

# No Developer Left Behind Workbook

---

## Signing Up for Azure Data Market

Once you've found the data set you want to work with, we need to sign up to use it. If this is your first time in the Azure Data Market, this is a two-step process. If you already have a data set, it's only a one-step process.

### Step 1: Register for Azure Data Market

Before we can use any Azure Data Market data set, we need to register with the Azure Data Market. We only have to do this the first time, so if you already are using Azure Data Market, you can probably skip to step 2.

On our data set is a big green Sign Up button.



DATA.GOV

2006 - 2008 Crime in the United States  
(Data.gov)  
data

Published by: Government of the United States of America  
Categories: Science & Statistics  
Date added: 10/26/2010  
[Get support for this offering](#)

Subscription\* \$0.00  
(Unlimited tx/month)

**SIGN UP**

\*Each page of results returned from a query uses a single transaction(tx) and will count toward your transaction limit. A page of results may return up to 100 records, but will never return more than 100 records.

When you click the Sign Up button, you'll be asked to either log in with an existing Windows Live ID, or create a new one. Do what you need to here—remember that if you're signing up for a service that may incur charges, this Live ID will be used for the billing information.

After you've logged in (and created a Live ID, if necessary), you'll be presented with another registration screen. This one creates your Azure DataMarket account, and only needs to be completed the first time. You won't see this when you sign up for additional datasets.

# Registration

Please enter these additional details for your Windows Azure Marketplace account.

Your privacy is important to us, for more information check out our [privacy statement](#).

## ACCOUNT DETAILS

First name:\*

Last or family name:\*

Organization name:\*

E-mail address:\*

Country/Region:\*

**\* Required Fields**

I agree that Microsoft may use my email address to provide information and offers regarding Windows Azure Marketplace.

CONTINUE

Next up is the usual gigantic page of legalese. These are the DataMarket Terms and Conditions, which you'll have to agree to if you want to consume data. For our sample apps, we're unlikely to run afoul of these, but if you're an ISV, you might want to read these over. This is also a first-time-only requirement, until they change.

# Registration

You must accept the terms of use before your account will be created in Windows® Azure™ Marketplace. If you do not accept the terms of use, your account registration will not be completed. You may print the terms of use by clicking the Print button below. The terms of use are available at [Terms of Use](#).

Print

## MICROSOFT WINDOWS AZURE MARKETPLACE

### AGREEMENT

DATED: September 2011

THANK YOU FOR CHOOSING MICROSOFT WINDOWS AZURE MARKETPLACE! Please read the following agreement carefully.

This Microsoft Windows Azure Marketplace Agreement ("Agreement") is a contract between you, corporation, or other legal entity ("you") and Microsoft Corporation, One Microsoft Way, Redmond, Washington, USA, or based on your primary headquarters, one of the Microsoft Corporation affiliates identified in Section 15 ("Microsoft," "we," "us" or "our"). This Agreement applies to your access and use of the Microsoft online marketplace "Microsoft Windows Azure Marketplace", including any Windows Azure Marketplace services, catalogues, data, applications, software, sites, information, materials, content and updates made available to you as part of such marketplace ("Marketplace Service").

By checking the box, below, you acknowledge that you have read this Agreement and agree to its terms, including the electronic delivery of notices and disclosures in connection with the Marketplace Service. If you do not accept these terms or do not want to enter into this Agreement with Microsoft, please close this web page and exit this site.

#### 1. HOW YOU MAY USE THE MARKETPLACE SERVICES

I accept the terms of use.

◀ Previous

REGISTER

## Step 2: Registering for the Dataset

Now that we've signed up for DataMarket, we need to agree to the terms and conditions of our dataset. Again, for our sample app using a free dataset, we're not likely to run afoul of these, but pay attention if there is money involved, or if the dataset is possibly copyrighted (e.g., non-government, like baseball stats or Zillow). You will see one of these forms for every dataset you sign up for.

# Subscription

◀ Return to Dataset

Dataset Details

2006 - 2008 Crime in the United States (Data.gov)

Published by: Government of the United States of America

Pricing

Subscription  
\$0.00

DATA TERMS

**2006 - 2008 CRIME IN THE UNITED STATES (DATA.GOV)**  
Government of the United States of America  
data  
[View Offer Terms](#)

I have read and agree to the above publisher's Offer Terms and Privacy Policy.

◀ Cancel

We can now start consuming our dataset. The Thank You page has links to some useful resources for us, notably the DataSet explorer, and using the dataset in Visual Studio. You can access a similar page from Account >> My Data, too.

# Thank You

RECEIPT

 Print

Dataset: 2006 - 2008 Crime in the United States (Data.gov)  
Published by: Government of the United States of America  
Type: Subscription

Windows Azure Marketplace  
<https://datamarket.azure.com>

You can cancel your service at any time by visiting <https://datamarket.azure.com/account/datasets> or by calling 866-MSONLINE (676-6546).

**.NET C# Class Library**   
Use Add Service Reference in Visual Studio to create client classes



## EXPLORE THIS DATASET

Launch the DataMarket service explorer which allows you to browse the dataset interactively.



## MICROSOFT POWERPIVOT FOR EXCEL 2010

Learn how to use Microsoft PowerPivot for Excel 2010 with this dataset. PowerPivot gives users the power to create compelling self-service BI solutions, facilitates sharing and collaboration on user-generated BI solutions in a Microsoft SharePoint Server 2010 environment, and enables IT organizations to increase operational efficiencies through Microsoft SQL Server 2008 R2-based management tools.



## TABLEAU SOFTWARE

Learn how to visualize this dataset in Tableau. Tableau provides drag-and-drop data visualization based on best practices and patented technology from Stanford University. Tableau allows you to publish dashboards to the web with one click. It's rapid-fire business intelligence that anyone can use.



## LEARN HOW TO USE THIS DATA IN VISUAL STUDIO

Learn how to seamlessly consume DataMarket data in Visual Studio. Visual Studio provides strongly-typed data access and full IntelliSense support to speed and ease development.

## Finding Your Account Key

Azure Data Market supports two methods of authentication—a basic form, using an API key, and OAuth 2.0. The basic form is simpler and is good for our needs, so we can use that.

