## External Data in SharePoint 2010

**Lab Time:** 60 minutes

**Lab Folder**: C:\Student\Labs\BCS

**Lab Overview:** Business Connectivity Services (BCS) allows you to create SharePoint applications that can utilize external data. BCS relies on the creation of External Content Types to support the applications, which can then be extended to Office clients. In this exercise, you will use BCS to create a SharePoint application.

**Lab Setup Requirements**

* Before you begin this lab, you must run the batch file named **SetupLab.bat**. This batch file runs a PowerShell script which creates a new SharePoint site collection at the location **http://intranet.wingtip.com/sites/BCS**.

### Exercise 1: Creating External Content Types in SPD

External Content Types are the foundation of BCS applications. External Content Types represent the business objects in the external data that you would like to use in your application. In this exercise, you will use the SharePoint Designer (SPD) to create entities based on the **AdventureWorks** database.

1. Start **SharePoint Designer 2010** (SPD) and click **Open Site** and enter **http://intranet.wingtip.com/sites/BCS**. You may need to login when prompted. If so, use the **administrator** account.
2. After the site opens, click **External Content Types** in the left-hand pane. Give SPD a moment to build the list of existing entities, which should be empty at this point.
3. When the entity report completes, click the **External Content Type** button in the **New** group on the ribbon.
4. Use the following to update the **External Content Type**:

**Name:** Contact

**Display Name:** Contact

**Namespace:** AdventureWorks

**Office Item Type:** Contact

1. On the ribbon, click the **Operations Design View** button in the **Views** group.
2. On the **Contact** tab, click the **Add Connection** button.
3. In the **Select Your Data Source Type** dialog, select **SQL Server** and click the **OK** button.
4. In the **SQL Server Connection** dialog, use the following and click **OK**:

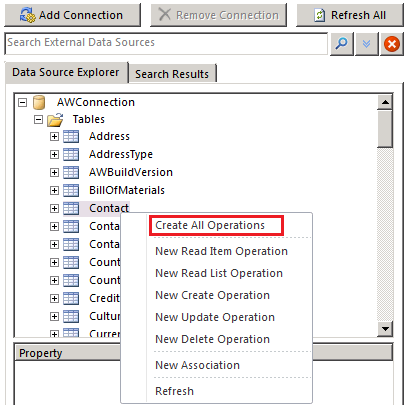
**SQL Database Server**: WingtipServer

**Database Name:** AdventureWorks

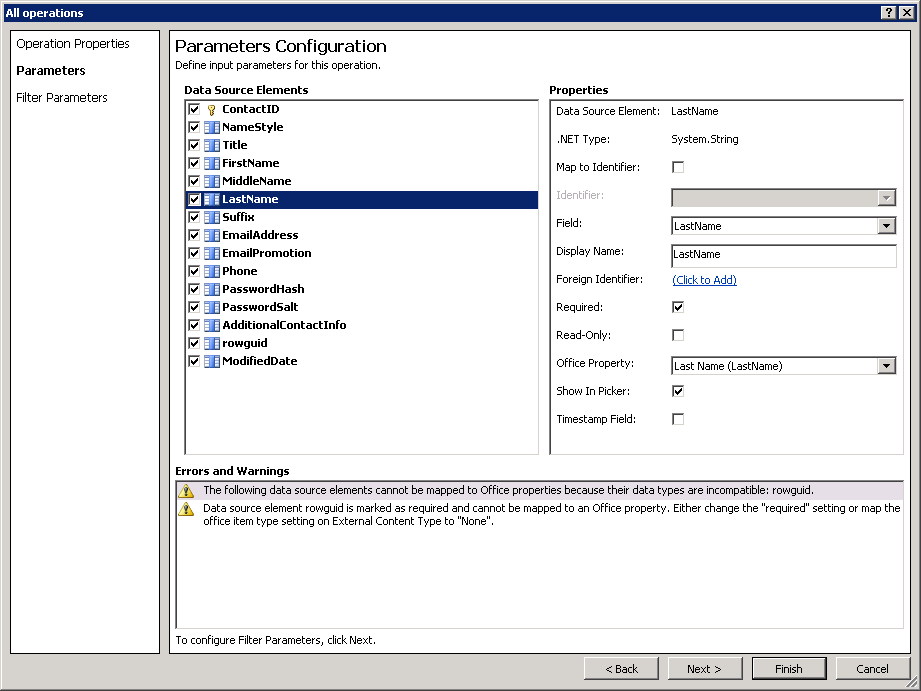
**Name (optional):** AWConnection

Choose **Connect with User’s Identity**

1. Expand the **AWConnection** node and the **Tables** folder and select the **Contact** table.
2. Right-click the **Contact** table, and select **Create All Operations** from the context menu.

