is stored in a SQL Server Compact database, there is no support for ADO.NET, or the ability to issue Transact-SQL (DDL or DML) statements directly to the database.
Creating the project

This example uses Windows Phone application. Please note that we use 7.1 (Mango) setup.
Creating the DB to be used.

There are two ways to do that

1. Define the properties of the DB in the program and the program will generate the DB, initiate it and use it.

2. Define the DB and connect it to the system.
   1. Please note that there are many ways to create the DB, if you use a tool which is not defined as part of the Phone development tools, make sure that the DB type is Compact SQL version 3.5.
   2. Server SQL and Azure SQK will be able to share it.
   3. There are tools that know how to use DB of SQL 4.0 but this is out of the scope of this discussion.
Creating the DB – cont.

- Right click on “data Connections”
- Choose “add connection” (not new DB)
- Select the place you like to place the DB.
- Please note that although the prefix is “sdf” it can be read by any SQL server
- Hit “create”
- Hit OK
- Now you can see the DB in the Server Explorer TAB
Create Tables

- Open the DB tab and right click on the table
- Add table name Product
- Select the primary key
Create another table -- Order

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Length</th>
<th>Allow Nulls</th>
<th>Unique</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderID</td>
<td>uniqueidentifi...</td>
<td>16</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Name</td>
<td>nvarchar</td>
<td>100</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ProductID</td>
<td>uniqueidentifi...</td>
<td>16</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CustomerName</td>
<td>nvarchar</td>
<td>100</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Quantity</td>
<td>int</td>
<td>4</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Using relations between the Tables

- We like to correlate the ProductID in the two tables.
- Expand Table
- Right-Click the Order table and select Table Properties and then click on the Add Relations tab.
- Fill the fields respectively and hit the “add columns”
- Hit the “Add Relations”
- OK
- Now the connection was defined.
Adding Linq interfaces to the DB

- You can build the interfaces your self or use an existing tools which is provided as part of the “visual studio 2010” called SqlMetal.
- Start -> All Programs -> Visual Studio 2010 -> Visual Studio Tools -> Visual Studio Command Prompt
- Move to the folder into which you created the SQL Server Compact database file.
- sqlmetal productsales.sdf /code:productsales.cs /pluralize
Add references and class

- Add the newly created productsales.cs file via “Add Existing Item”.
- Add a reference to the class library System.Data.Linq.dll (via the Add Reference option from the Solution Explorer window).
Comment out containers not needed

- Only string connection is needed
- Now you can compile it
Create the DB in the ISOStorage

```csharp
using System.Windows.Media;  
using System.Windows.Shapes;  
using Microsoft.Phone.Controls;

namespace Linq2SQL
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();

            string DBConnectionString = "Data Source=Isostore://SalesData.sdf";  
            using (var db = new Productsales(DBConnectionString))
            {
                if (db.DatabaseExists() == false)
                {
                    db.CreateDatabase();
                }
            }
        }
    }
}
```
Create and initialize the Table

- Please note that the last insert below is using Add since this is a foreign key.
DB Query

- We like to search the DB using Linq
Code and output

```
var order = new Order()
    {
        OrderID = Guid.NewGuid(),
        CustomerName = "Tom",
        Quantity = 5
    };

//    db.Orders.InsertOnSubmit(order);
prod.Orders.Add(order);
    db.SubmitChanges();

private void button1_Click(object sender, RoutedEventArgs e)
{
    using (var do = new Productsales(DBConnectionString))
    {
        var expensiveProducts = from product in do.Products
                                where product.Price > 10
                                select product;
        foreach (var prod in expensiveProducts)
        {
            textBox1.Text += prod.Name + " = " + prod.Price + "\n";
        }
    }
}
```
Using local DB as a cache

- There are few tools that allow you to do that
- But this simplest way is
  - Create a local SQL-DB
  - Create a similar DB on a server or cloud (AZURE)
  - Option-1: each time you change a field on the local DB modify it on the global DB as well
  - Option-2:
    - Add a field to each record that indicate if you change it.
    - Periodically update all modified fields to the global and “clean” the modified field
    - You can also count how many modified fields exist and update only if it exceed some number.
  - You can also add an indication of last time accessed and clean each entry that aged.