

CSOM Programming



Agenda

- CSOM Fundamentals
- User and App Authentication
- CSOM Code Optimization
- Remote Exception Handling
- Creating Content Types and Lists



Why Client Object Model (CSOM)?

- Advantages of CSOM over the REST API
 - Strongly-typed programming
 - Format Digest managed automatically
 - Higher productivity when writing C# or VB
 - Provides ability to batch requests to web server
 - CSOM provides functionality beyond REST APIs
- CSOM more preferable on server-side
 - CSOM isn't great fit for JavaScript apps



Supported CSOM Functionality

- What can you do with CSOM?
 - Work within a specific site collection
 - Read and modify site properties
 - Create site columns and content types
 - Create lists, items, views and list types
 - Register remote event handlers
 - Create folder and upload and download files
 - Add web part and web part pages
 - Create new site collections



CSOM in SharePoint 2016

- CSOM Assemblies for SharePoint Foundation
 - Version 15 intended for SharePoint 2013 On-premises
 - Version 16 intended for SharePoint 2016 On-premises
 - Version 16.1 intended for SharePoint Online

- Microsoft.SharePoint.Client
- Microsoft.SharePoint.Client.Runtime

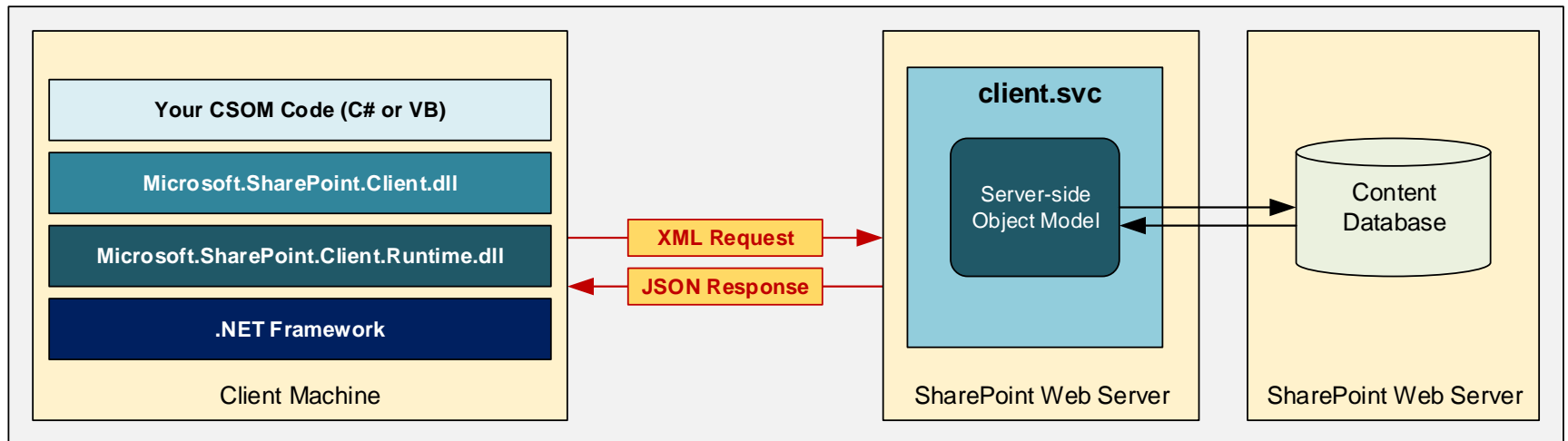
- CSOM Assemblies for SharePoint Server

- Microsoft.SharePoint.Client.DocumentManagement
- Microsoft.SharePoint.Client.Publishing
- Microsoft.SharePoint.Client.Search
- Microsoft.SharePoint.Client.Taxonomy
- Microsoft.SharePoint.Client.UserProfiles
- Microsoft.SharePoint.Client.WorkflowServices



CSOM Architecture

- CSOM Objects act as client-side proxies
 - CSOM uses Windows Communication Foundation (WCF)
 - CSOM Runtime layer handles WCF calls behind scenes
 - Request body contains XML document of instructions
 - Response returned in JavaScript Object Nation (JSON)



ClientContext

- CSOM coding starts with ClientContext
 - Provides connection to SharePoint site
 - Provides access to site and site collection
 - Provides authentication behavior
 - Provides ExecuteQuery method to call server

```
string siteUrl = "http://intranet.wingtip.com";  
ClientContext clientContext = new ClientContext(siteUrl);
```