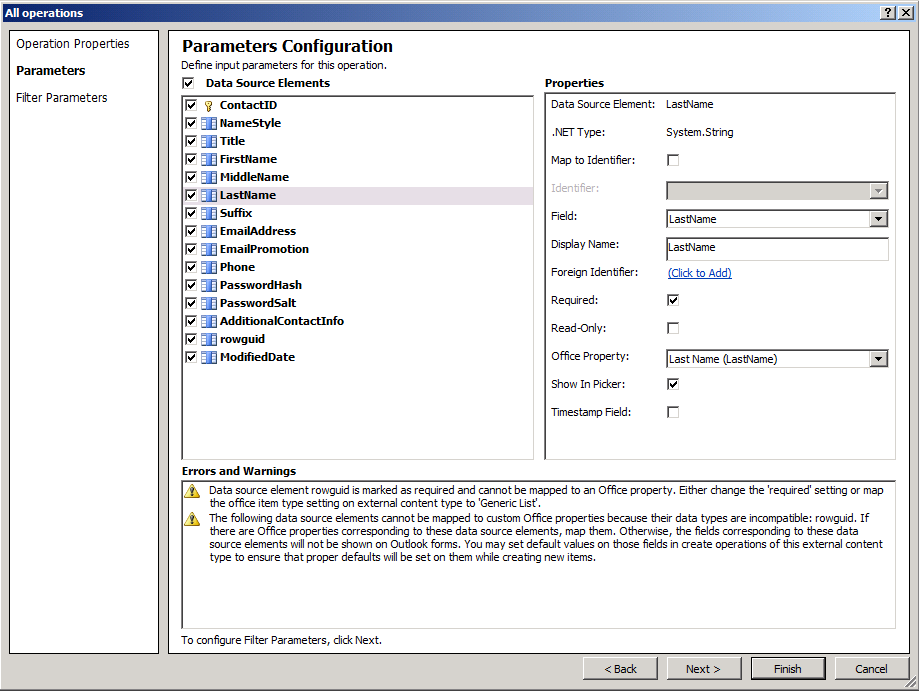


1. After clicking **Create All Operations**, SharePoint Designer will launch a wizard which leads you through three pages.
2. The first page is the **Operations Properties** dialog. There should be no errors or warning on this page. Click the **Next** button.
3. The second page is the **Parameters Configuration** dialog. There should be several warnings and an error. Move through the following steps to correct the error and two of the warnings.
   1. In the list of **Data Source Elements**,select the **LastName** field.
   2. In the **Office Property** drop-down list, select **Last Name (LastName)**.
   3. **Check** the **Show In Picker** checkbox.

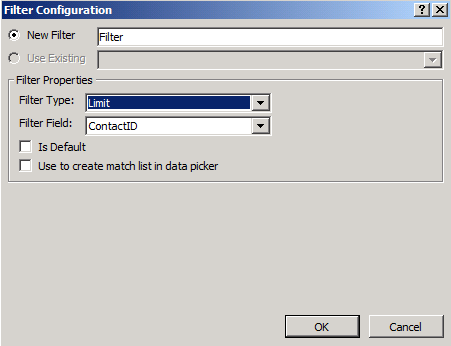


* 1. While there are still two warnings about the **rowguid** field, you can ignore them.
  2. Click the **Next** button.

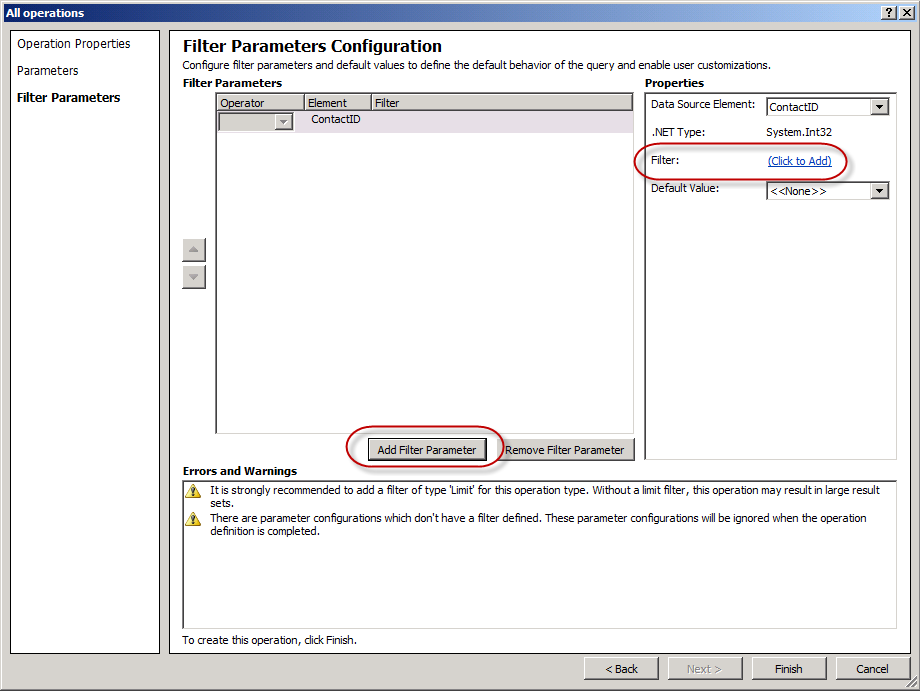
1. In the **Filter Parameters Configuration**, do the following:
2. Click the **Add Filter Parameter** button to add a new **Query Parameter**.
3. Click the **Click to Add** link to add a new filter.



