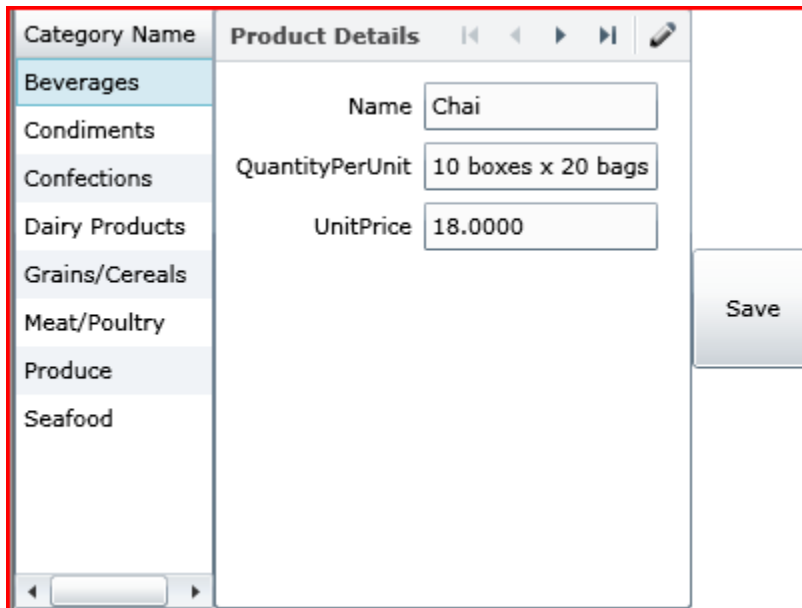


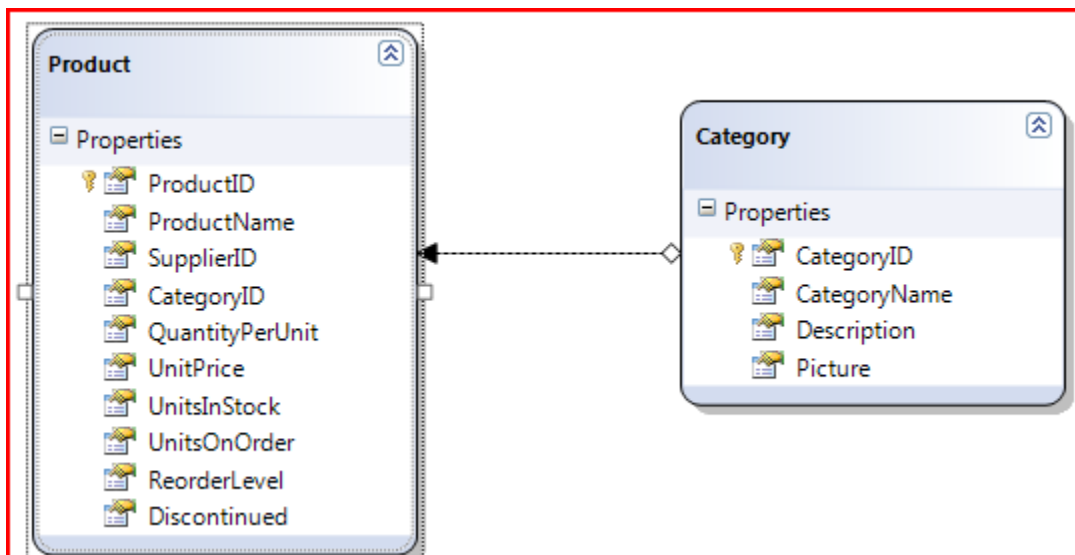
Chapter 3 Working with Object Hierarchy

The next example we will see how to use an object hierarchy to produce a UI that looks like:

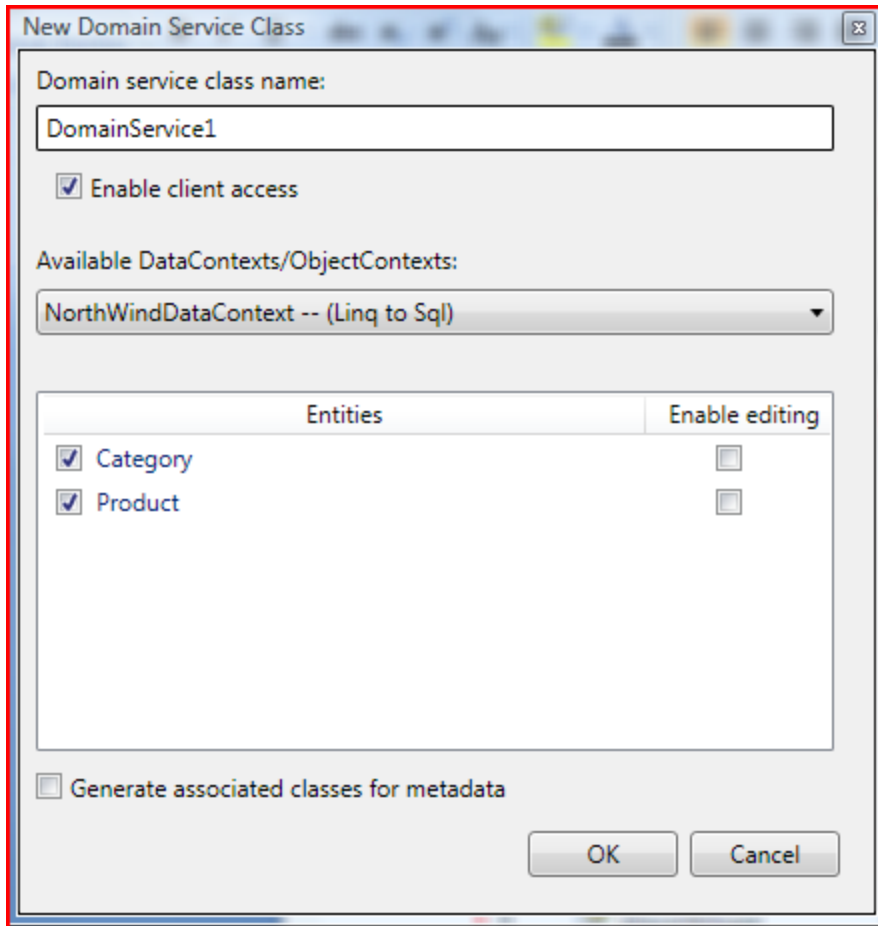


The screenshot shows a UI form titled "Product Details" with a sidebar on the left. The sidebar lists categories: Beverages, Condiments, Confections, Dairy Products, Grains/Cereals, Meat/Poultry, Produce, and Seafood. The main area contains three input fields: "Name" with the value "Chai", "QuantityPerUnit" with the value "10 boxes x 20 bags", and "UnitPrice" with the value "18.0000". A "Save" button is located on the right side of the form.

The XAML is included below and I want to show how the detail DataForm is populated using an object hierarchy and to show that the Ria change management works great even in this case. To start, we use the Northwind database and Category and products table where there is a foreign key relationship between the parent category and child product tables.



We then add the DomainContext to our project and making sure to select Category and Product :



When this is done we need to modify the GetCategories in the LinqToSqlDomainService class so we will load the products with the categories using the Linq **LoadWith**:

```
public IQueryable<Category> GetCategories()  
{  
    var LoadOptions = new DataLoadOptions();  
    LoadOptions.LoadWith<Category>(p => p.Products);  
    this.Context.LoadOptions = LoadOptions;  
    return this.Context.Categories;  
}
```

Now my XAML for the simple UX looks like:

```
<StackPanel Orientation="Horizontal">

<data:DataGrid x:Name="MyGrid" Width="100"
AutoGenerateColumns="False"
SelectionChanged="MyGrid_SelectionChanged">
    <data:DataGrid.Columns>
        <data:DataGridTextColumn Header="Category Name"
            Binding="{Binding CategoryName}"></data:DataGridTextColumn>
    </data:DataGrid.Columns>
</data:DataGrid>

<dataControls:DataForm HorizontalAlignment="Stretch"
x:Name="Details" AutoGenerateFields="False"
CanUserAddItems="True" Header="Product Details" >
    <dataControls:DataForm.Fields>
        <dataControls:DataFormTextField FieldLabelContent="Name"
Binding="{Binding ProductName }">
            </dataControls:DataFormTextField>
        <dataControls:DataFormTextField
            FieldLabelContent="QuantityPerUnit" Binding="{Binding
QuantityPerUnit}">
            </dataControls:DataFormTextField>
        <dataControls:DataFormTextField
            FieldLabelContent="UnitPrice" Binding="{Binding UnitPrice,
Mode=TwoWay}">
            </dataControls:DataFormTextField>
    </dataControls:DataForm.Fields>
</dataControls:DataForm>

<Button Width="60" Height="60" Content="Save"
Click="Button_Click"></Button>

</StackPanel>
```

DataGrid for Category name, single column bound to CategoryName

DataForm for the child products. DataFormTextField are bound to properties of child List<Products>

As the mainPage is loaded the List of categories is bound to the Datagrid and the child List of products to the DataForm. The Dataform is a really rich control that will lets us scroll thru the Product list using the control below:



As the selection changes in the datagrid we rebind the associated products list to the DataForm:

```
private void MyGrid_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    Category cat = e.AddedItems[0] as Category;
    Details.ItemsSource = cat.Products;
}
```

Now the really cool part is how good the change management is as you can change any of the product properties like unit price and when the Save changes button is clicked the changed product is sent back to the Middle tier and persisted to the database.