Hello CSOM

```
using System;
using Microsoft.SharePoint.Client;

namespace HelloCSOM {
    class Program {
        static void Main() {

            ClientContext clientContext = new ClientContext("http://intranet.wingtip.com");

            Site siteCollection = clientContext.Site;
            Web site = clientContext.Web;

            clientContext.Load(siteCollection);
            clientContext.Load(site);

            clientContext.ExecuteQuery();

            Console.WriteLine("The site collection URL is " + siteCollection.Url);
            Console.WriteLine("The site title is " + site.Url);
        }
    }
}
```



Inspecting CSOM Calls with Fiddler

- ExecuteQuery triggers call to SharePoint web server
 - CSOM calls made behind the scenes using WCF
 - CSOM calls target `/_vti_bin/client.svc/ProcessQuery`
 - Can be helpful to inspect CSOM calls using Fiddler Web Debugger

```
using System;
using Microsoft.SharePoint.Client;

namespace HelloCSOM {
    class Program {
        static void Main() {

            ClientContext clientContext = new ClientContext("http://intranet.wingtip.com");

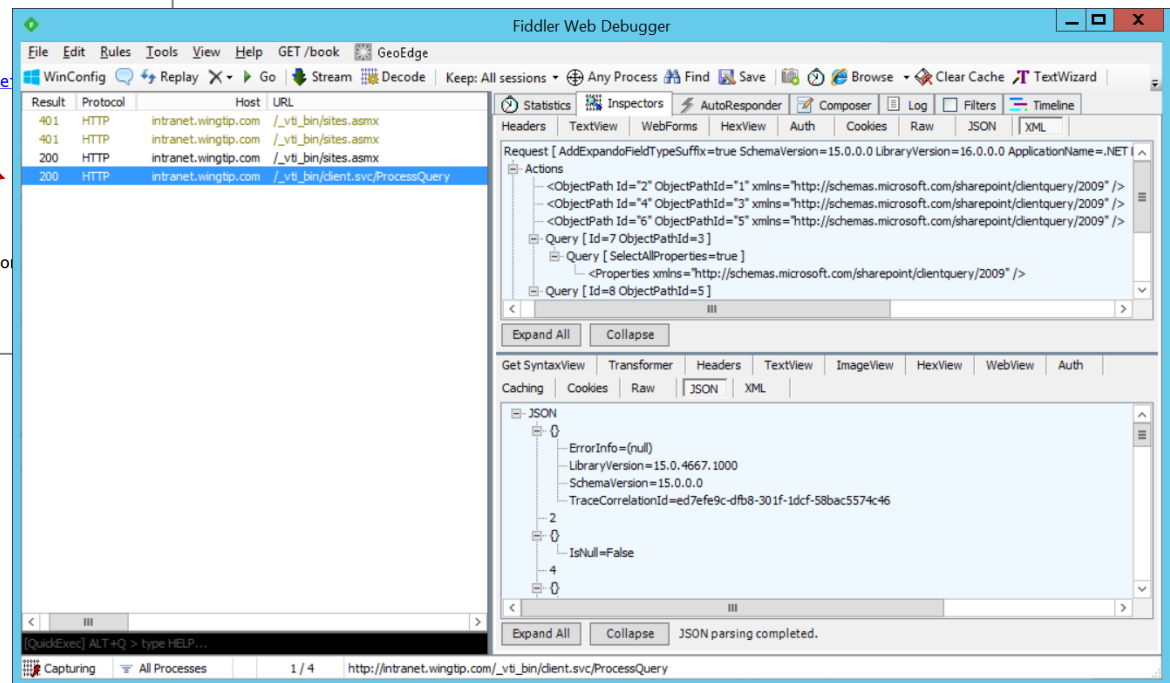
            Site siteCollection = clientContext.Site;
            Web site = clientContext.Web;

            clientContext.Load(siteCollection);
            clientContext.Load(site);

            clientContext.ExecuteQuery();

            Console.WriteLine("The site collection URL is " + siteCollection.Url);
            Console.WriteLine("The site title is " + site.Title);

        }
    }
}
```



Agenda

- ✓ CSOM Fundamentals
- User and App Authentication
 - CSOM Code Optimization
 - Remote Exception Handling
 - Creating Content Types and Lists



User Authentication (On-premises)

```
string siteUrl = "http://intranet.wingtip.com";
ClientContext clientContext = new ClientContext(siteUrl);

// set up authentication credentials
string userName = @"WINGTIP\Administrator";
string userPassword = "Password1";
clientContext.Credentials = new NetworkCredential(userName, userPassword);

// get title of the target site
Web site = clientContext.Web;
clientContext.Load(site);

// call across network
clientContext.ExecuteQuery();

// display title
Console.WriteLine(site.Title);
```