* 1. Add a new **Limit** filter and click the **OK** button. A limit filter is required to ensure that the records returned do not exceed the 2000 item throttling limit imposed by BCS.

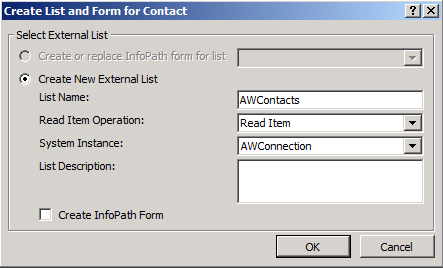


* 1. Click on **Default Value** field on the **Filter Configuration** screen and type in **100**.

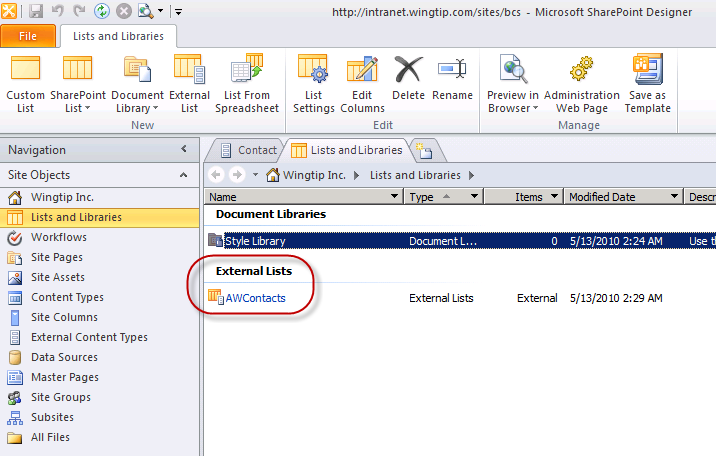


* 1. Click the **Finish** button to complete the wizard.

1. Now you must save the **Contact** external content type. You can accomplish this by clicking the **Save** button at the top of the SharePoint Designer ribbon in the Quick Access Toolbar (disk icon) when the tab for the **Contact** external tab has focus.
2. Now it is time to create a new external list to surface the data retrieved by the Contact external content type in the current SharePoint site. On the ribbon, click the **Create List and Forms** button.
3. Enter **AWContacts** in the **List Name** field, uncheck the **Create InfoPath Form** and click the **OK** button.



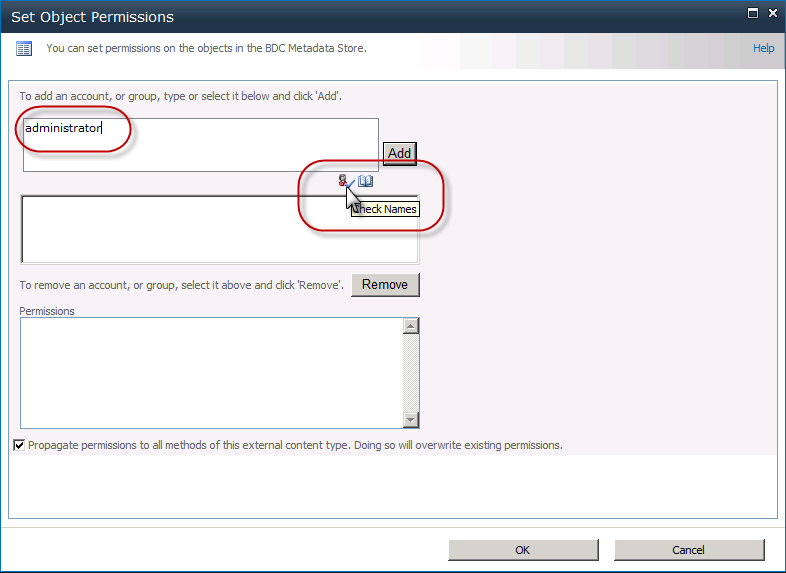
1. Verify that the external list has been created by examining the **List and Libraries** collection.



