Fixed Queries

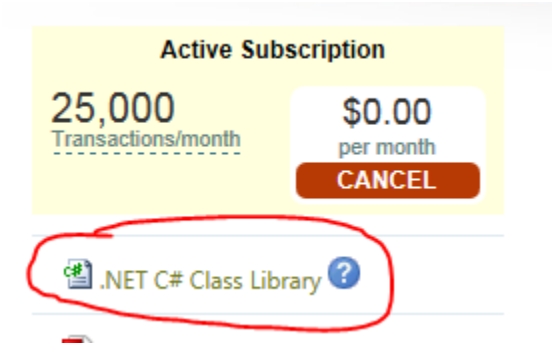
Fixed queries are one of two methods to retrieve data from the Azure Data Market. Fixed queries are those which use predefined operations for the manipulation of data; queries can be made either via URI or the use of a service client class library (.NET only). Here's how to get started with Fixed Queries in a Windows Phone 7 application, using the service client class.

Typically you don't know if a dataset supports fixed or flexible queries until you've located it. The Details tab gives us information about the URI, type of query supported, and the methods, parameters and properties available to us. This information is all very important to us—if you need a refresher, read the first part of the flexible query workbook.

For simplicity, this sample is also using the basic authentication, rather than OAuth 2.0. If you're comfortable with OAuth, you can swap out authentication schemes if you want. This sample is the simpler of the two, since I give it second in the presentation. You may want to run through the flexible query sample first of you are unfamiliar with some of the steps below.

The Service Class

At the current time, Visual Studio 2010's Add Service Reference does not work with DataMarket fixed queries. So, for every fixed query, a client library is provided for us. The link for the .NET C# Class Library is found in the right margin, just below the subscription pricing, but only after you sign up for the dataset.



Creating a Fixed Query Application

In a nutshell, here is how to create a fixed query application for the Windows Phone. If you read the code sample from the top down, this is what you'll see:

1. Sign up for the datset, and download the service class.
2. Start new Windows Phone application, and import the service class.
3. Add a "using" statement for the service class to the class you're performing the data access in (e.g., page class or ViewModel).
4. Add a project reference and using statement for System.Data.Services.Client.
5. Build. Not a requirement, but helps.

6. Add class level variables for the data container (this will be specified by the service class), and constants for the service URI, email and account key.
7. In constructor, instantiate the URI, container and credentials.
8. Since we're using Basic Authentication, we need to add the user ID and account key o every request. We do so by adding these to the header of every request by adding a SendingRequest event handler, and creating a method to add the credentials into the header.
9. Create the query object, and then execute the query using BeginExecute, providing a callback and a state object.
10. In callback, we need to pull processing back to the UI thread, since callbacks don't operate on that thread. Otherwise, we can't update the UI or interact with the other objects on that thread (such as the ViewModels or observable properties). We then process the results as desired.