Contents

The **Windows Store Developer's Tour** provides an introduction to developing a Windows Store app with DevForce 2012.

What you'll learn:

- How to write n-tier LINQ queries in DevForce
- How to handle asynchronous queries and saves
- Simple navigation techniques in Windows Store apps
- Abstracting data management into a DataService

This version of the tour uses a **Database First** model. See **here** for the Code First version.

- **Platform:** Windows Store
- **Language:** C#
- **Download:** Windows Store Dev Tour
- **Prerequisites:** Windows 8 RTM or above

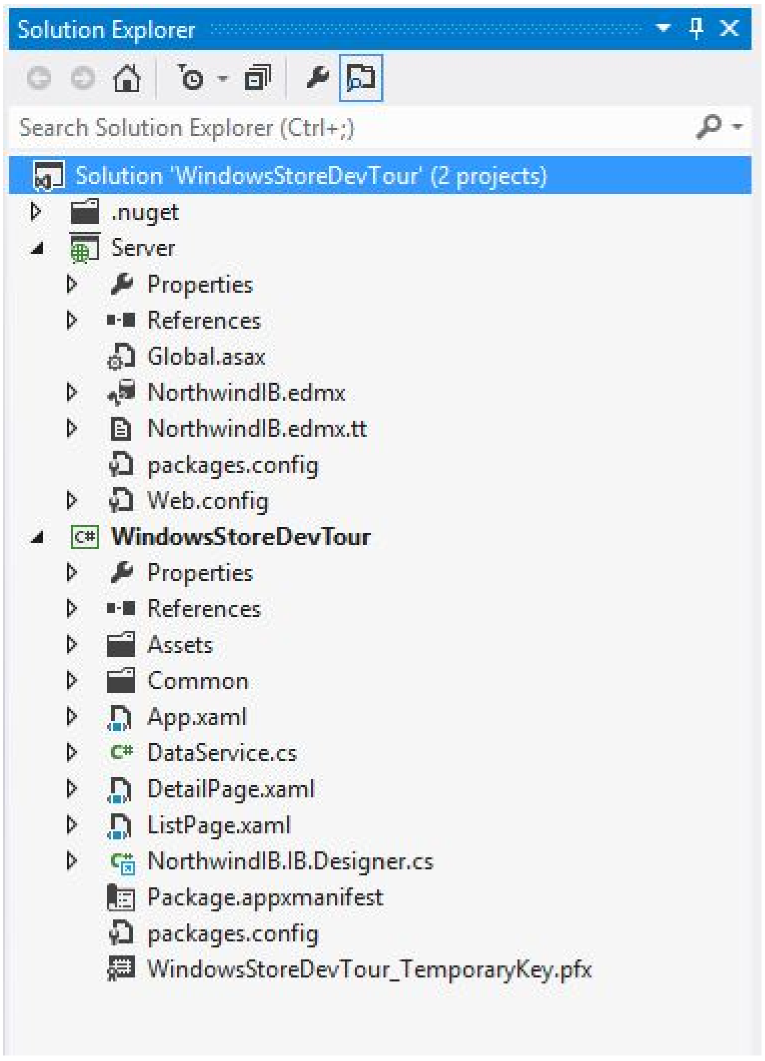
Getting started

The Windows Store Developer's Tour demonstrates a simple two page master/detail application. The first page lists all customers in the NorthwindIB sample database and provides search capability. Tap a customer and it takes you to the detail page where you can edit the customer and save. Tapping the back button takes you back to the list.

You must enable NuGet package restore within Visual Studio in order to restore the required **DevForce NuGet packages** and other dependencies. The first time you build the application, the dependent NuGet packages are installed.

The Developer's tour includes a SQL CE version of the *NorthwindIB* sample database.