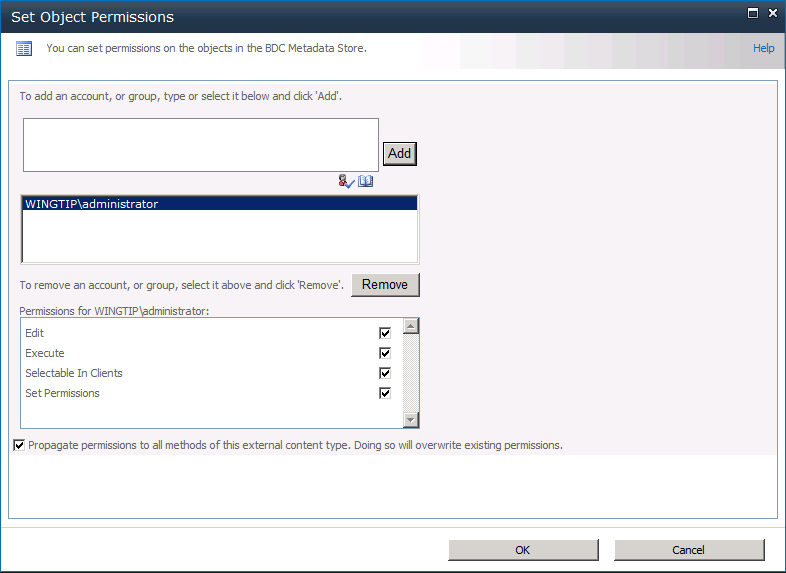
1. Once you have successfully created the **AWContacts** external list, close SharePoint Designer.

**Note:** Once the External Content Types and Related Items are created, you will have a new entity-backed list available in SharePoint. In this part of the exercise, you’ll work with the new list inside the browser.

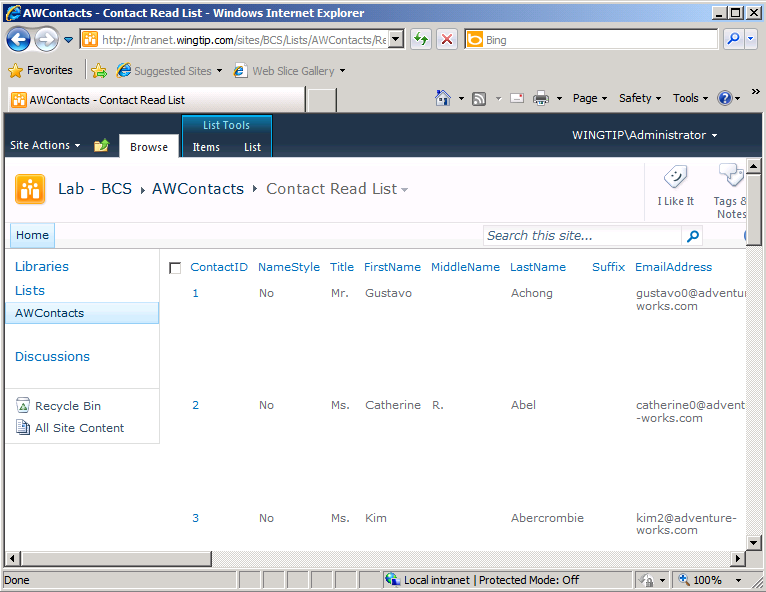
1. Make sure you are in **Internet Explorer** inside the site **http://intranet.wingtip.com/sites/BCS.**
2. Before accessing the data defined in the external content type you just created, you first need to grant permissions to a user who can use the external content type. This is to ensure only select users can access external data.
3. Open **SharePoint 2010 Central Administration**.
4. Select **Application Management » Manage Service Applications**.
5. Select the **Business Data Connectivity Service** and click the **Manage** button in the ribbon.
6. Select the **Contact** external content type and from his ECB menu (the drop down menu when you select an item in a list), select **Set Permissions**.
7. In the top-most box enter **Administrator** and click the person check icon just below the box to validate the user:



1. Once validated, click the **Add** button.
2. With the username selected, grant this user all the available permissions listed by checking each box and clicking **OK**.



1. To have this change take effect immediately, recycle IIS by going to a command prompt and typing **IISRESET.EXE**.
2. Locate the **AWContacts** list and navigate to it in the browser. You should now see information from the **Contacts** table in the **AdventureWorks** database.



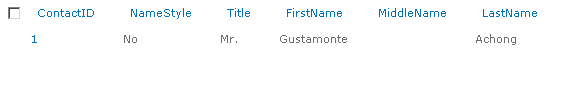
1. Now go into edit mode for the first item in the list for the contact named **Gustavo Achong**. Accomplish this by choosing the **Edit Item** command available from the ECB menu in the **ContactID** column.



1. When the edit form appears, change the contact’s first name from **Gustavo** to **Gustamonte**. Click the **Save** button to save your changes back to the **AdventureWorks** database.



1. Verify that the external list has been updated with your changes as shown below. The main point of the last few steps is that from the user’s perspective updating items in an external list is just like updating items in a native SharePoint list.

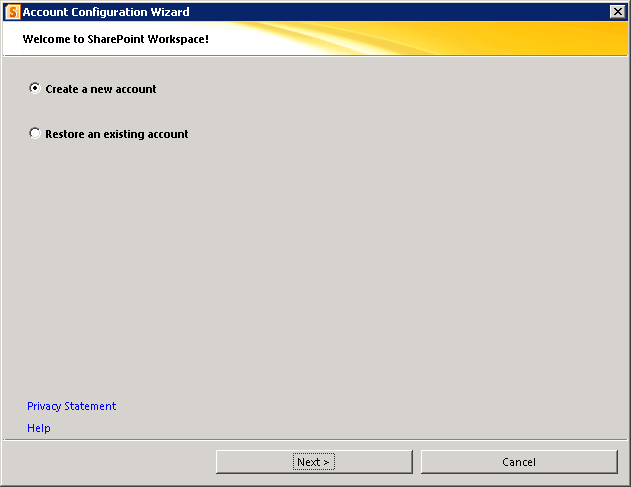


In this exercise you created an external content type and an external list surfacing data within a database as a SharePoint list.