Every account in the Data Market is assigned a unique API Key. You can add additional keys to your account if you want to, perhaps one key per application you make.

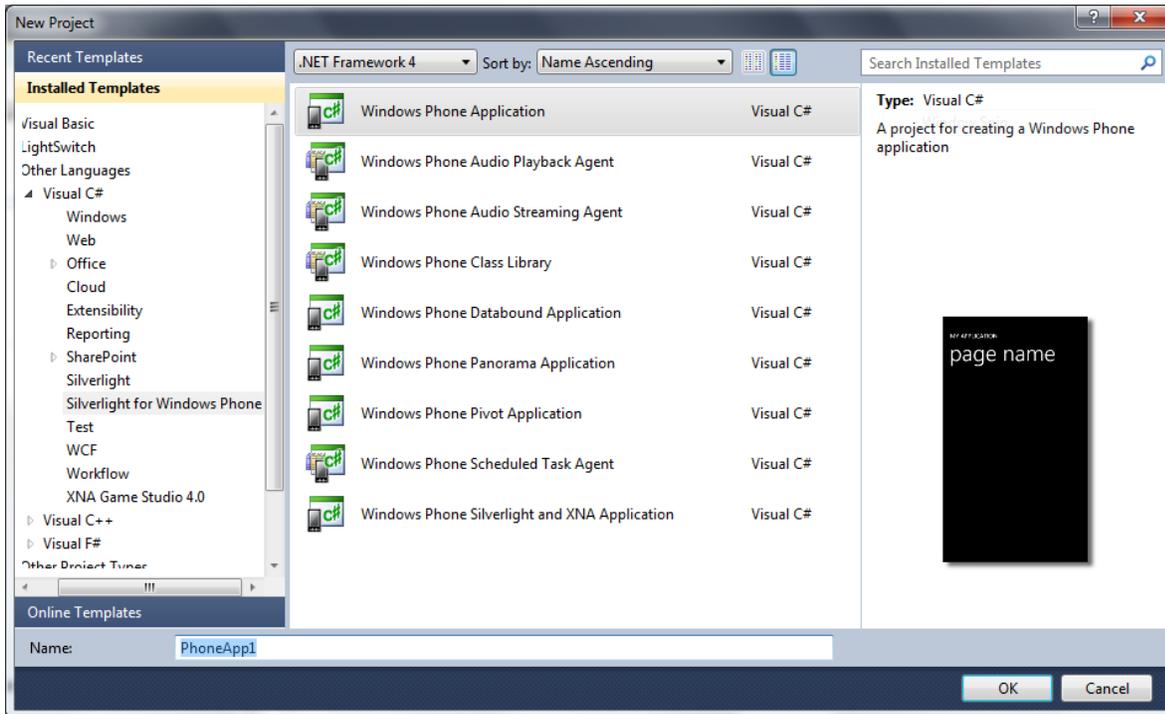
To find your key, log in to the Data Market, click on the My Account button on the top menu, then Account Keys on the left hand menu. You'll see the key listed there.

One quirky thing about the Data Market authentication is that when we have to supply a username (in the form of an email address) and the API key as the password, but the system ignores the email address. That means we can supply a phony email address if we want to.

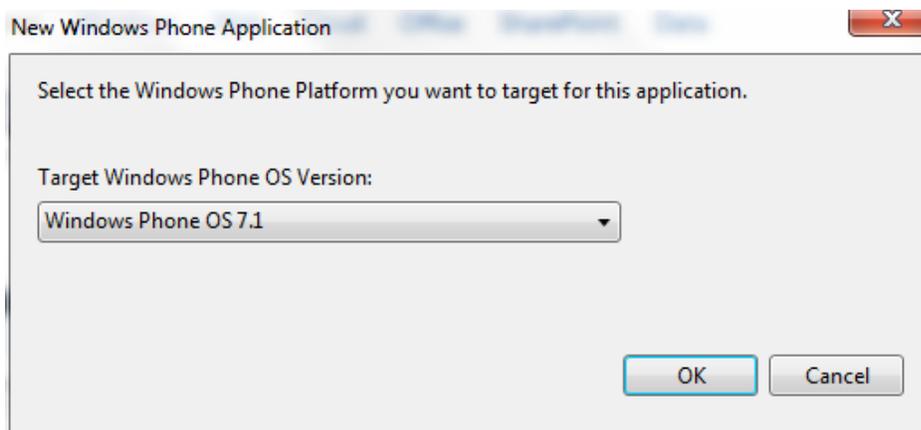
## Code Snippets

### Step 1: Create a new project

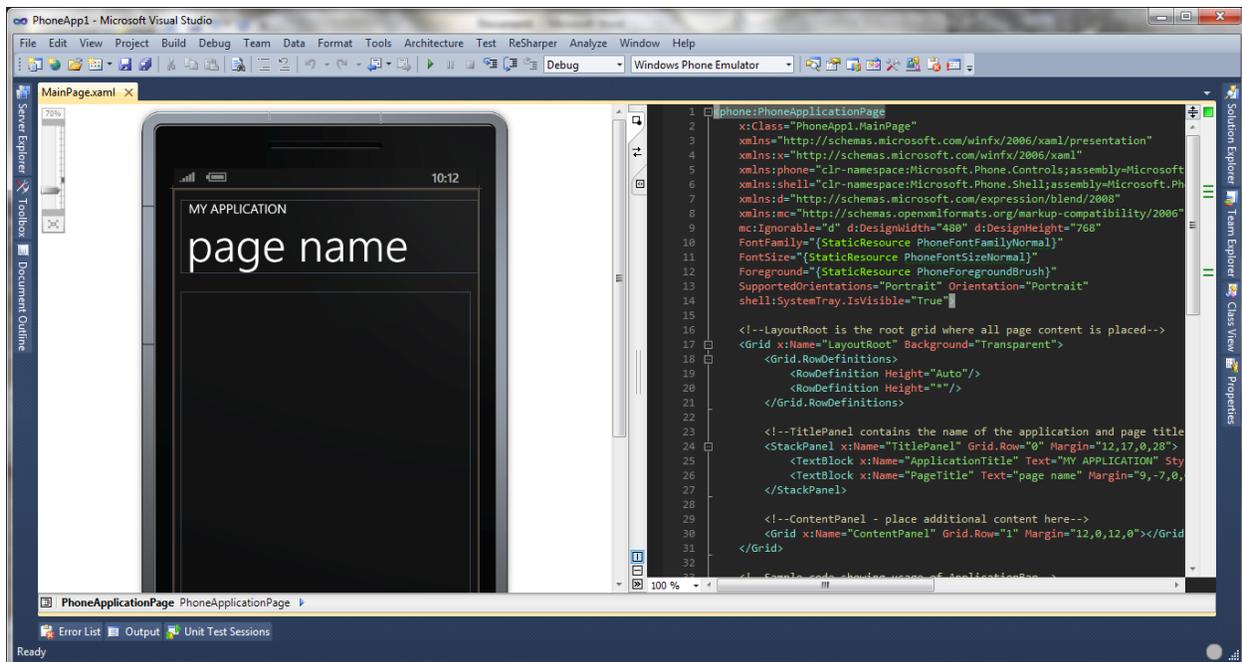
Open Visual Studio, choose File >> New Project. Under Visual C#, select Silverlight for Windows Phone 7. We'll start with a Windows Phone Application. Name it whatever you like, this is just practice.