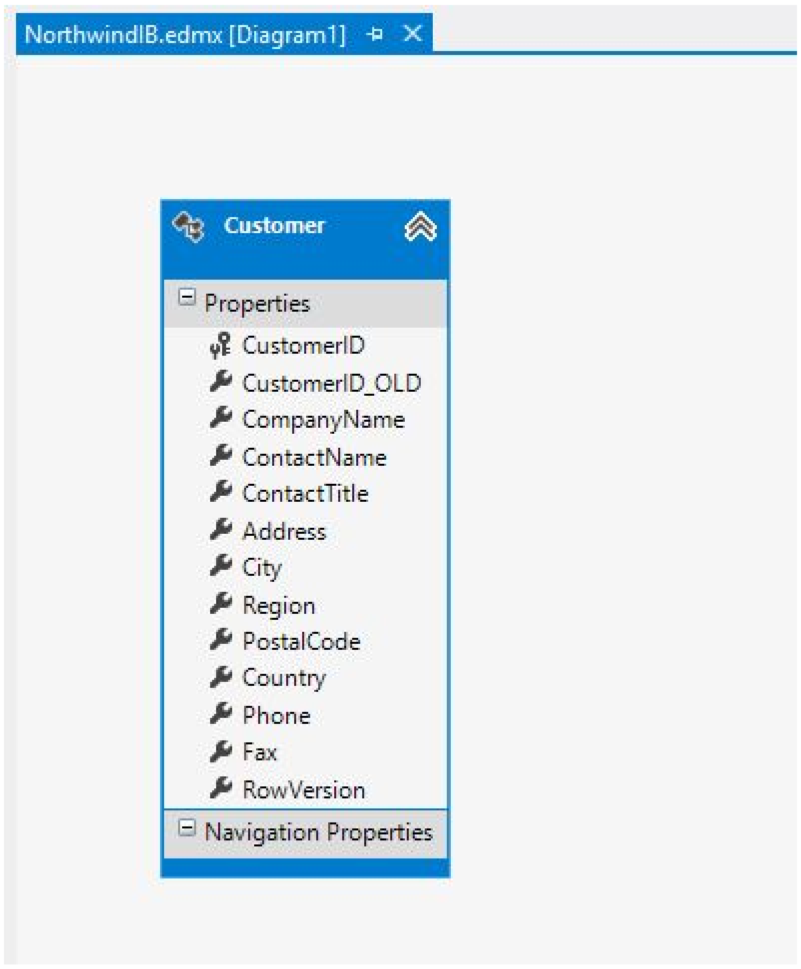
Let's first take a look at the solution structure:



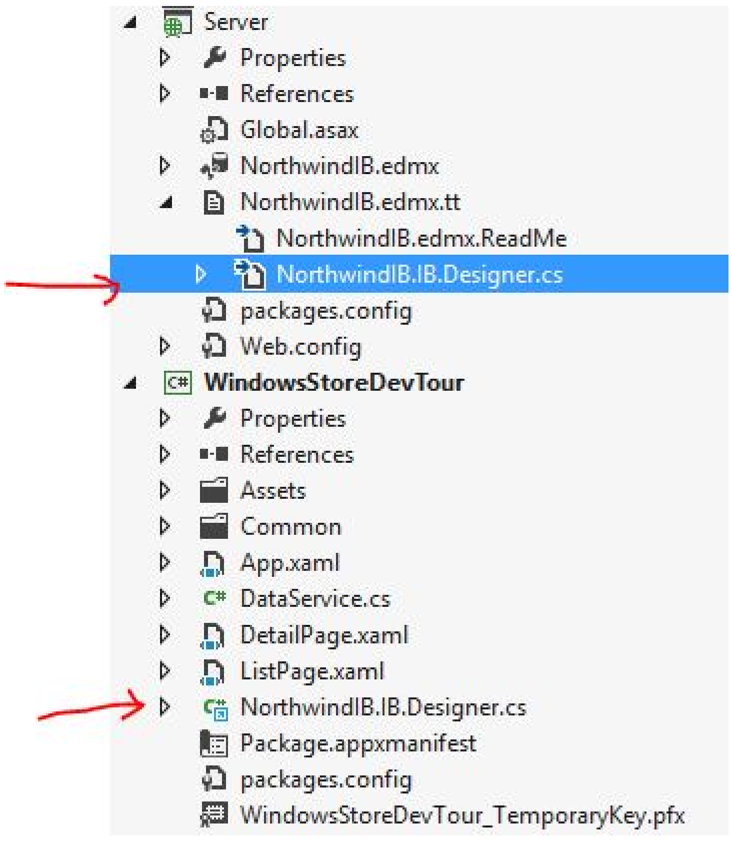
```
Solution 'WindowsStoreDevTour' (2 projects)
  .nuget
  Server
    Properties
    References
    Global.asax
    NorthwindIB.edmx
    NorthwindIB.edmx.tt
    packages.config
    Web.config
  WindowsStoreDevTour
    Properties
    References
    Assets
    Common
    App.xaml
    DataService.cs
    DetailPage.xaml
    ListPage.xaml
    NorthwindIB.IB.Designer.cs
    Package.appxmanifest
    packages.config
    WindowsStoreDevTour_TemporaryKey.pfx
```

A DevForce Windows Store application is an n-tier application, and uses an EntityServer to perform all backend data access. We see that reflected in the solution structure: *WindowsStoreDevTour* is the client application project, and *Server* is the web application project hosting the *EntityServer*.