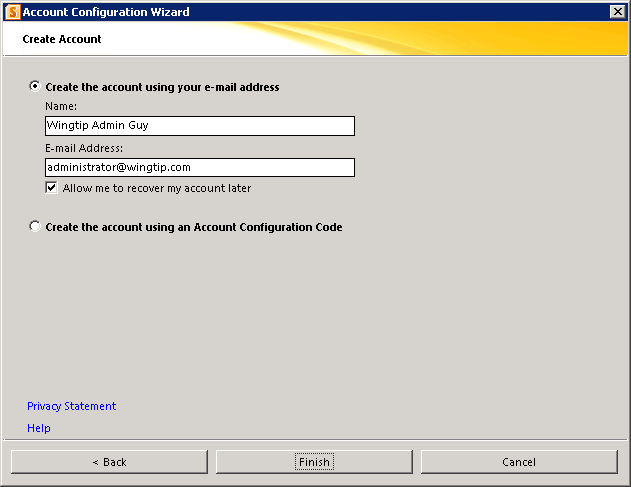
### Exercise 2: Taking External Data Offline with SharePoint Workspace 2010

In this exercise you will take the list offline that you created as an external list.

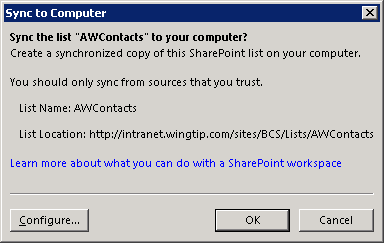
1. Launch **SharePoint Workspace 2010** (**Start » All Programs » SharePoint**). If this is the first time that SharePoint Workspace 2010 has been launched on the VM, you may have to create a new account. If the Account Configuration Wizard appears, Click **Next** when you see the **Welcome to SharePoint Workspace** page.



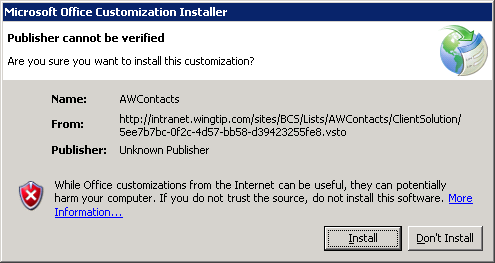
1. On the **Create Account** page, ensure the form data is filled out as shown below and click **Finish**.



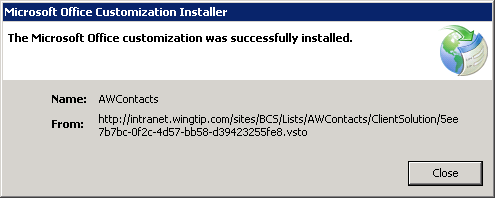
1. Now, while leaving SharePoint Workspace open, return to the browser and navigate to the external list you created earlier named **AWContacts**.
2. In the page for the **AWContacts** list, use the ribbon and select **List » Sync to SharePoint Workspace** to open the list in the SharePoint Workspace. You will be prompted with the following dialog. Click **OK** to continuing with the synchronization process.



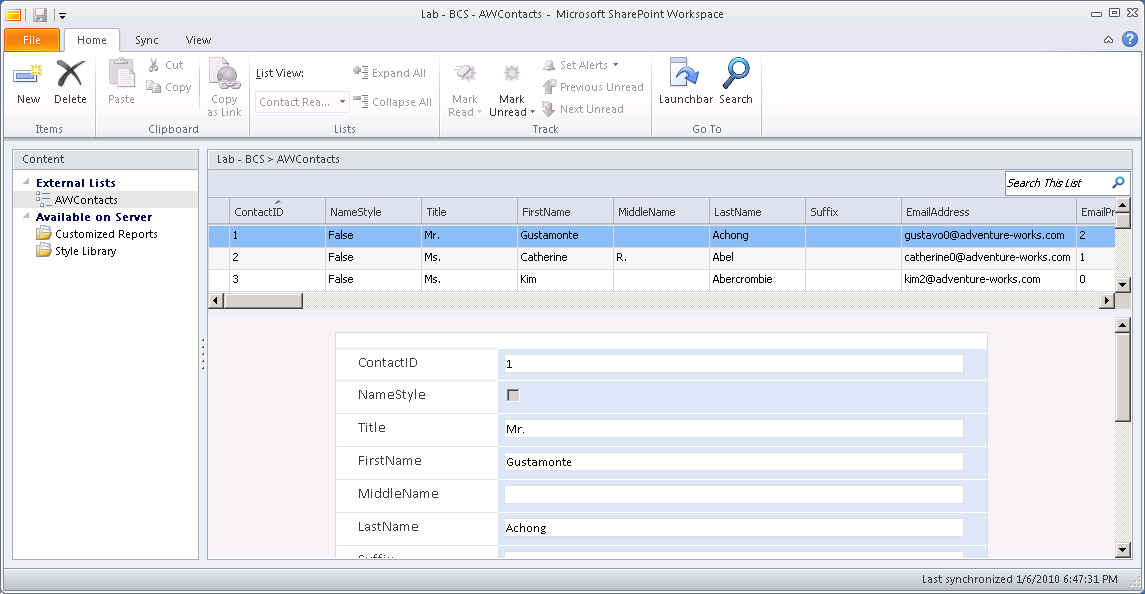
1. You may be prompted with this second dialog to install a client-side component which assists with SharePoint Workspace synchronization. Click **Install**.