You will be asked which version of the OS you want to use. For all new applications, chose the highest version—it's the most recent. In our case, we want 7.1.

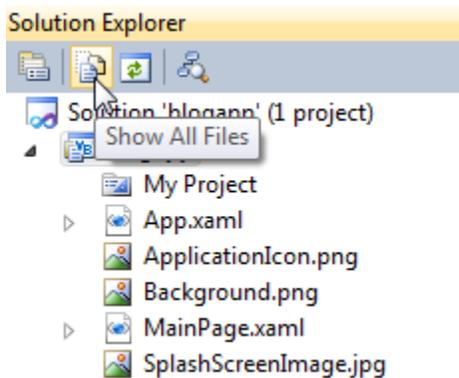


If you see something similar to below, success!

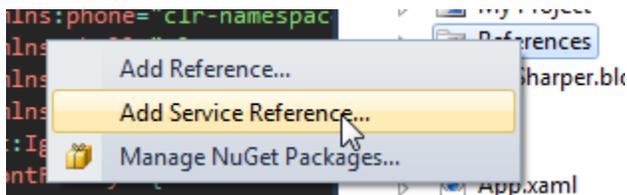


## Step 2: Add a Service Reference

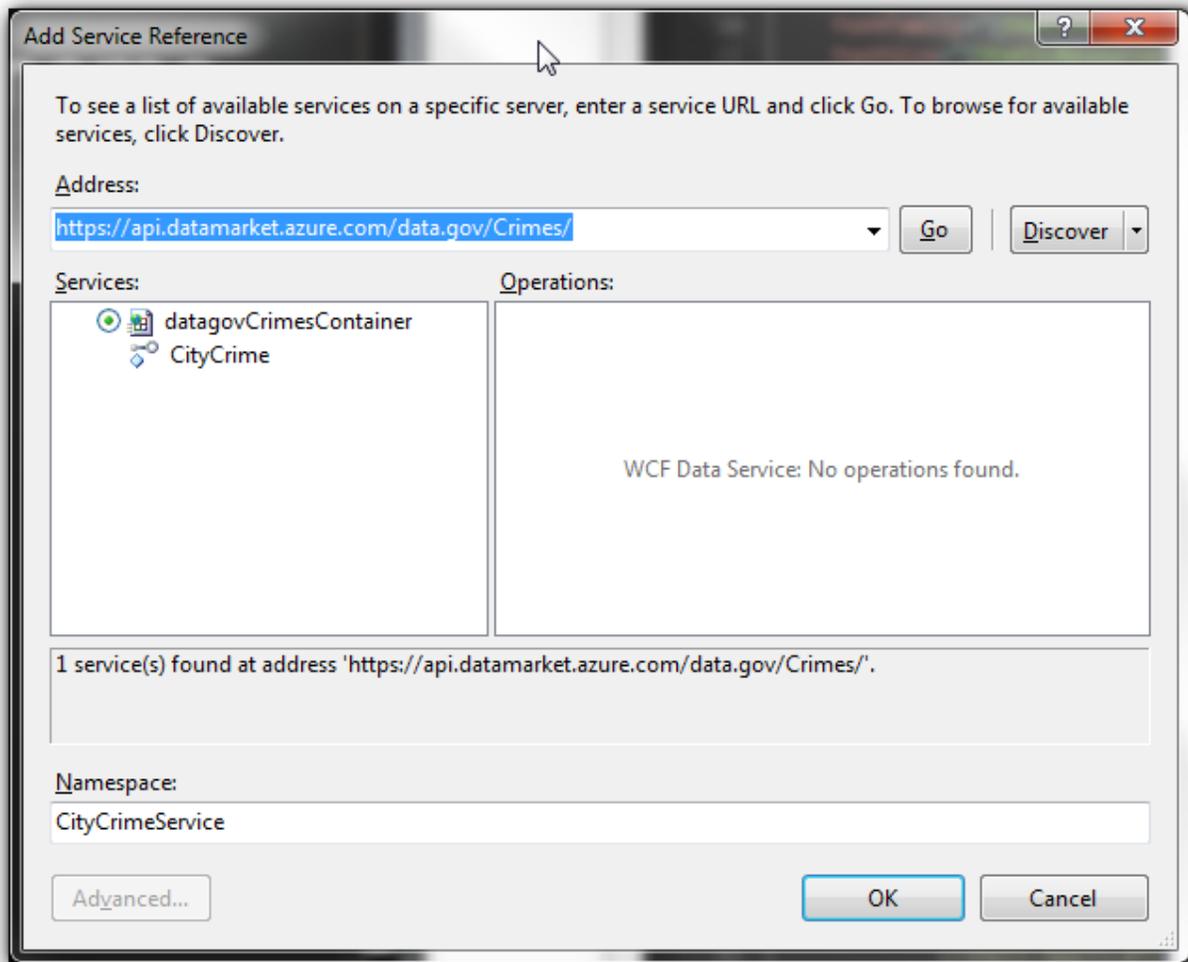
If you haven't done this before, open your Solution Explorer, select your project, then click the Show All Files icon:



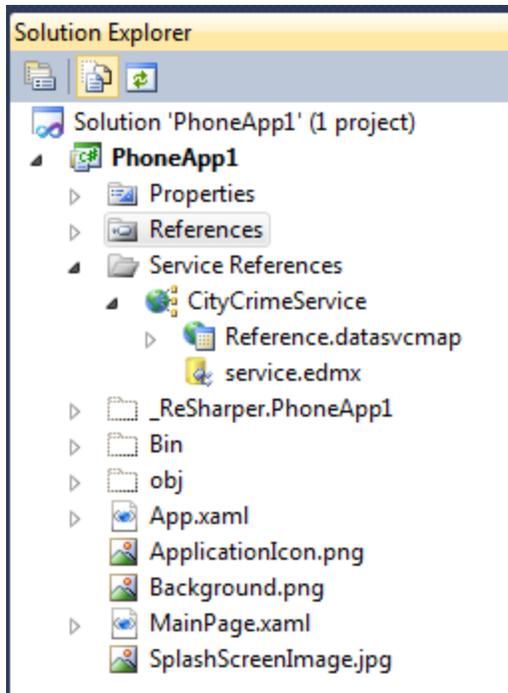
This exposes the References node to us. Right-click on References, and choose Add Service Reference:



Paste the service root URL (<https://api.datamarket.azure.com/data.gov/Crimes/>) into the Address box and click Go. Visual Studio will download the metadata for the service and create the proxy classes we need. Before you click OK, make sure to change the Namespace to something useful:

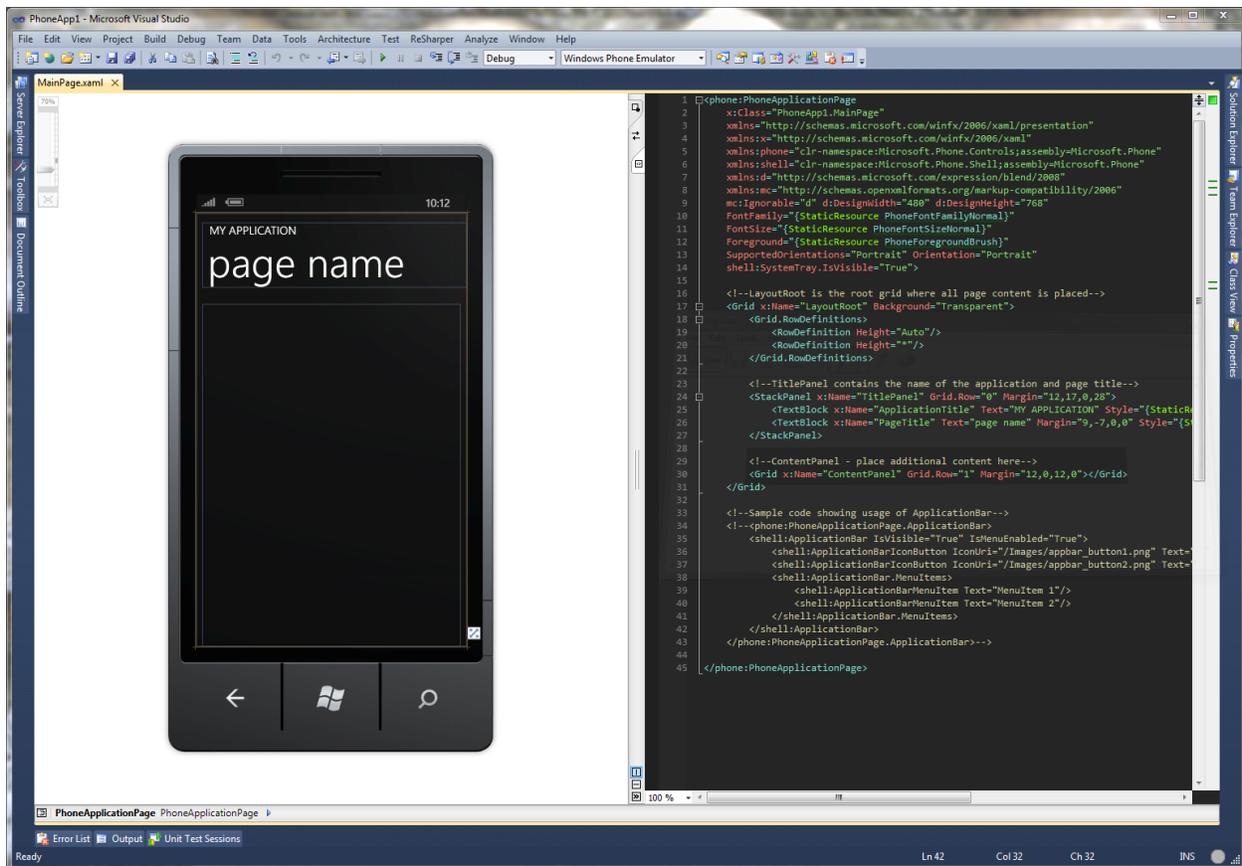


If all goes well, we should see the following underneath the Service References in our Solution Explorer:



### Step 3: Add a Place to Display the Results

We know we're going to retrieve some data, so we need to set a place to show it first. When you created the new project, the `MainPage.xaml` file opened in Visual Studio. On the left is a designer canvas, where you can see what some of the changes we're going to make look like. On the right is the XAML code, which is where we'll be working. Like HTML is the layout for a web page, XAML is the layout for Silverlight. Depending on how your Visual Studio is configured, these might be a top/bottom arrangement.