Next, let's look at the model. The Entity Data Model was added to the *Server* project, and contains only a single entity type, *Customer*:



Finally, note that while the entity model is located in the *Server* project, the generated code for the entities, in the *NorthwindIB.IB.Designer.cs* file, is linked in the Windows Store project so that these entities (and any business logic you add to the entities) can execute on the client:



```
◢ Server
    ▷ Properties
    ▷ References
      Global.asax
    ▷ NorthwindIB.edmx
    ◢ NorthwindIB.edmx.tt
          NorthwindIB.edmx.ReadMe
        ▷ NorthwindIB.IB.Designer.cs
      packages.config
    ▷ Web.config
◢ WindowsStoreDevTour
    ▷ Properties
    ▷ References
    ▷ Assets
    ▷ Common
    ▷ App.xaml
    ▷ DataService.cs
    ▷ DetailPage.xaml
    ▷ ListPage.xaml
    ▷ NorthwindIB.IB.Designer.cs
      Package.appxmanifest
      packages.config
      WindowsStoreDevTour_TemporaryKey.pfx
```

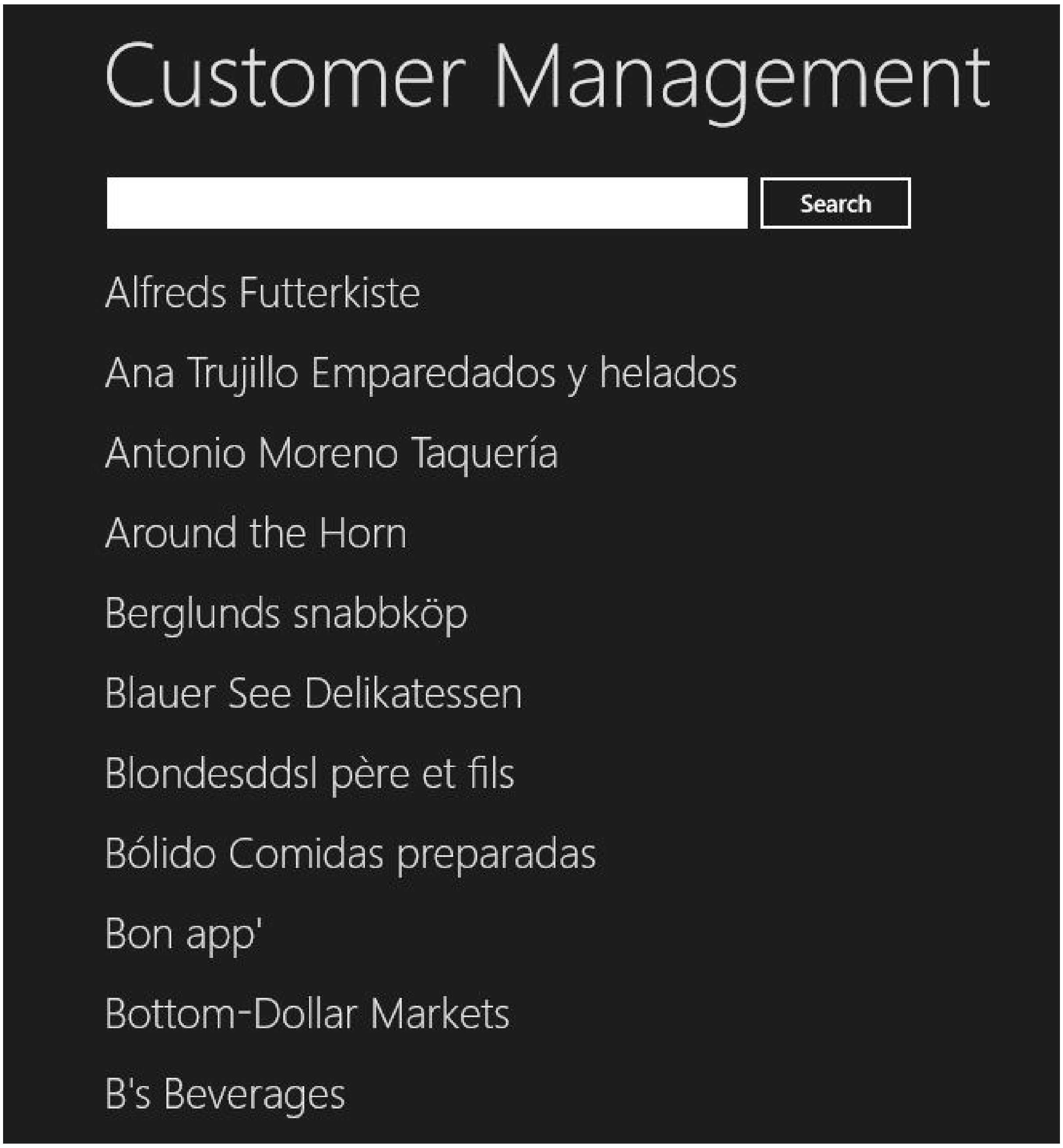
You can easily extend these entities by declaring a partial class for any of them and linking that file to the client too. These techniques are common to DevForce client development.