1. After the installation process is complete, the following dialog appears. Click **Close**.



1. When the synchronization process finishes, navigate back to SharePoint Workspace and click the **AWContacts** list to make sure you can see and edit the data in the **AdventureWorks** database.



In this exercise you took an external content type, the associated forms and external list and sync’d the list with SharePoint Workspace 2010 for offline use.

### Exercise 3: Modeling BCS Applications by Creating a .NET Assembly Connector

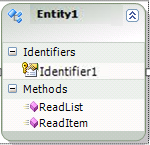
In this exercise you will create a Business Data Catalog Model with Visual Studio that will allow clients to access external data within a SharePoint site.

1. Launch **Visual Studio 2010** and create a new **Visual C# » SharePoint » 2010 »** **Business Data Connectivity Model** by selecting **File » New » Project** and give it a name of **ProductModel**.
2. Complete the wizard that appears using the following information.

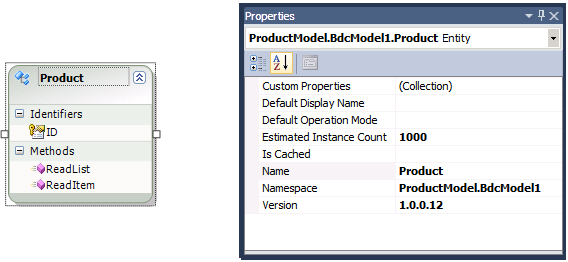
**Debugging site:** http://intranet.wingtip.com/sites/BCS

**Deploy as a farm solution**: selected

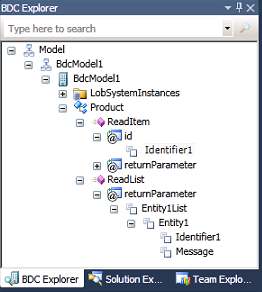
1. When the project opens, you will see the definition for a single external content type named **Entity1.**



1. First, rename the provided external content type to something more useful. Right-click **Entity1** in the **Designer** and select **Properties**. Then, in the **Properties** window, change the name of the entity to **Product**.

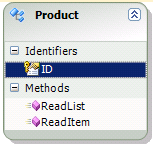


1. Now you need to make changes to the external content type. This is done using the BDC Explorer tool window.
2. First, open the **BDC Explorer** tool window. Expand every node in the model so that you can see the entire model.

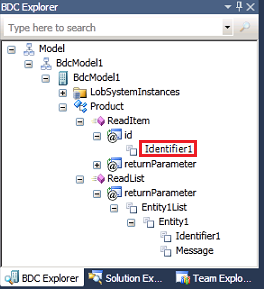


**Note**: If the BDC Explorer tool window isn’t visible, select **View » Other Windows » BDC Explorer** within Visual Studio 2010 to open this tool window.

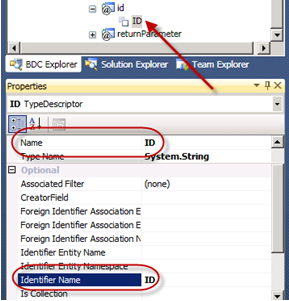
1. When you click on a method, you also get a **BDC Method Details** view at the bottom of the developer environment. From here you can edit the type descriptors, identifiers, return types, and add and modify methods.
2. First change the name of the **Identifier1**.
3. Select the **Product** entity in the designer.
   1. Right click the **Identifier1** identifier the **Product** entity and select **Properties**.
   2. Set its **Name** property to **ID**.



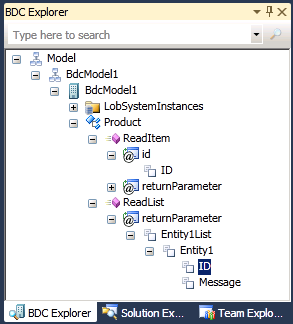
1. Now change the input value used for finding a single record in the external content type.
2. In the **BDC Explorer** tool window, click on the **ReadItem » id » Identifier1** node.



