## Developing Apps with SharePoint Services

**Lab Time**: 60 minutes

**Lab Folder**: C:\Student\Modules\SharePointServices\Labs

**Lab Overview**: In this lab you will create several different apps. Each one will utilize a different SharePoint service including User Profiles, Search, and Business Connectivity Services.

**Lab Setup**: Ensure you have a development site available for this lab

### Exercise 1: Setup Lab Environment

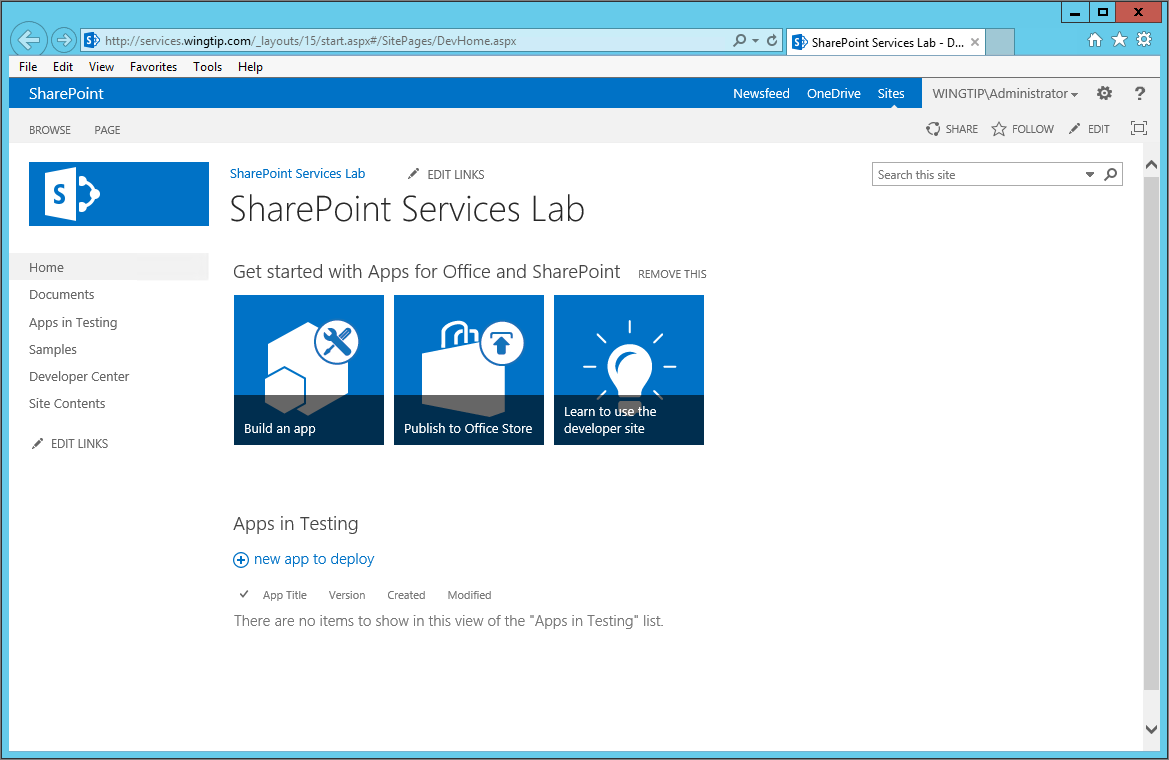
In this exercise you will setup your environment.

All exercises in this lab assume you will work in a new site collection, http://services.wingtip.com.

1. Setup a new site collection for this lab:
   1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\Administrator**.
   2. Using Windows Explorer, navigate to the folder at the following path.

C:\Student\Modules\SharePointServices\Lab\

* 1. Locate the PowerShell script named **SetupLab.ps1** and execute it by right-click itand selecting **Run with PowerShell**.
  2. When the script completes, it will launch a new browser and navigate to the lab site collection.

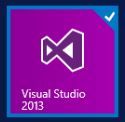


* 1. Close the PowerShell console window.

### Exercise 2: Programming with User Profiles

In this exercise you will create an App Part that welcomes the current user with information obtained from their user profile.

1. Launch **Visual Studio 2013** as administrator:
   1. **Windows** Keyboard Key 🡪 Right click on the **Visual Studio 2013** tile and select **Run as administrator**.