User Authentication (SPO)

```
string siteUrl = "https://SharepointConfessions.sharepoint.com";
ClientContext clientContext = new ClientContext(siteUrl);

string userName = "tedp@sharepointconfessions.onmicrosoft.com";
string userPassword = "PinkieDoo@42";
// convert password to SecureString format
SecureString secureUserPassword = new SecureString();
foreach (char c in userPassword.ToCharArray()) {
    secureUserPassword.AppendChar(c);
}

// create SharePointOnlineCredentials object to authenticate
clientContext.Credentials =
    new SharePointOnlineCredentials(userName, secureUserPassword);

// get title of the target site
Web site = clientContext.Web;
clientContext.Load(site);

// call across network
clientContext.ExecuteQuery();

// display title
Console.WriteLine(site.Title);
```



Authentication with TokenHelper

- On-premises with S2S

```
string hostWebUrl = Request.QueryString["SPHostUrl"];  
Uri hostWebUri = new Uri(hostWebUrl);  
WindowsIdentity userIdentity = Request.LogonUserIdentity;  
  
ClientContext clientContext =  
    TokenHelper.GetS2SClientContextWithWindowsIdentity(hostWebUri, userIdentity);
```

- In SharePoint Online with OAuth

```
string hostWebUrl = Request.QueryString["SPHostUrl"];  
string remoteWebUrl = Request.Url.Authority;  
  
string contextTokenString = TokenHelper.GetContextTokenFromRequest(Request);  
  
ClientContext clientContext =  
    TokenHelper.GetClientContextWithContextToken(hostWebUrl, contextTokenString, remoteWebUrl);  
  
return clientContext;
```



Authentication with SharePointContext

- SharePointContext simplifies your code
 - Automatically tracks SharePoint query string variables
 - Abstracts away issues for OAuth vs S2S
 - Provides four ways to create ClientContext

```
SharePointContext spContext =  
    SharePointContextProvider.Current.GetSharePointContext(HttpContext);  
  
// create ClientContext to access host web with [app + user] credentials  
ClientContext clientContext1 = spContext.CreateUserClientContextForSPHost();  
  
// create ClientContext to access host web with app-only credentials  
ClientContext clientContext2 = spContext.CreateAppOnlyClientContextForSPHost();  
  
// create ClientContext to access app web with [app + user] credentials  
ClientContext clientContext3 = spContext.CreateAppOnlyClientContextForSPAppWeb();  
  
// create ClientContext to access app web with app-only credentials  
ClientContext clientContext4 = spContext.CreateUserClientContextForSPAppWeb();
```



ClientContext Usage Pattern

- ClientContext is a disposable object
 - Should be disposed after you are done using it
 - Common to use within **using** statement

```
[SharePointContextFilter]
public ActionResult Index() {

    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);

    using (ClientContext clientContext = spContext.CreateUserClientContextForSPHost()) {
        // work with ClientContext inside using statement
        Web site = clientContext.Web;
        clientContext.Load(site);
        clientContext.ExecuteQuery();
        ViewBag.HostWebTitle = site.Title;
        ViewBag.HostWebUrl = site.Url;
    }
    return View();
}
```



Agenda

- ✓ CSOM Fundamentals
- ✓ User and App Authentication
- CSOM Code Optimization
 - Remote Exception Handling
 - Creating Content Types and Lists
 - Managed Metadata and Publishing



What's Wrong with This Code?

```
web site = clientContext.Web;
clientContext.Load(site);
clientContext.Load(site.Lists);

clientContext.ExecuteQuery();

string html = "<h2>List in host web</h2>";

html += "<ul>";

foreach (var list in site.Lists) {
    if (list.Hidden != true) {
        html += "<li>" + list.Title + "</li>";
    }
}

html += "</ul>";

WriteContentToPage(html);
}
```



Inspecting CSOM Calls using Fiddler

The screenshot shows the Fiddler Web Debugger interface. The left pane displays a list of intercepted requests. The right pane shows the details of the selected request, including the XML body. A red arrow points from a code snippet to the request body.