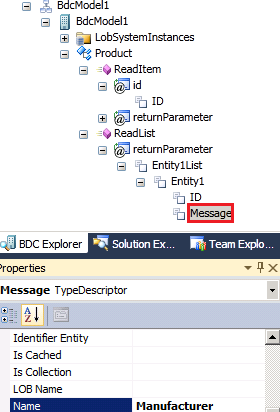
1. In the Property tool window, change his **Name** and **Identifier** properties to **ID**:



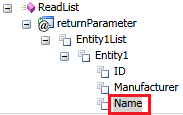
1. Now you need to make the exact same changes to the ReadList() method’s returnParameter.



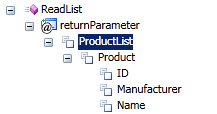
1. Let’s also change the Message property and add a property to the entity. First make changes to the ReadList() method and then we’ll copy them over to the ReadItem() method.
2. Right-click the Message node under ReadList » returnParameter » Entity1List » Entity1, select **Properties** and change its Name to Manufacturer.



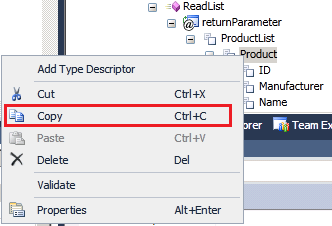
1. Next, right-click the Entity1 node under the ReadList() method and select **Add Type Descriptor**. Assign the new node Name.



1. Using the same method, rename **Entity1** into **Product** and rename **Entity1List** into **ProductList**.



1. With these changes made, let’s apply them to the ReadItem() method.
2. Right-click **ReadList » returnParameter » ProductList » Product** node and select **Copy**.



1. Right-click the **ReadItem » returnParameter** and select **Paste**. When prompted to replace the type description, select **Yes**.

With the entity and its associated finder methods created, it’s now time to write the code for the .NET Assembly Connector.

1. In the Solution Explorer, right-click the Entity1.cs file and choose **Rename**. Give it the name Product.cs. Answer **Yes** in the popup dialog.
2. In the **Solution Explorer**, double-click the Product.cs file to open the code window.
3. Change the property definitions to appear as follows:

public partial class Product

{

public string ID { get; set; }

public string Name { get; set; }

public string Manufacturer { get; set; }

}

1. In the **Solution Explorer**, right-click the Entity1Service.cs file and rename it to ProductService.cs.

**Note**: This may have been done by Visual Studio 2010’s built in refactoring capabilities when you renamed the Product object.

1. In the **Solution Explorer**, open the code window for the ProductService.cs file.
2. Add to the top of the file: **using System.Xml.Linq;**
3. Delete all of the existing code within the class and replace it with the following code. Note that you can copy and paste this code from the snip1.txt file in the [[LAB FILES]]\StarterFiles folder in your lab folder.

public class ProductService

{

private static string GetData()

{

StringBuilder xml = new StringBuilder();

xml.Append("<Products>");

xml.Append("<Product ID=\"1\" Manufacturer=\"Microsoft\" Name=\"XBox-360\" />");

xml.Append("<Product ID=\"2\" Manufacturer=\"Seagate\" Name=\"Harddrive\" />");

xml.Append("<Product ID=\"3\" Manufacturer=\"Dell\" Name=\"Laptop\" />");

xml.Append("<Product ID=\"4\" Manufacturer=\"Microsoft\" Name=\"Zune\" />");

xml.Append("</Products>");

return xml.ToString();

}

public static Product ReadItem(string id)

{

XDocument d = XDocument.Parse(GetData());

var q = from c in d.Descendants("Product")

where c.Attribute("ID").Value == id

select new

{

ID = c.Attribute("ID").Value,

Name = c.Attribute("Name").Value,

Manufacturer = c.Attribute("Manufacturer").Value

};

Product product = new Product();

product.ID = q.First().ID;

product.Name = q.First().Name;

product.Manufacturer = q.First().Manufacturer;

return product;

}

public static IEnumerable<Product> ReadList()

{

XDocument d = XDocument.Parse(GetData());

var q = from c in d.Descendants("Product")

select new

{

ID = c.Attribute("ID").Value,

Name = c.Attribute("Name").Value,

Manufacturer = c.Attribute("Manufacturer").Value

};

List<Product> products = new List<Product>();

foreach (var p in q)

{

Product product = new Product();

product.ID = p.ID;

product.Name = p.Name;

product.Manufacturer = p.Manufacturer;

products.Add(product);