1. Open the starter solution in Visual Studio 2013:
   1. In Visual Studio select **File 🡪 Open 🡪 Project**./ **Solution**
   2. In the **Open Project** dialog:
      1. Browse to the **Starter Files** folder and locate the file **WelcomeClientWebPart.sln**.
      2. **Open** the solution.
2. Update the Site URL
   1. In the **Solution Explorer**, click the **WelcomeClientWebPart** project node.
   2. Select **View 🡪 Properties Window** from the Main Menu.
   3. In the Properties Window, locate the **Site URL** property, and update it to refer to the SharePoint Services site: <http://services.wingtip.com>
3. Add a Client Web Part
   1. In the **Solution Explorer**, right click the project node.
   2. Select **Add 🡪 New Item** from the context menu.
   3. In the Add New Item dialog, select **Client Web Part (Host Web)**.
   4. **Name** the Client Web Part **Welcome**.
   5. Click **Add**.
   6. When prompted enter the following values:
      1. Select **Create a new client web part page for the client web part content**
      2. Page name: **Welcome**.
   7. Click **Finish**.
4. Open **Welcome.aspx** for editing.
   1. **Add** the following meta tag directly beneath the head tag to ensure Internet Explorer does not revert to IE8 behavior.

<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE9" >

* 1. In the Solution Explorer, expand the Scripts folder and note the current jQuery-#.#.#.min.js number.
  2. Update the jQuery reference in Welcome.aspx to match the version needed by the project, which is 1.6.2.

<script type="text/javascript" src="../Scripts/jquery-1.6.2.min.js"></script>

* 1. **Add** the following script references directly under the existing script references.

<script type="text/javascript" src="../Scripts/knockout-2.1.0.js"></script>

<script type="text/javascript" src="../Scripts/WelcomeViewModel.js"></script>

<script type="text/javascript" src="../Scripts/App.js"></script>

* 1. **Add** the following HTML in the body section of the page.

<div>

<article>

<div id="portrait" style="text-align:center;">

<figure>

<img src="#" data-bind="attr: { src: get\_pictureUrl() }" alt="photo" />

</figure>

</div>

<div style="text-align:center;">

<figcaption>

Welcome <span data-bind="text: get\_displayName()"></span>

</figcaption>

</div>

</article>

</div>

1. **Edit** the Client Web Part CAML:
   1. In the Solution Explorer:
      1. Expand the **Welcome** node.
      2. Double-click the **elements.xml** file.
   2. **Replace** the **ClientWebPart** element with the following CAML to define a property named **ShowImage**.

<ClientWebPart

Name="WelcomeAppPart"

Title="Welcome App Part"

Description="A web part to welcome users"

DefaultWidth="300"

DefaultHeight="200">

<Content

Type="html"

Src="~appWebUrl/Pages/Welcome.aspx?{StandardTokens}&amp;ShowImage=\_ShowImage\_" />

<Properties>

<Property

Name="ShowImage"

Type="boolean"

WebBrowsable="true"

WebDisplayName="Show Image"

WebDescription ="Determines whether the user's picture is displayed"

WebCategory="Configuration"

DefaultValue="true"

RequiresDesignerPermission="true"

PersonalizableIsSensitive="false"

PersonalizationScope="shared"/>

</Properties>

</ClientWebPart>

The **ShowImage** property will be accessible from the app part properties in the host web. The property is a Boolean that specifies whether the app part should display the current user’s portrait.

1. Code the **init** function
   1. In the Solution Explorer:
      1. Expand the **Scripts** node.
      2. Double-click the **WelcomeViewModel.js** file.
   2. Locate the comment **//CODE GOES HERE** within the init function.
   3. Add the following code to complete the **init** function

$.ajax({

url: Wingtip.Utilities.getQueryStringParameter("SPAppWebUrl") +

"/\_api/SP.UserProfiles.PeopleManager/GetMyProperties",

method: "GET",

headers: { "accept": "application/json;odata=verbose" },

success: function (data) {

displayName(data.d.DisplayName);

pictureUrl(data.d.PictureUrl);

},

error: function (err) {

alert(JSON.stringify(err));

}

});

Note in the above code how the jQuery Ajax function calls the User Profile service. The return values are then set into knockout observables that will cause the display to immediately update with the name and picture of the current user.