In the XAML, find the Grid named ContentPanel. This is where we'll put our controls.

```

<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>

```

Later on, when we execute our query, we'll get back a list of items. We'll first add a basic ItemsControl to hold this list. Paste the following code between the <Grid></Grid> tags. The ItemsSource attribute is what connects this ItemsControl to the data retrieved from our query—we'll see the other half of that shortly.

```

<ItemsControl x:Name="MainListBox" ItemsSource="{Binding}" >
</ItemsControl>

```

Your XAML should look like this now:

```

<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <ItemsControl x:Name="MainListBox" ItemsSource="{Binding}" >
    </ItemsControl>
</Grid>
</phone:PhoneApplicationPage>

```

Next, we need to tell our items control how to display the data. To do this, we add an ItemTemplate to our ItemsControl. Add the following code between the <ItemsControl ...></ItemsControl> tags.

```
<ItemsControl.ItemTemplate>
  <DataTemplate>
    <StackPanel >
      <TextBlock Text="{Binding Path=City}"/>
      <TextBlock Text="{Binding Path=Year}"/>
      <TextBlock Text="{Binding Path=ViolentCrime}"/>
    </StackPanel>
  </DataTemplate>
</ItemsControl.ItemTemplate>
```

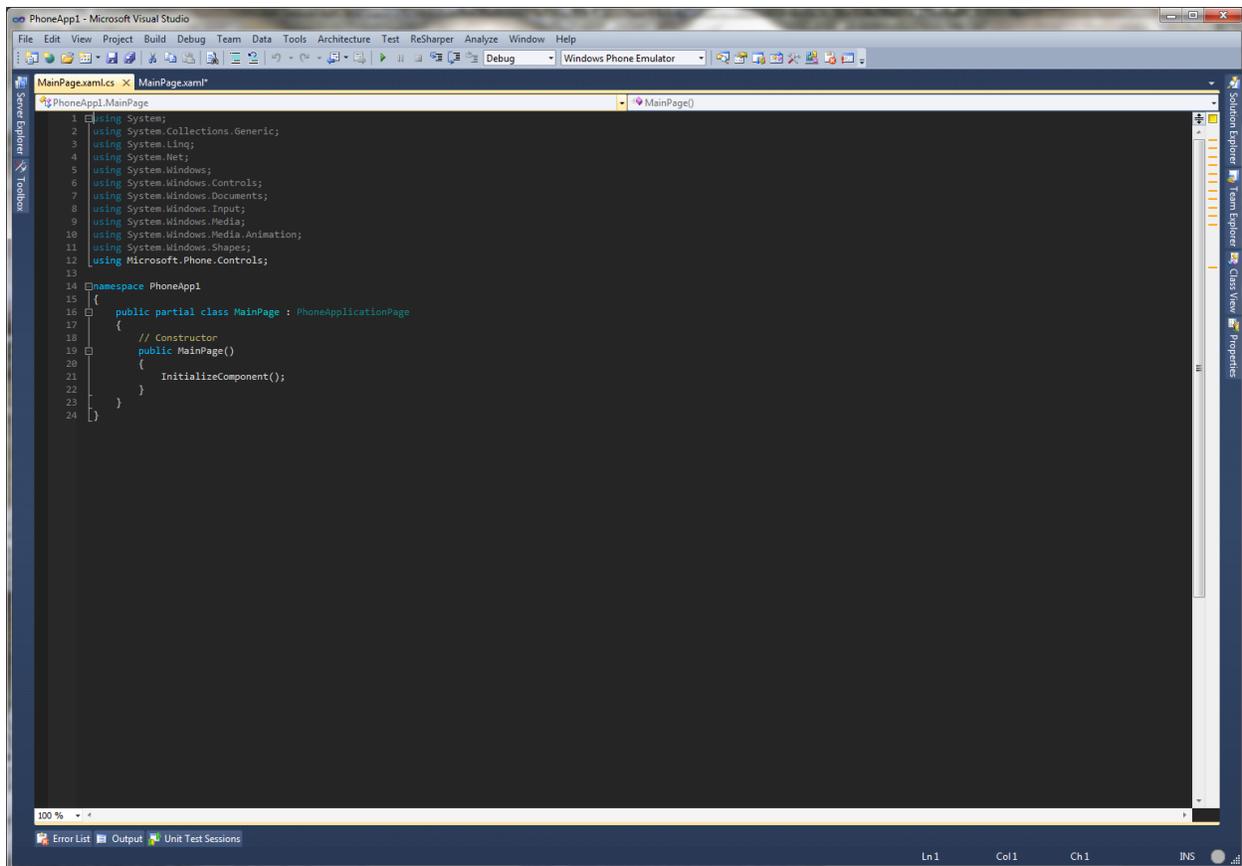
Your XAML should look like this:

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <ItemsControl x:Name="MainListBox" ItemsSource="{Binding}">
    <ItemsControl.ItemTemplate>
      <DataTemplate>
        <StackPanel >
          <TextBlock Text="{Binding Path=City}"/>
          <TextBlock Text="{Binding Path=Year}"/>
          <TextBlock Text="{Binding Path=ViolentCrime}"/>
        </StackPanel>
      </DataTemplate>
    </ItemsControl.ItemTemplate>
  </ItemsControl>
</Grid>
```

The above code will display the city name, year and number of violent crimes in that city in that year. We'll add the other crimes in a little bit. This is a good start.

#### Step 4: Get the Data

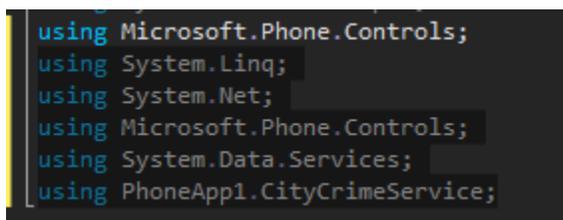
Now it's time to call our data service, which means we have to edit the code in a different file. In the Solution Explorer, open MainPage.xaml.cs. You should see something similar to below:



At the top of this file, we need to add some references to libraries we'll use on this page. Add the following statements to the end of the "using block" at the top of the page.