Request Details:

- Host: wingtipsrvr
- URL: /_vti_bin/client.svc/ProcessQuery
- Method: GET
- Result: 200
- Protocol: HTTP

Request Body (XML):

```
<?xml version='1.0' type='application/json'>
  <StaticProperty Id='1' TypeId='{3747adcd-a3c3-41b9-bfab-4a64dd2f1e0a}' Name='Current' xmlns='http://schemas.microsoft.com/sharepoint/clientquery/2009' />
  <Property Id='3' ParentId='1' Name='Web' xmlns='http://schemas.microsoft.com/sharepoint/clientquery/2009' />
  <Property Id='6' ParentId='3' Name='Lists' xmlns='http://schemas.microsoft.com/sharepoint/clientquery/2009' />
</StaticProperty>
```

Code Snippet:

```
web site = clientContext.Web;
clientContext.Load(site);
```



Coding with Lambda Expressions

- C# supports the use of lambda expressions
 - Syntax Introduced as part of LINQ with .NET 3.5
 - Can (and should) be used with CSOM
- Lambda expression is anonymous function
 - It defines a parameter list and a function body

```
clientContext.Load(site, s => s.Title );
```

Input Parameter(s)

Lambda Operator

Statement Block



Using Lambda Expressions

- Loading an object populates all scalar property values
 - Can result in inefficient use of network bandwidth

```
web site = clientContext.Web;  
clientContext.Load(site);  
clientContext.ExecuteQuery();
```



```
{  
  "_ObjectType": "SP.Web",  
  "AllowRssFeeds": true,  
  "AppInstanceId": "Guid(00000000-0000-0000-0000-000000000000)",  
  "Configuration": 0,  
  "Created": "Date(2013/5/31, 3, 53, 32, 0)",  
  "CustomMasterUrl": "/_catalogs/masterpage/seattle.master",  
  "Description": "",  
  "DocumentLibraryCalloutToOfficeWebAppPreviewersDisabled": false,  
  "EnableMinimalDownload": true,  
  "Id": "Guid(8e70e4a1-7528-4822-ac08-45a443d31bbd)",  
  "Language": 1033,  
  "LastItemModifiedDate": "Date(1379086272000)",  
  "MasterUrl": "/_catalogs/masterpage/seattle.master",  
  "QuickLaunchEnabled": true,  
  "RecycleBinEnabled": true,  
  "ServerRelativeUrl": "",  
  "SyndicationEnabled": true,  
  "Title": "Wingtip Team Site",  
  "TreeViewEnabled": false,  
  "UIVersion": 15,  
  "UIVersionConfigurationEnabled": false,  
  "Url": "http://wingtipsserver",  
  "WebTemplate": "STS"  
}
```

- Lambda expressions can be used to optimize
 - You can indicate which properties you want populated

```
web site = clientContext.Web;  
clientContext.Load(site, s => s.Title);  
clientContext.ExecuteQuery();
```



```
{  
  "_ObjectType": "SP.Web",  
  "Title": "Wingtip Team Site"  
}
```



Using Where() and Include()

- Where lets you pass filter criteria to server

```
// instead of this
clientContext.Load(site.Lists);

// use this instead
clientContext.Load(site.Lists, lists => lists.Where(list => !list.Hidden));
```

- Include lets you pick fields on item in a collection

```
// indicate which list properties you want to populate for each list
clientContext.Load(site.Lists,
    lists => lists.Include(list => list.Title, list => list.DefaultViewUrl));
```

- Syntax is powerful but tricky to read and write

```
ListCollection Lists = clientContext.Web.Lists;
clientContext.Load(Lists, lists => lists.Where(list => !list.Hidden)
    .Include(list => list.Title,
        list => list.DefaultViewUrl));

clientContext.ExecuteQuery();
```



Check Whether List Exists

- How do you determine if a list already exists
 - CSOM doesn't provide simple approach
 - Query for the list by it's title or URL
 - Check to see if match list exists

```
web site = clientContext.Web;
ListCollection listCollection = clientContext.Web.Lists;
clientContext.Load(listCollection,
    lists => lists.Include(list => list.Title)
                    .where(list => list.Title == "Customers"));

clientContext.ExecuteQuery();

bool listDoesNotExist = (listCollection.Count == 0);

if (listDoesNotExist) {
    // create customers list if it does not exist
}
```



Retrieving Data using LoadQuery

- LoadQuery can be used instead of Load
 - Allows you to write LINQ query expressions

```
ClientContext clientContext = new ClientContext("http://intranet.wingtip.com");  
ListCollection Lists = clientContext.Web.Lists;  
  
// retrieve standard lists that are not hidden  
IQueryable<List> query = from list in Lists  
                        where (list.Hidden == false) && (list.BaseType == 0)  
                        select list;  
  
var queryResults = clientContext.LoadQuery(query);  
clientContext.ExecuteQuery();  
  
foreach (List list in queryResults) {  
    Console.WriteLine(list.Title);  
}
```