1. Code the **ready** function
   1. In the Solution Explorer:
      1. Expand the **Scripts** node.
      2. Double-click the **Apps.js** file.
   2. Locate the comment **//CODE GOES HERE** within the ready function.
   3. Add the following code to complete the **ready** function

Wingtip.WelcomeViewModel.init();

var showImage = Wingtip.Utilities.getQueryStringParameter("ShowImage");

if (showImage=="true")

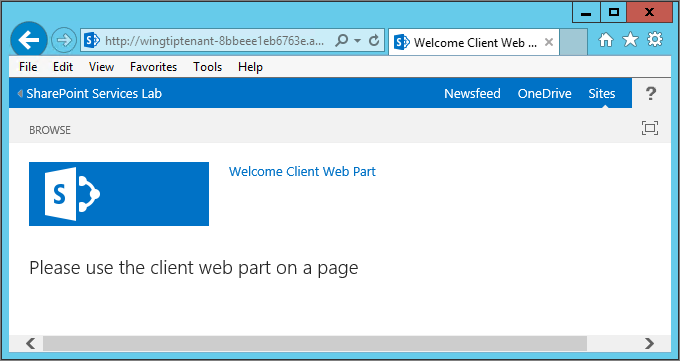
$("#portrait").show();

else

$("#portrait").hide();

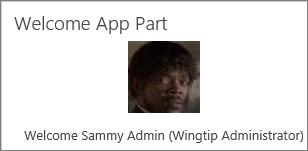
ko.applyBindings(Wingtip.WelcomeViewModel);

1. Run the app:
   1. Press the **F5** key.
   2. When the app starts, click **Trust It**.
   3. Your app should appear as below:

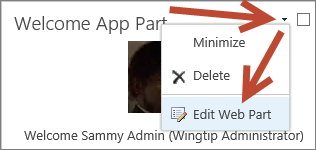


* 1. Return to the host web (by clicking on the **SharePoint Services Lab** name located above the Browse link in the upper left corner of the page).

1. Add the App Part:
   1. In the host web, select **Page 🡪Edit** from the ribbon.
   2. Click the **Insert** tab in the ribbon.
   3. Click the **App Part** button.
   4. Select the **Welcome App Part** in the catalog and click the **Add** button.
      1. You should now see A welcome message for Sammy Admin along with his picture as shown below:



* 1. Place the App Part in edit mode and change the value of the **Show Image** property to see the effect.
     1. Use the screenshot below to see how to place the App Part in edit mode:



* + 1. **Show Image** is a check box located under the **Configuration** Section of the App Part Editor
    2. Remove the check from the **Show Image** checkbox and click the **Apply** button to see the difference.

1. Close Internet Explorer to stop debugging the application and close Visual Studio.

In this exercise we created a Welcome App Part that displayed a simple message to users along with their User Profile Picture.

### Exercise 3: Executing Queries against the SharePoint Search Service

In this exercise you will create an “Employee Directory” app that displays an A through Z “Rolodex”-style interface. The app will execute queries against SharePoint 2013 people search to display employee contact information.

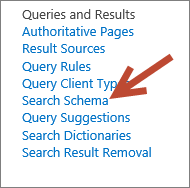
1. Edit Managed Properties to support sorting:
   1. Open the Central Administration home page (Press the Windows key and type “central” then click on the SharePoint 2013 Central Administration tile).



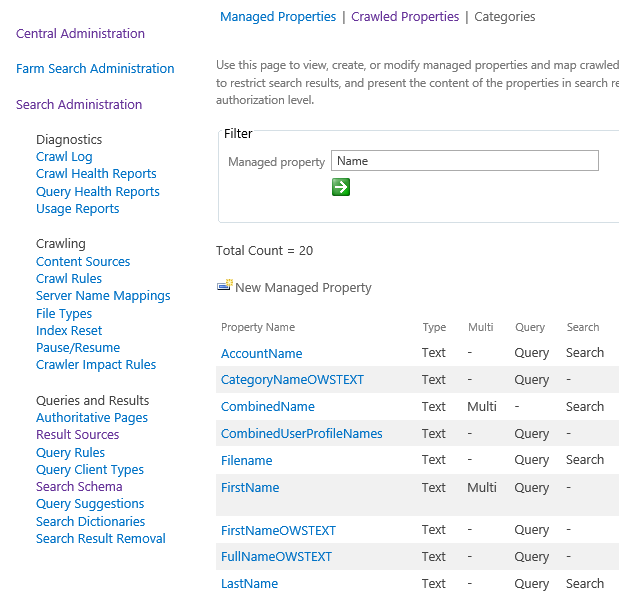
* 1. Click **Manage Service Applications (**located underneath **Application Management)**
  2. Click **Search Service Application**