Overview

1. In order to query data from the server, you need to specify the URL of the *EntityServer*. You can do this either with an embedded app.config file, or programmatically, as show in the application constructor:

```
public App()
{
  this.InitializeComponent();
  this.Suspending += OnSuspending;
   IdeaBladeConfig.Instance.ObjectServer.RemoteBaseUrl = "http://localhost";
   IdeaBladeConfig.Instance.ObjectServer.ServerPort = 57209;
   IdeaBladeConfig.Instance.ObjectServer.ServiceName = "EntityService.svc";
}
```

2. ListPage.xaml is the master/search page. It displays all customers and provides search capability by customer name.


Customer Management

Alfreds Futterkiste

Ana Trujillo Emparedados y helados

Antonio Moreno Taquería

Around the Horn

Berglunds snabbköp

Blauer See Delikatessen

Blondesddsl père et fils

Bólido Comidas preparadas

Bon app'

Bottom-Dollar Markets

B's Beverages

Looking at the code, notice the Start method, called from the *OnNavigatedTo* handler:

```
public async void Start()
{
  try
  {
      IsBusy = true;
      var customers = await DataService.Instance.GetAllCustomersAsync();
      Customers = new ObservableCollection<Customer>(customers);
  }
  finally
  {
      IsBusy = false;
  }
}
```

A few things to note here: data retrieval is asynchronous using the *async/await* keywords, and data management activities are performed by a *DataService* class.

The *IsBusy* flag is used, via data binding, to display a busy indicator:

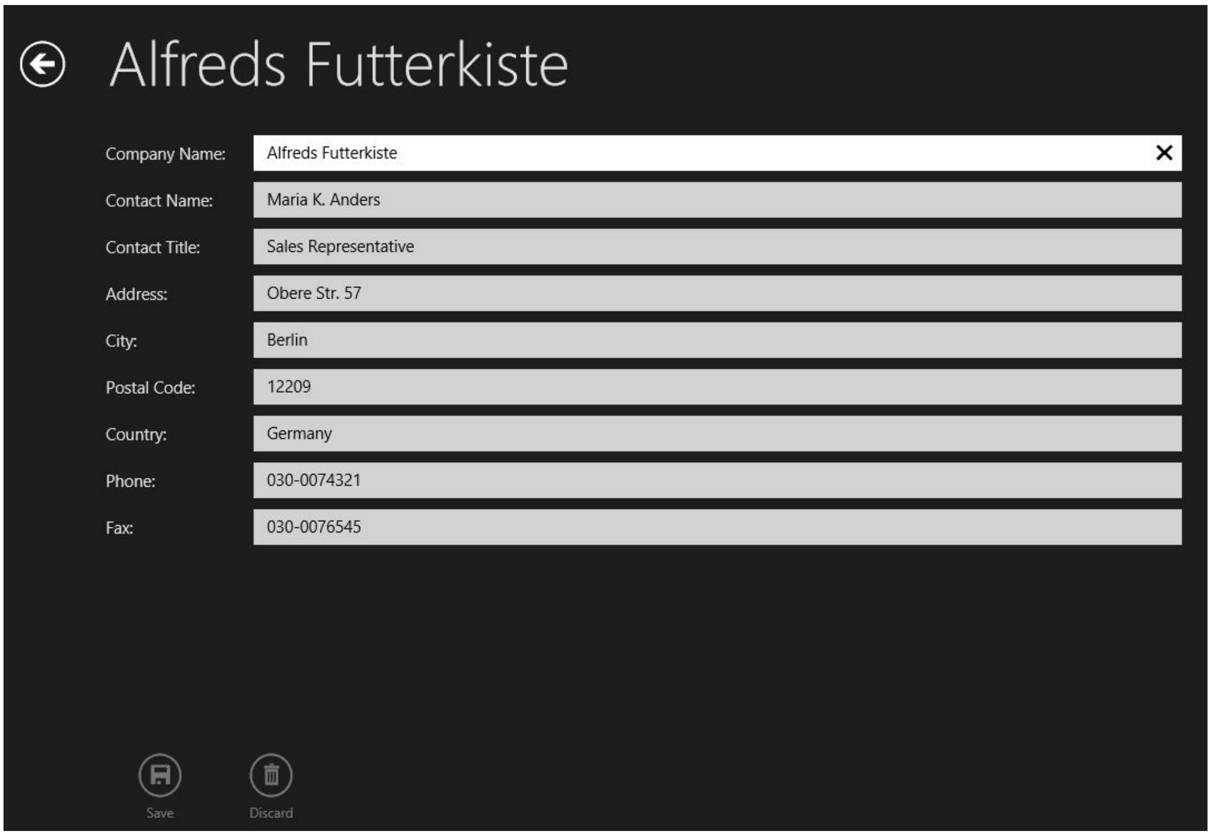
```
ProgressRing Grid.Row="2" IsActive="{Binding IsBusy}"/>
```

Let's also look at the *search* button event handler:

```
private async void Search(object sender, RoutedEventArgs e)
{
   try
    {
        IsBusy = true;
       if (string.IsNullOrEmpty(SearchText))
        {
            Start();
          return;
        }
        var customers = await DataService.Instance.FindCustomersAsync(SearchText);
        Customers = new ObservableCollection<Customer>(customers);
    }
   finally
    {
        IsBusy = false;
    }
}
```

Again, we see asynchronous data retrieval using the *DataService* which encapsulates the DevForce LINQ queries.

3. When a customer is selected, a detail page is displayed. *DetailPage.xaml* shows customer details, and allows editing with save and undo capabilities.



Alfreds Futterkiste

Company Name:	Alfreds Futterkiste
Contact Name:	Maria K. Anders
Contact Title:	Sales Representative
Address:	Obere Str. 57
City:	Berlin
Postal Code:	12209
Country:	Germany
Phone:	030-0074321
Fax:	030-0076545

Navigation to the detail page uses *Frame.Navigate*:

```
private void NavigateToDetailPage()
{
 Frame.Navigate(typeof (DetailPage), SelectedCustomer.CustomerID);
}
```

Looking at the code for DetailPage.xaml.cs, we see the *Start* method is called from the *OnNavigatedTo* handler, which receives the ID of the customer to be displayed in the *NavigationEventArgs*.

```
public async void Start(Guid customerId)
{
   Customer = await DataService.Instance.GetCustomerAsync(customerId);
}
```

And here's the *Save* button handler:

```
private async void Save(object sender, RoutedEventArgs e)
```

```
{
 await DataService.Instance.SaveAsync();
}
```

4. The *DataService.cs* contains a singleton data service used throughout the application. Both the *ListPage* and *DetailPage* work directly with the *DataService*, which encapsulates the DevForce EntityManager and performs all queries and saves.

Here's a sample method, this one to retrieve all customers:

```
public Task<IEnumerable<Customer>> GetAllCustomersAsync()
{
    return _entityManager.Customers.OrderBy(x => x.CompanyName).ExecuteAsync();
}
```

The *Task* returned is awaited upon by the caller, the *ListPage Start* method we saw above.

Additional resources

- Windows Store Dev Center