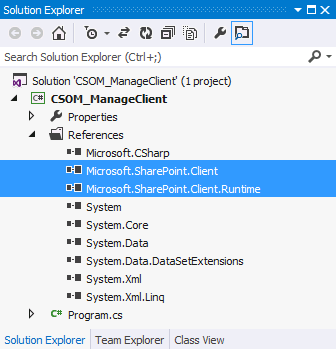
# Demo 1: Programming CSOM from a C# Project

In this Demo, you will demo a C# console application that programs against CSOM to create a new list in a SharePoint site.

## Step 1 – Open the sample Managed CSOM project and run its code

In this Step, you will use a pre-existing Console application project to demonstrate using CSOM from managed code written in C#.

1. Begin the demo by navigating to a site in the local farm such as [http://](http://w15-sp)intranet.contoso.com/sites/RESTLab. Explain that this is a SharePoint site that will be access by a managed application using CSOM.
2. Open the Visual Studio project named **CSOM\_ManageClient**. Explain to students that this is a sample console application that talks to SharePoint using CSOM. Show the references to the SharePoint assemblies.



1. Open the source file **Program.js** and ensure that it is using the correct URL for your test site. If not, update the URL to point to your local test site.

namespace CSOM\_ManageClient {

class Program {

static void Main() {

ClientContext cc = new ClientContext("http://intranet.contoso.com/sites/RESTLab");

1. Step through the code and discuss the programming logic required to create a new list using CSOM.

static void Main() {

ClientContext cc = new ClientContext("http://intranet.contoso.com/sites/RESTLab");

cc.Credentials = CredentialCache.DefaultCredentials;

Web site = cc.Web;

ListCollection lists = site.Lists;

**// load site info**

cc.Load(site);

cc.ExecuteQuery();

Console.WriteLine("Site Title: " + site.Title);

**// create list**

ListCreationInformation newList = new ListCreationInformation();

newList.Title = "Customers CSOM";

newList.Url = "Lists/Customers\_CSOM";

newList.QuickLaunchOption = QuickLaunchOptions.On;

newList.TemplateType = (int)ListTemplateType.Contacts;

site.Lists.Add(newList);

**// refresh lists collection**

cc.Load(lists);

**// make round trip to Web server to do all the work**

cc.ExecuteQuery();

**// display lists to user**

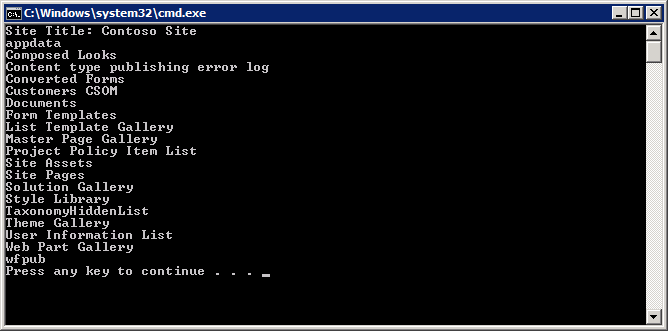
foreach (List list in lists) {

Console.WriteLine(list.Title);