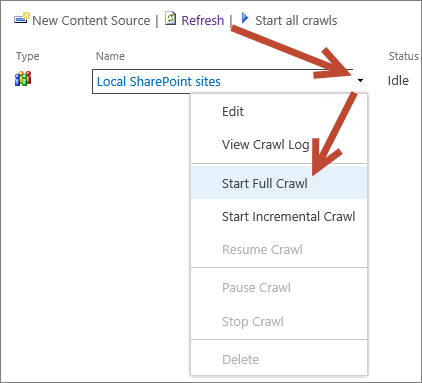
* 1. Click **Search Schema.**



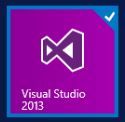
* 1. Enter **Name** in the filter criteria and press Enter or click Apply



* 1. Edit the **LastName** Managed property
     1. Click the **LastName** Managed Property.
     2. Change the Sortable property to **Yes – Active.**
     3. Click **OK**.
  2. Edit the **FirstName** Managed property
     1. Again enter **Name** in the filter criteria and press Enter or click Apply
     2. Click the **FirstName** Managed Property.
     3. If needed, change the Sortable property to **Yes – Active**.
     4. Click **OK**.
  3. Initiate a Full Crawl
     1. Click **Content Sources** (located under the Crawling section of the left navigation area)
     2. Using the drop-down menu associated with the Local SharePoint Sites content source, select **Full Crawl**.  
        (Note: this will take ~5 minutes to run… but we can continue with the lab at this point without waiting for this to finish)



1. Launch **Visual Studio 2013** as administrator (if it is not already open):
   1. **Windows** Keyboard Key 🡪 Right click on the **Visual Studio 2013** tile and select **Run as administrator**.



1. Open the starter solution in Visual Studio 2013:
   1. In Visual Studio select **File 🡪 Open 🡪 Project/ Solution**.
   2. In the **Open Project** dialog:
      1. Browse to the **Starter Files** folder, **EmployeeDirectory** folder, and locate the file **EmployeeDirectory.sln**.
      2. **Open** the solution.
2. Update the Site URL
   1. In the **Solution Explorer**, click the project node.
   2. Select **View 🡪 Properties Window** from the Main Menu.
   3. In the Properties Window, locate the **Site URL** property, and update it to refer to the [**http://services.wingtip.com**](http://services.wingtip.com) site where you will deploy the completed app.
3. Request App Permissions
   1. In the **Solution Explorer**, double click the **AppManifest.xml** file.
   2. Click the **Permissions** tab.
   3. In the **Scope** box, select **Search**.
   4. In the **Permission** box, select **QueryAsUserIgnoreAppPrincipal**.
   5. Save and close the **AppManifest.xml** file
4. Code the **ContactViewModel.js** library
   1. In the Solution Explorer:
      1. Expand the **Scripts** node.
      2. Expand the **ViewModels** folder
      3. Double-click the **ContactViewModel.js** file.
   2. Locate the comment **//CODE GOES HERE** within the **load** function.
   3. Add the following code to complete the **load** function

$.ajax({

url: \_spPageContextInfo.webAbsoluteUrl + "/\_api/search/query?querytext='LastName:"

+ query

+ "\*'&selectproperties='LastName,FirstName,JobTitle,WorkEmail,WorkPhone'"

+ "&sourceid='B09A7990-05EA-4AF9-81EF-EDFAB16C4E31'"

+ "&sortlist='LastName:ascending,FirstName:ascending'",

method: "GET",

headers: { "accept": "application/xml" },

success: onSuccess,

error: onError

});

In the above code, the variable named “query” will contain a letter A though Z depending upon what letter is clicked by the user. Also note the Source ID of the “People” source; this GUID is the same in every SharePoint farm.