```
using System.Linq;
using System.Net;
using Microsoft.Phone.Controls;
using System.Data.Services.Client;
using PhoneApp1.CityCrimeService;
```

It should look like this:



Add the following code to the top of the MainPage partial class:

```
private datagovCrimesContainer _context;
```

```

private readonly Uri _crimesUri = new
Uri("https://api.datamarket.azure.com/data.gov/Crimes/");
private DataServiceCollection<CityCrime> _crimes;

private const string _userId = "me@myself.com";
private const string _key = "Use your real API key here";

```

Your code should look like this:

```

public partial class MainPage : PhoneApplicationPage
{
    private datagovCrimesContainer _context;
    private readonly Uri _crimesUri = new Uri("https://api.datamarket.azure.com/data.gov/Crimes/");
    private DataServiceCollection<CityCrime> _crimes;

    private const string _userId = "me@myself.com";
    private const string _key = "FtsDefZvJNf0MXiTgFgxmiqGqYKkK6kkNrC3xiPCEdU=";

    // Constructor
    public MainPage()
    {
        InitializeComponent();
    }
}

```

Inside our constructor is where we'll query the data set. This query will execute when the page is loaded. Copy the following code into the Constructor method (feel free to use your city and state in the query):

```

    _context = new datagovCrimesContainer(_crimesUri);
    _crimes = new DataServiceCollection<CityCrime>(_context);
    var _query = from c in _context.CityCrime where c.State == "Pennsylvania" && c.City ==
    "Pittsburgh" orderby c.Year select c;

    _context.Credentials = new NetworkCredential(_userId, _key);

    // Register for the LoadCompleted event.
    _crimes.LoadCompleted += new
    EventHandler<LoadCompletedEventArgs>(crimes_LoadCompleted);

    // Load the customers feed by executing the LINQ query.
    _crimes.LoadAsync(_query);

```

Your code should look like this:

```

// Constructor
public MainPage()
{
    InitializeComponent();

    _context = new datagovCrimesContainer(_crimesUri);
    _crimes = new DataServiceCollection<CityCrime>(_context);
    var _query = from c in _context.CityCrime where c.State == "Pennsylvania" && c.City == "Pittsburgh" orderby c.Year select c;

    _context.Credentials = new NetworkCredential(_userId, _key);

    // Register for the LoadCompleted event.
    _crimes.LoadCompleted += new EventHandler<LoadCompletedEventArgs>(crimes_LoadCompleted);

    // Load the customers feed by executing the LINQ query.
    _crimes.LoadAsync(_query);
}

```

After our query is executed, our application needs to know what to do with the data. In Silverlight, we pass off the next steps to another function, called a “callback method”, where we process the data. The name is highlighted in red above. Add the following code to the MainPage class, but outside the MainPage constructor:

```

void crimes_LoadCompleted(object sender, LoadCompletedEventArgs e)
{
    this.LayoutRoot.DataContext = _crimes;
}

```

Your code should now look something like this:

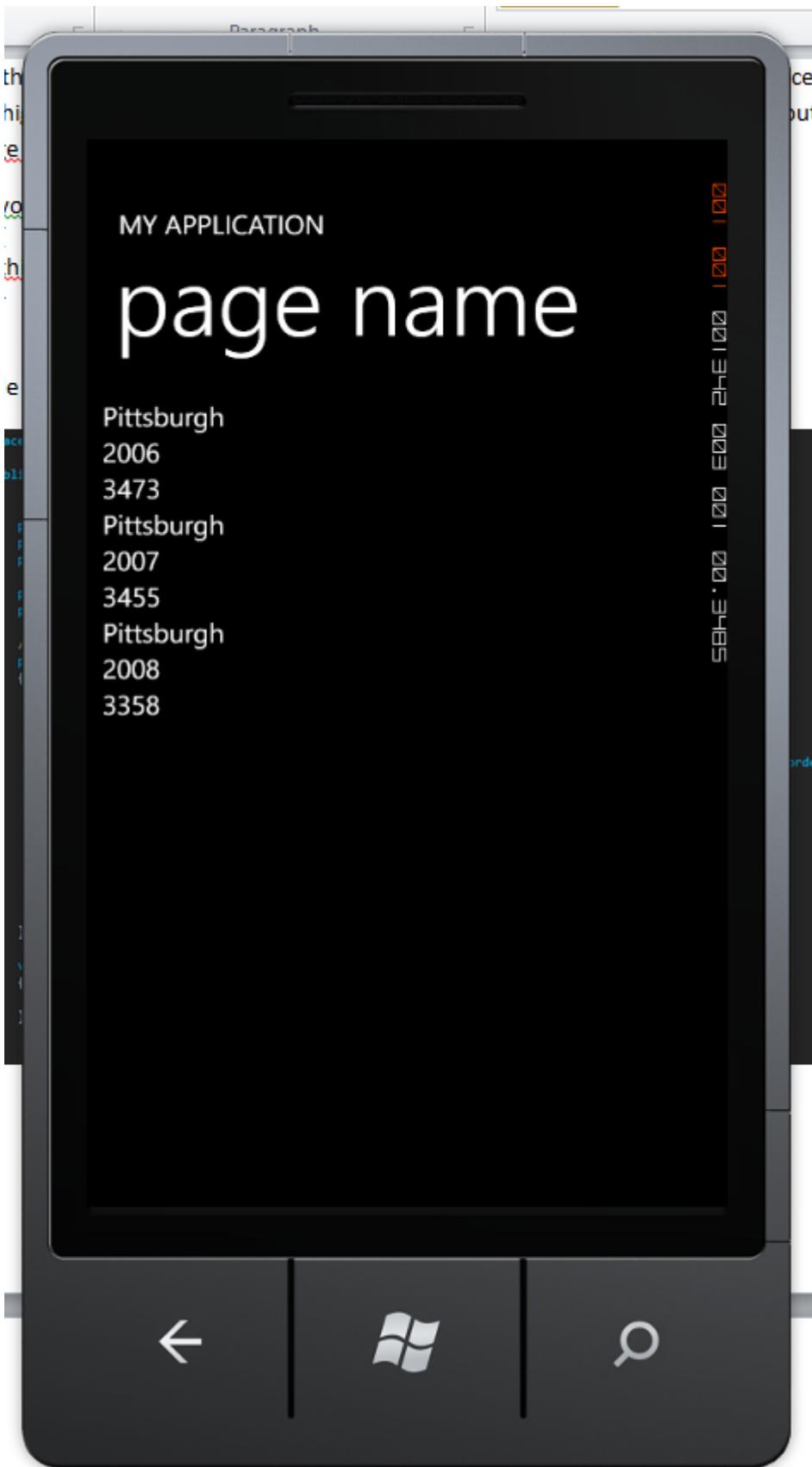
```