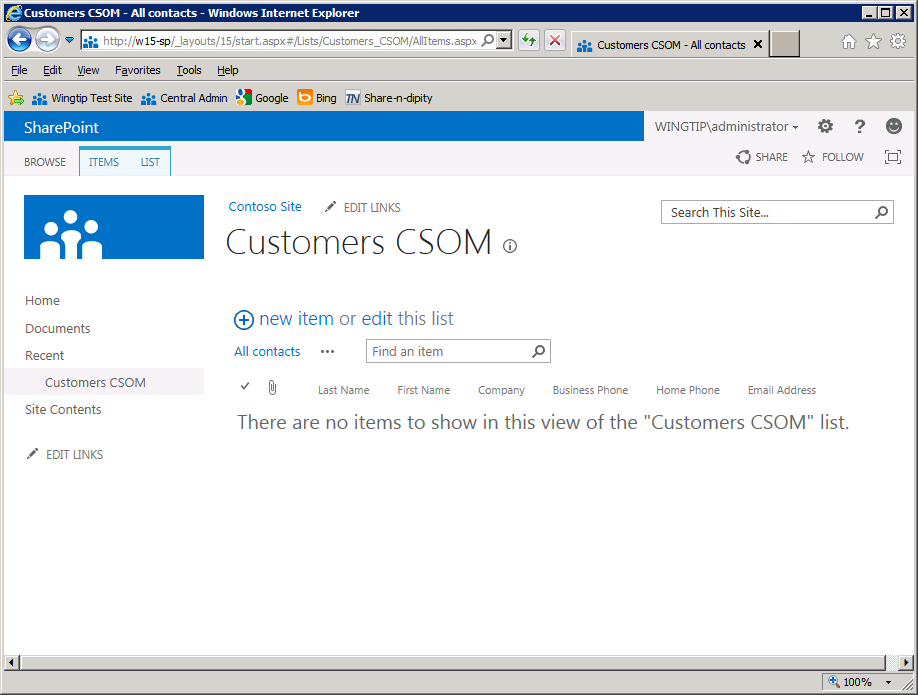
}

}

1. Run the Console app by pressing Ctrl + {F5}. The application should run and complete without errors.



1. Go to the site in the browser and verify that a list has been created.



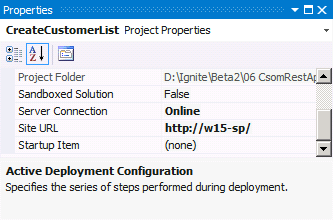
# Demo 2: Executing REST Queries through the Browser

In this Demo, you will create a sample Customers list and demonstrate how to create the URLs required by the REST API to query data using only the browser.

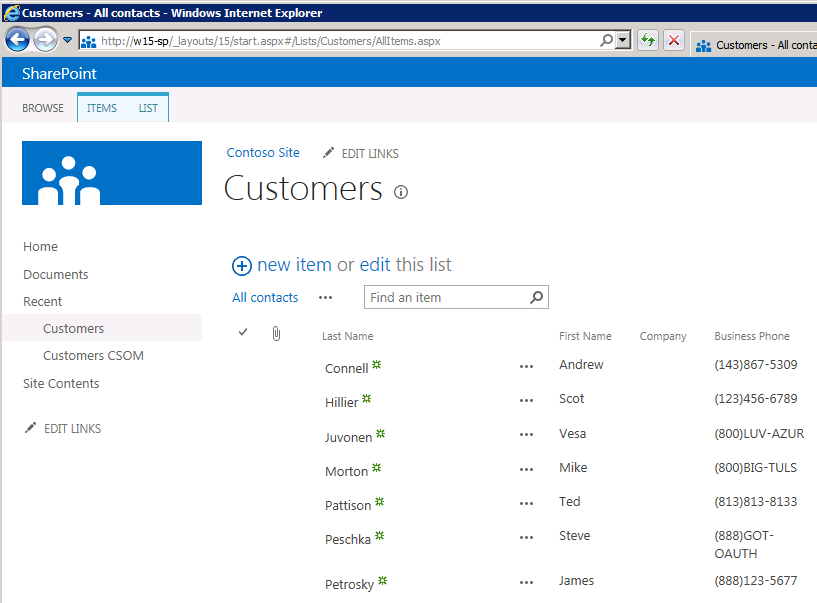
## Step 1 – Create a Customers list and query it using the REST API

In this Step, you will use a pre-existing farm solution to create a new sample list of customer items.

1. Launch Visual Studio and open the farm solution project named **CreateCustomerList.**
2. Ensure the project has a Site URL property that is configured to point to a working site in the local farm such as [http://](http://w15-sp)intranet.contoso.com/sites/RESTLab.



1. Right click on the project in the Solution Explorer and run the **Deploy** command to deploy the farm solution and to activate the feature in the target site to create a sample **Customers** list.
2. Navigate to the test site in the browser and verify that the Customer list has been created and that it contains a few Customer items.



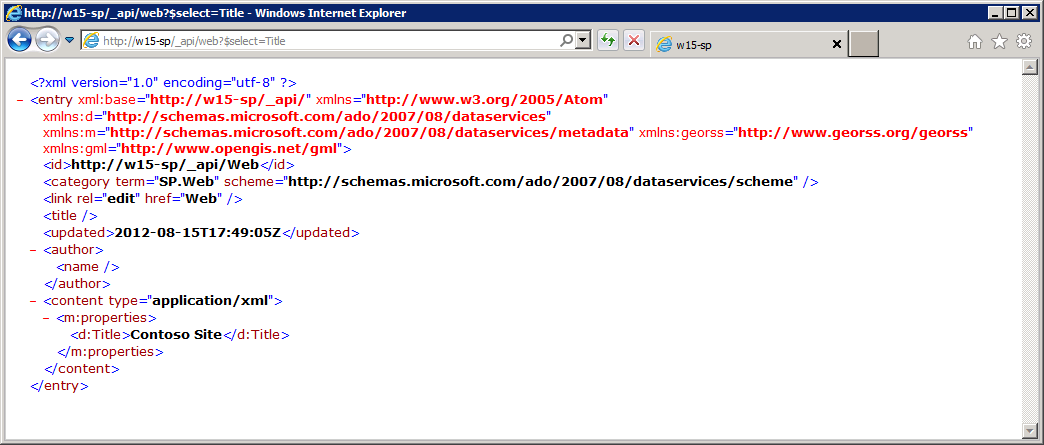
1. Run a REST API query by typing the following URL into the browser's address bar and hitting the ENTER key.