* 1. Locate the comment **//CODE GOES HERE** within the **onSuccess** function.
  2. Add the following code to complete the **onSuccess** function

contacts.removeAll();

$(data).find("d\\:Rows").children("d\\:element").each(

function () {

$(this).find("d\\:Cells").each(

function () {

var contact = new EmployeeDirectory.Contact;

$(this).find("d\\:element").each(

function () {

if ($(this).children("d\\:Key").first().text() == "LastName")

contact.set\_lname($(this).children("d\\:Value").first().text());

if ($(this).children("d\\:Key").first().text() == "FirstName")

contact.set\_fname($(this).children("d\\:Value").first().text());

if ($(this).children("d\\:Key").first().text() == "JobTitle")

contact.set\_title($(this).children("d\\:Value").first().text());

if ($(this).children("d\\:Key").first().text() == "Path")

contact.set\_path($(this).children("d\\:Value").first().text());

if ($(this).children("d\\:Key").first().text() == "WorkEmail")

contact.set\_email($(this).children("d\\:Value").first().text());

if ($(this).children("d\\:Key").first().text() == "WorkPhone")

contact.set\_phone($(this).children("d\\:Value").first().text());

}

);

contacts.push(contact);

}

);

}

);

Note that this sample returns the search results as XML. The XML is then parsed using jQuery.

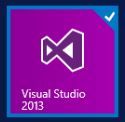
1. Run the app:
   1. Press the **F5** key.
   2. When the app starts, click **Trust It**.
   3. Click on the tabs and verify that results are returned. (Hint: try the letters A, B, D, H, M, P, R, and S as Wingtip Toys is still a fairly small company and doesn’t have too many employees yet).
2. Close Internet Explorer to stop debugging the application and close Visual Studio.

Congratulations on finishing your second SharePoint Services App that makes use of the User Profile Data by using People Search!

### Exercise 4: Creating an App that uses the Business Connectivity Service (BCS)

In this exercise you will create an app that uses a publically-available OData source to create a dashboard.

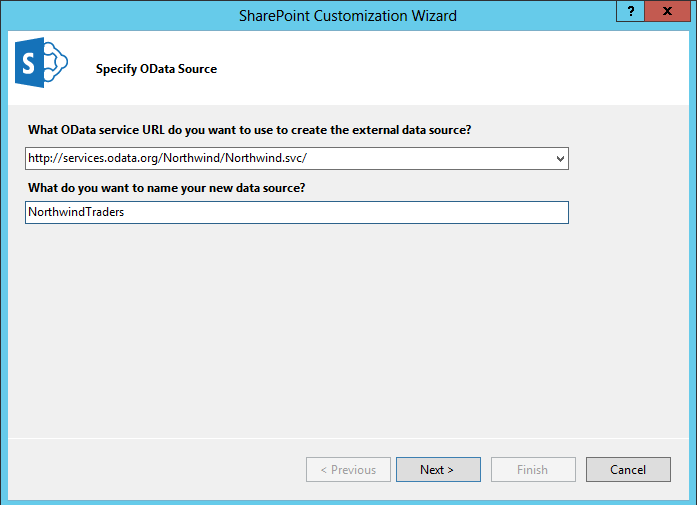
1. Launch **Visual Studio 2013** as administrator:
   1. **Windows** Keyboard Key 🡪 Right click on the **Visual Studio 2013** tile and select **Run as administrator**.