19 namespace PhoneApp1
20 {
21     public partial class MainPage : PhoneApplicationPage
22     {
23
24         private datagovCrimesContainer _context;
25         private readonly Uri _crimesUri = new Uri("https://api.datamarket.azure.com/data.gov/Crimes/");
26         private DataServiceCollection<CityCrime> _crimes;
27
28         private const string _userId = "me@myself.com";
29         private const string _key = "FtsDefZvJNf0MXiTgFgxmIqGqYKkK6kkNrC3xiPCeU=";
30
31         // Constructor
32         public MainPage()
33         {
34             InitializeComponent();
35
36             _context = new datagovCrimesContainer(_crimesUri);
37             _crimes = new DataServiceCollection<CityCrime>(_context);
38             var _query = from c in _context.CityCrime where c.State == "Pennsylvania" && c.City == "Pittsburgh" orderby c.Year select c;
39
40             _context.Credentials = new NetworkCredential(_userId, _key);
41
42             // Register for the LoadCompleted event.
43             _crimes.LoadCompleted += new EventHandler<LoadCompletedEventArgs>(crimes_LoadCompleted);
44
45             // Load the customers feed by executing the LINQ query.
46             _crimes.LoadAsync(_query);
47
48         }
49
50         void crimes_LoadCompleted(object sender, LoadCompletedEventArgs e)
51         {
52             this.LayoutRoot.DataContext = _crimes;
53         }
54     }
55 }

```

All we're doing in our callback is telling the application, "anything inside the LayoutRoot control should use our query results as its data source". This is the other half of the Binding directive we used above in the XAML.

If we run our application at this time, we should see the following results:



If you see data, excellent! Now it's time to do a little more. Let's add all the crimes to our list, and the name of the crime. Replace your ItemTemplate with the code below:

```
<ItemsControl.ItemTemplate>
  <DataTemplate>
    <StackPanel>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Year: " />
          <Run Text="{Binding Path=Year}"/>
        </TextBlock.Inlines>
      </TextBlock>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Population: " />
          <Run Text="{Binding Path=Population}"/>
        </TextBlock.Inlines>
      </TextBlock>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Violent Crimes: " />
          <Run Text="{Binding Path=ViolentCrime}"/>
        </TextBlock.Inlines>
      </TextBlock>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Murders: " />
          <Run Text="{Binding Path=MurderAndNonEgligentManslaughter}"/>
        </TextBlock.Inlines>
      </TextBlock>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Forcible Rapes: " />
          <Run Text="{Binding Path=ForcibleRape}"/>
        </TextBlock.Inlines>
      </TextBlock>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Robberies: " />
          <Run Text="{Binding Path=Robbery}"/>
        </TextBlock.Inlines>
      </TextBlock>
      <TextBlock>
        <TextBlock.Inlines>
          <Run Text="Aggravated Assaults: " />
          <Run Text="{Binding Path=AggravatedAssault}"/>
        </TextBlock.Inlines>
      </TextBlock>
    </StackPanel>
  </DataTemplate>
</ItemsControl.ItemTemplate>
```

```

<TextBlock>
<TextBlock.Inlines>
<Run Text="Property Crimes: " />
<Run Text="{Binding Path=PropertyCrime}"/>
</TextBlock.Inlines>
</TextBlock>
<TextBlock>
<TextBlock.Inlines>
<Run Text="Burglaries: " />
<Run Text="{Binding Path=Burglary}"/>
</TextBlock.Inlines>
</TextBlock>
<TextBlock>
<TextBlock.Inlines>
<Run Text="Larcenies: " />
<Run Text="{Binding Path=LarcenyTheft}"/>
</TextBlock.Inlines>
</TextBlock>
<TextBlock>
<TextBlock.Inlines>
<Run Text="Motor Vehicle Thefts: " />
<Run Text="{Binding Path=MotorVehicleTheft}"/>
</TextBlock.Inlines>
</TextBlock>
<TextBlock>
<TextBlock.Inlines>
<Run Text="Arson: " />
<Run Text="{Binding Path=Arson}"/>
</TextBlock.Inlines>
</TextBlock>
</StackPanel>
</DataTemplate>
</ItemsControl.ItemTemplate>

```

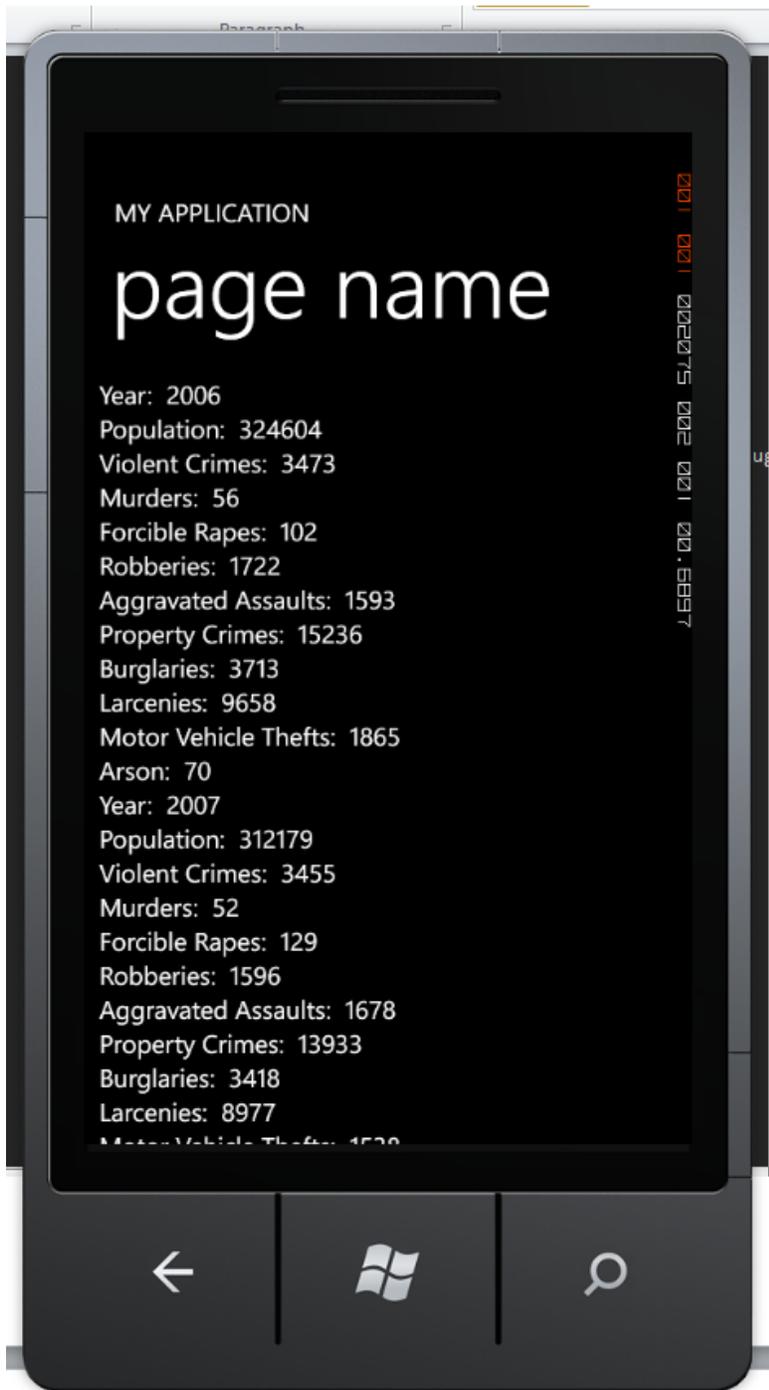
In order to have both the name of the crime, and the number of crimes from the data set, we have to use two separate Runs inside of our TextBlock. It adds a little extra code, but the end result looks nice. Your code should look something like below:

```

<ItemsControl x:Name="MainListBox" ItemsSource="{Binding}">
  <ItemsControl.ItemTemplate>
    <DataTemplate>
      <StackPanel>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Year: " />
            <Run Text="{Binding Path=Year}"/>
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Population: " />
            <Run Text="{Binding Path=Population}"/>
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Violent Crimes: " />
            <Run Text="{Binding Path=ViolentCrime}"/>
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Murders: " />
            <Run Text="{Binding Path=MurderAndNonEgligentManslaug" />
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Forcible Rapes: " />
            <Run Text="{Binding Path=ForcibleRape}"/>
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Robberies: " />
            <Run Text="{Binding Path=Robbery}"/>
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Aggravated Assaults: " />
            <Run Text="{Binding Path=AggravatedAssault}"/>
          </TextBlock.Inlines>
        </TextBlock>
        <TextBlock>
          <TextBlock.Inlines>
            <Run Text="Property Crimes: " />
            <Run Text="{Binding Path=PropertyCrime}"/>
          </TextBlock.Inlines>
        </TextBlock>
      </TextBlock>
    </DataTemplate>
  </ItemsControl.ItemTemplate>
</ItemsControl>

```

If we run our application, we should see the following output:



If you notice, the data scrolls off the bottom, but our screen won't scroll. We need to add a `ScrollView` around our `ItemsControl` to enable scrolling.

At the top of the `ItemsControl`, add a `<ScrollView>` tag, and at the bottom, add a `</ScrollView>` tag. Your code should look like below:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <ScrollView>
    <ItemsControl x:Name="MainListBox" ItemsSource="{Binding}">
      <ItemsControl.ItemTemplate>
```

And

```
    </ItemsControl.ItemTemplate>
  </ItemsControl>
</ScrollView>
</Grid>
</Grid>
```

If we run our app now, we have a scrolling list of data, ready to be made more attractive!

## References/Resources

### Blogs to Read

Microsoft Public Sector DPE - <http://blogs.msdn.com/b/publicsector/>

Rich Dudley - <http://c1.ms/richd>

Greg Lutz - [http://our.componentone.com/author/c1\\_greg/](http://our.componentone.com/author/c1_greg/)

### Windows Phone 7 Development

AppHub: <http://create.msdn.com>

Get [Phone SDK](http://aka.ms/getphone) (<http://aka.ms/getphone>)

Microsoft Forums - <http://forums.create.msdn.com/forums/>

WP 7.5 Training Kit - <http://www.microsoft.com/download/en/details.aspx?id=28564>

MSDN Documentation - [http://msdn.microsoft.com/en-us/library/ff402535\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/ff402535(v=vs.92).aspx)

Windows Phone Camps - <http://channel9.msdn.com/Series/Windows-Phone-Camps-Online>

Free "Programming Windows Phone 7" ebook - <http://www.charlespetzold.com/phone/>

InfoKit - <http://bit.ly/t8lpAM>

Windows Phone 7 Development for Absolute Beginners - <http://channel9.msdn.com/Series/Windows-Phone-7-Development-for-Absolute-Beginners>

31 Days of Windows Phone 7 - <http://www.jeffblankenburg.com/2010/09/30/31-days-of-windows-phone-7/>

31 Days of Mango - <http://www.jeffblankenburg.com/2010/09/30/31-days-of-windows-phone-7/>

20 hours of free Windows Phone 7 training - <http://www.goodcat-trainings.net/>

## Linq

101 Linq Samples, C# - <http://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b>

101 Linq Samples, VB.NET - <http://msdn.microsoft.com/en-us/vstudio/bb688088>

## Additional Code Libraries

Odata libraries - <http://odata.codeplex.com/releases/view/54698>

## Azure

Get [Azure SDK](http://aka.ms/getazure) (or <http://aka.ms/getazure>)

Get [Azure Trial](http://aka.ms/azuretrial) (or <http://aka.ms/azuretrial>)

Data Market - <http://datamarket.azure.com/>

Self-publish wizard - <http://blogs.msdn.com/b/windowsazure/archive/2011/12/06/new-self-service-publishing-wizard-eases-distribution-of-new-datasets-on-the-windows-azure-marketplace.aspx>