http://intranet.contoso.com/sites/RESTLab/\_api/web

1. Note that SharePoint returns quite a pit of XML. Modify the URL to request on the Title property by adding the following query string

http://intranet.contoso.com/sites/RESTLab/\_api/web?$select=Title

1. Point out to students that this results in a far more efficient result being returned.



1. Now execute the following URL to demonstrate retrieving Id property of a list.

http://intranet.contoso.com/sites/RESTLab/\_api/web/lists/getByTitle('Customers')?$select=Id

1. Now execute the following URL to demonstrate retrieving a set of list item.

http://intranet.contoso.com/sites/RESTLab/\_api/web/lists/getByTitle('Customers')/items

1. Now execute the following URL to show selecting specific columns for items. Point out that the **Title** column in a **Contacts** list such as Customers has a Display Name of Last Name but you still reference it using Title.

/getByTitle('Customers')/items?$select=FirstName,Title,WorkPhone

1. Now execute the following URL to demonstrate filtering list items. This query is for finding Customers whose last name starts with 'P'.

/getByTitle('Customers')/items?$filter=startswith(Title, 'P')

1. Now an URL to demonstrate using two query options together in a single URL.

$filter=startswith(Title, 'P')& $select=FirstName,Title,WorkPhone

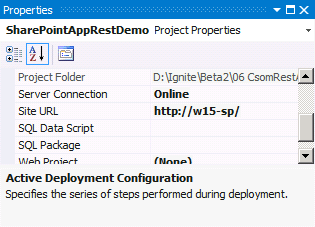
# Demo 3: Querying and Updating Content Using REST

In this Demo, you will use a pre-existing SharePoint app project to demonstrate reading and writing SharePoint list items using JavaScript and the REST API.

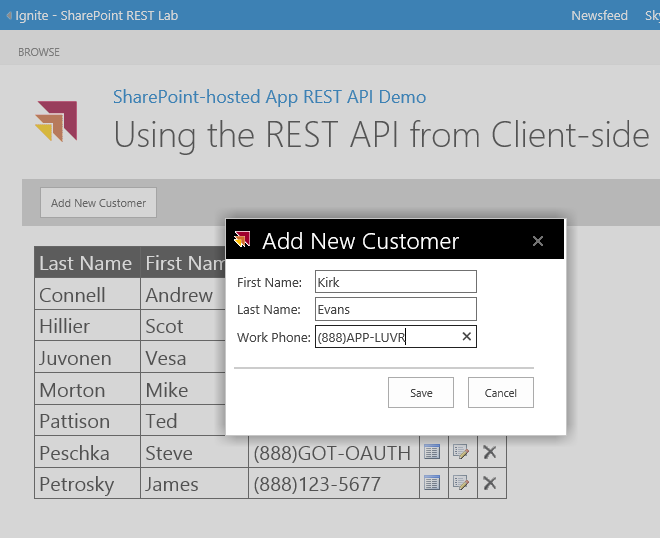
## Step 1 – Demonstrating JavaScript using the REST API

In this Step, you will open a SharePoint-hosted app project in Visual Studio and demo it to show how to program using the REST API.

1. Launch Visual Studio and open the project named **SharePointAppRestDemo**.
2. Ensure the project property named Site URL is configured to point to a test site in your local farm.



1. Press the {F5} key to run this SharePoint app in debug mode. Monitor the installation progress in the Visual Studio Output window.
2. After installation, you should be redirected to the Site Contents page of the test site where you should see an icon for the app and a caption of **SharePoint-hosted App REST Demo**. Click on that icon to redirect to the app's start page.
3. Show the students how the app works. You can see existing customers on the start page. You can add new customers by clicking the **Add New Customer** button and filling out the dialog that appears.



1. Demo how existing customer items can be edited and deleted by using the icon button in the grid.
2. Now open the JavaScript named **App.js** and explain that all the behavior is implemented by JavaScript code in this file. Walk through the code in the following methods and explain what the code is doing.
   1. getCustomers
   2. onDataReturned
   3. saveNewCustomer
   4. onDeleteCustomer
   5. onUpdateCustomer
   6. onCustomerReturned
   7. updateCustomer