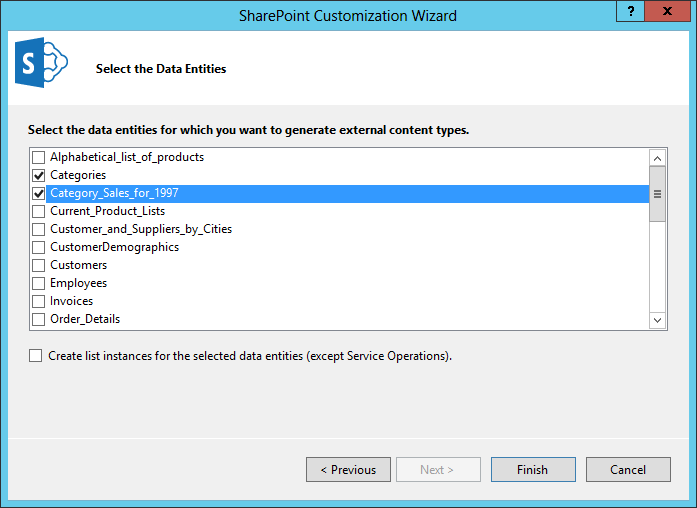
1. Open the starter solution in Visual Studio 2013:
   1. In Visual Studio select **File 🡪 Open 🡪 Project/Solution**.
   2. In the **Open Project** dialog:
      1. Browse to the **Starter Files** folder, **BCSDashboardJSOM** folder, and locate the file **BCSDashboardJSOM.sln**.
      2. **Open** the solution.
2. Update the Site URL
   1. In the **Solution Explorer**, click the project node.
   2. Select **View 🡪 Properties Window** from the Main Menu.
   3. In the Properties Window, locate the **Site URL** property, and update it to refer to the [**http://services.wingtip.com**](http://services.wingtip.com) site where you will deploy the completed app.
3. Create External Content Types
   1. In the **Solution Explorer**, right-click the **BCSDashboardJSOM** project node.
   2. Select **Add🡪 Content Types for an External Data Source** from the context menu.
   3. Enter the following URI in the field entitled **What OData Service URL do you want to use to create the external data Source?**

http://services.odata.org/Northwind/Northwind.svc/

* 1. Enter **NorthwindTraders** in the field entitled **What do you want to name your new data source?**



* 1. Click **Next**.
  2. Select **Categories** and **Category\_Sales\_for\_1997**.
  3. **Uncheck** the box labeled **Create list instances for the selected data entities (except Service operations)**.



* 1. Click **Finish**.

1. Add a Comparison filter
   1. In the Solution Explorer:
      1. Expand the **External Content Types** node.
      2. Expand the **NorthwindTraders** node.
      3. **Double**-click the **Category\_sales\_for\_1997.ect** file.
   2. In the designer click the link entitled **Click here to add a filter**.
      1. Set the **Filter Type** to **Comparison**.
      2. Set the **Associated Column** to **CategoryName**.
      3. Click **OK**.
2. Code the **northwind.query.js** library
   1. In the Solution Explorer:
      1. Expand the **Scripts** node.
      2. Double-click the **northwind.query.js** file.
   2. Locate the comment **//CODE GOES HERE** within the **load** function.
   3. Add the following code to complete the **getLobSystemInstances** function

var ctx = SP.ClientContext.get\_current();

// Get the ECT

ect = ctx.get\_web().getAppBdcCatalog().getEntity(entityNamespace, entityName);

ctx.load(ect);

// Get the LobSystem

lob = ect.getLobSystem();

ctx.load(lob);

// Save the LobSystemInstances as a property of the Deferred object

deferredLobSystemInstances.collection = lob.getLobSystemInstances();

ctx.load(deferredLobSystemInstances.collection);

ctx.executeQueryAsync(onLobSystemInstancesSuccess, onLobSystemInstancesError);

return deferredLobSystemInstances.promise();

The above code accesses the External Content Type directly through the API. An alternative to this approach is to create an External List and use the standard List API to perform CRUD operations.

* 1. Locate the comment **//CODE GOES HERE** within the **getFilters** function.
  2. Add the following code to complete the **getFilters** function

var ctx = SP.ClientContext.get\_current();

// Get the LobSystemInstance and save it

for (var i = 0; i < lobSystemInstances.get\_count() ; i++) {

if (lobSystemInstances.get\_item(i).get\_name() === lobSystemInstanceName) {

lobi = lobSystemInstances.get\_item(i);

break;

}

}

// Save the filter collection as a new property of the Deferred object

deferredFilters.collection = ect.getFilters(methodInstanceName);

ctx.load(deferredFilters.collection);

ctx.executeQueryAsync(onGetFiltersSuccess, onGetFiltersError);

return deferredFilters.promise();

The above code is working through the External Content Type to identify the associated methods. Later this information will be used to call the methods of the ECT and return results.