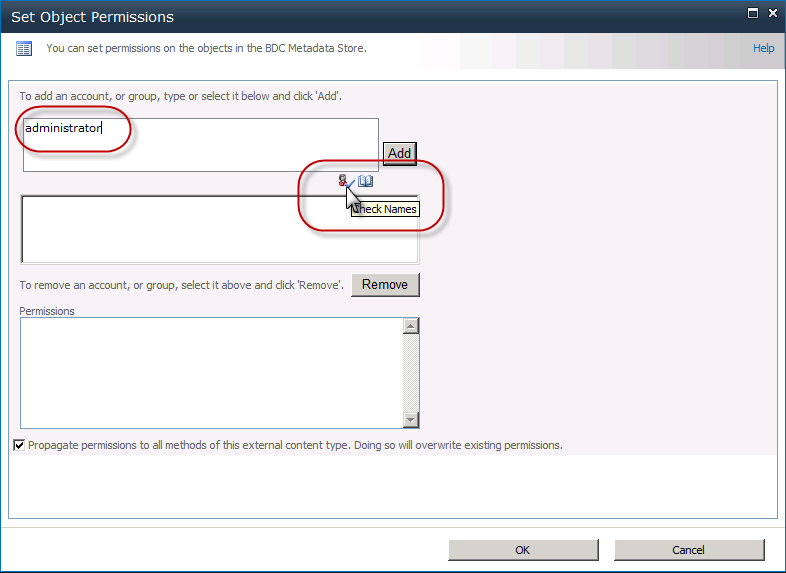
}

return products;

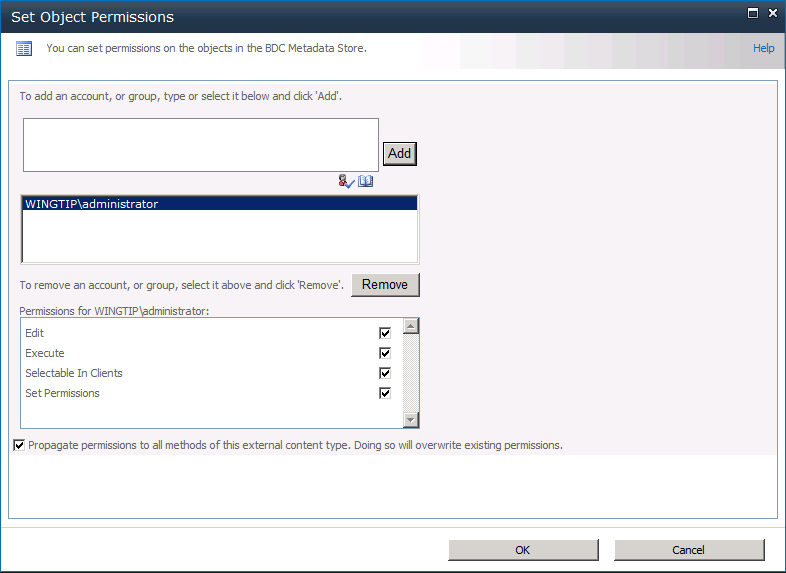
}

}

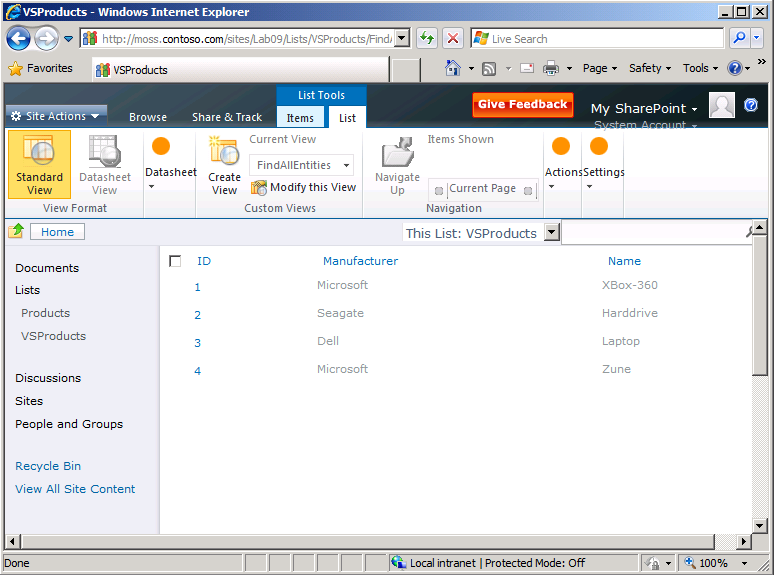
1. When the project is complete, try to build the project.
2. Once the project builds successfully, right click the project in the **Solution Explorer** and select **Deploy** from the context menu.
3. Before accessing the data defined in the external content type you just created, you first need to grant permissions to a user who can use the external content type. This is to ensure only select users can access external data.
4. Open **SharePoint 2010 Central Administration**.
5. Select **Application Management » Manage Service Applications**.
6. Select the **Business Data Connectivity Service** and click the **Manage** button in the ribbon.
7. Select the **Product** external content type and from his ECB menu (the drop down menu when you select an item in a list), select **Set Permissions**.
8. In the top-most box enter **Administrator** and click the person check icon just below the box to validate the user:



1. Once validated, click the **Add** button.
2. With the username selected, grant this user all the available permissions listed by checking each box and clicking **OK**.



1. To have this change take effect immediately, recycle IIS by going to a command prompt and typing **IISRESET.EXE**.
2. In the browser go back to the **http://intranet.wingtip.com/sites/BCS** site and create a new **External List** based on the deployed **Product** model using the same process as the previous exercise.



In this exercise you created a custom .NET Assembly Connector external content type in Visual Studio and an External Lists based off this external content type.