* 1. Locate the comment **//CODE GOES HERE** within the **executeFinder** function.
  2. Add the following code to complete the **executeFinder** function

var ctx = SP.ClientContext.get\_current();

// set the filter, if provided

if (filterValue) {

filters.setFilterValue("CategoryNameFilter", 0, filterValue);

}

// Execute the finder method with the specified filters

// Save the results of the operation as a new property on the Deferred object

deferredFinderResults.collection = ect.findFiltered(filters, methodInstanceName, lobi);

ctx.load(deferredFinderResults.collection);

ctx.executeQueryAsync(onExecuteFinderSuccess, onExecuteFinderError);

return deferredFinderResults.promise();

The above code calls the Finder method of the ECT. This method is used to generate views of the data be returning result sets.

1. Code the **northwind.viewmodel.js** library
   1. In the Solution Explorer:
      1. Expand the **Scripts** node.
      2. Double-click the **northwind.viewmodel.js** file.
   2. Locate the comment **//CODE GOES HERE** within the **load** function for the **Northwind.CategoryViewModel**.
   3. Add the following code to complete the **load** function

var query = new Northwind.Query();

// Initialize with the ECT model metadata

query.init("NorthwindModel", "Categories", "NorthwindTraders", "ReadAllCategory");

// Get the LobSystemInstances

query.getLobSystemInstances().then(

// Success getLobSystemInstances

function () {

//Get the filters

query.getFilters(this.collection).then (

// Success getFilters

function () {

// Execute the finder

query.executeFinder(this.collection, null).then(

// Success executeFinder

function () {

categories.removeAll();

// Load available categories for the dashboard

for (var i = 0; i < this.collection.get\_count() ; i++) {

var entityInstance = this.collection.get\_item(i);

var fields = entityInstance.get\_fieldValues();

categories.push(new Northwind.Category(fields.CategoryID, fields.CategoryName));

}

},

// failure executeFinder

function (args) {

alert("Error: " + args.get\_message());

}

);

},

// failure getFilters

function (args) {

alert("Error: " + args.get\_message());

});

},

// failure getLobSystemInstances

function (args) {

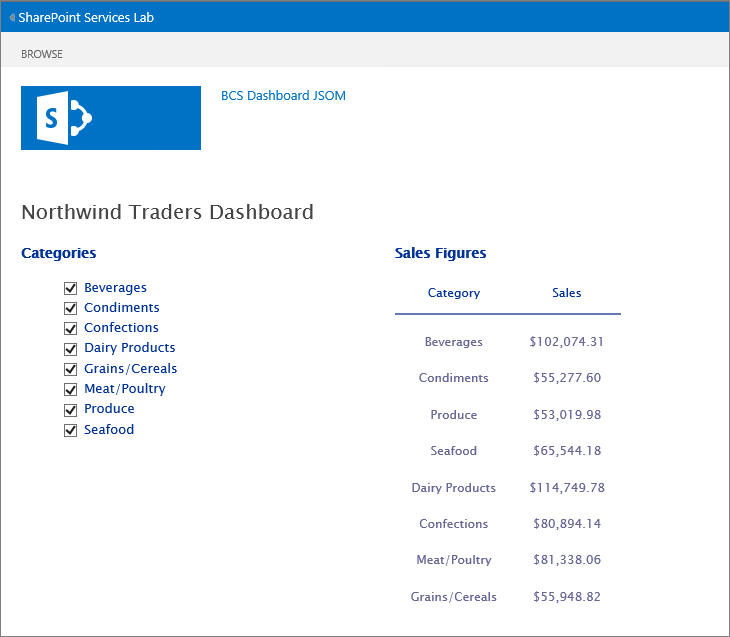
alert("Error: " + args.get\_message());

}

);

When working with the object model for External Content Types, you will have to perform several asynchronous operations. In order to sequence the operations, the above code uses promises. The promises allow the asynchronous calls to be nested so they execute sequentially.

1. Run the app:
   1. Press the **F5** key.
   2. Select Categories in the dashboard and view resulting Sales totals.



* 1. Note how the order you select the Categories determines the order of items on the Sales Figures chart.

1. When finished investigating this App:
   1. Close Internet Explorer and Close Visual Studio.

You have just finished the final SharePoint Services App for this Lab. Once you understand the basics of accessing a couple of SharePoint Services, the world is your oyster when it comes to retrieving SharePoint related information in your Apps!