Retrieving with a CamlQuery

```
ClientContext clientContext = new ClientContext("http://intranet.wingtip.com");

List list = clientContext.Web.Lists.GetByTitle("Customers");
CamlQuery query = new CamlQuery();
query.ViewXml =
    @"<View>
      <Query>
        <Where>
          <BeginsWith>
            <FieldRef Name='FirstName' />
            <Value Type='Text'>B</Value>
          </BeginsWith>
        </Where>
        <OrderBy>
          <FieldRef Name='Title' />
        </OrderBy>
      </Query>
      <ViewFields>
        <FieldRef Name='FirstName' />
        <FieldRef Name='Title' />
        <FieldRef Name='WorkPhone' />
      </ViewFields>
    </View>";

ListItemCollection queryResults = list.GetItems(query);
clientContext.Load(queryResults);
clientContext.ExecuteQuery();

foreach (ListItem item in queryResults) {
    Console.WriteLine(item["Title"] + ", " + item["FirstName"] + " - " + item["WorkPhone"]);
}
```

Batching Commands

```
private void CreateCustomers(ClientContext clientContext, int customerCount, int batchSize) {
    List list = clientContext.Web.Lists.GetByTitle("Customers");

    int batchCount = 0;
    foreach (var customer in CustomerFactory.GetCustomerList(customerCount, false)) {
        batchCount += 1;
        var lici = new ListItemCreationInformation();
        ListItem item = list.AddItem(new ListItemCreationInformation());
        item["FirstName"] = customer.FirstName; item["Title"] = customer.LastName;
        item["Company"] = customer.Company; item["WorkPhone"] = customer.WorkPhone;
        item["HomePhone"] = customer.HomePhone; item["Email"] = customer.EmailAddress;
        item.Update();
        // call ExecuteQuery only when reaching batch size
        if (batchCount == batchSize) {
            clientContext.ExecuteQuery();
            batchCount = 0;
        }
    }
    // make sure all items have been committed
    if (batchCount > 0) {
        clientContext.ExecuteQuery();
    }
}
```



Agenda

- ✓ CSOM Fundamentals
- ✓ User and App Authentication
- ✓ CSOM Code Optimization
- Remote Exception Handling
 - Creating Content Types and Lists
 - Managed Metadata and Publishing



Consider the following code...

```
clientContext clientContext =  
    new ClientContext("http://intranet.wingtip.com");  
  
clientContext.Web.Lists.GetByTitle("List1").DeleteObject();  
clientContext.Web.Lists.GetByTitle("List2").DeleteObject();  
  
try {  
    clientContext.ExecuteQuery();  
}  
catch (ServerException ex) {  
    Console.WriteLine(ex.GetType().ToString());  
    Console.WriteLine(ex.Message);  
    Console.WriteLine(ex.ServerErrorCode);  
    Console.WriteLine(ex.ServerErrorTraceCorrelationId);  
}
```



Remote Exception Handling

```
ClientContext clientContext =  
    new ClientContext("http://intranet.wingtip.com");  
  
ExceptionHandlingScope scope = new ExceptionHandlingScope(clientContext);  
  
using (scope.StartScope()) {  
    using (scope.StartTry()) {  
        // perform operations  
    }  
    using (scope.StartCatch()) {  
        // handle error  
    }  
    using (scope.StartFinally()) {  
        // add cleanup code  
    }  
}  
  
// execute batch with remote exception handling  
clientContext.ExecuteQuery();
```



General Usage

```
clientContext clientContext =
    new ClientContext("http://intranet.wingtip.com");

// attempt first operation
ExceptionHandlingScope scope1 = new ExceptionHandlingScope(clientContext);
using (scope1.StartScope()) {
    using (scope1.StartTry()) {
        clientContext.Web.Lists.GetByTitle("List1").DeleteObject();
    }
    using (scope1.StartCatch()) { /* do nothing */ }
}

// attempt second operation
ExceptionHandlingScope scope2 = new ExceptionHandlingScope(clientContext);
using (scope2.StartScope()) {
    using (scope2.StartTry()) {
        clientContext.Web.Lists.GetByTitle("List2").DeleteObject();
    }
    using (scope2.StartCatch()) { /* do nothing */ }
}

// execute batch with remote exception handling
clientContext.ExecuteQuery();
```



Agenda

- ✓ CSOM Fundamentals
- ✓ User and App Authentication
- ✓ CSOM Code Optimization
- ✓ Remote Exception Handling
- Creating Content Types and Lists



Creating a List

```
Web site = clientContext.Web;
clientContext.Load(site);

// create and initialize ListCreationInformation object
ListCreationInformation listInformation = new ListCreationInformation();
listInformation.Title = "Announcements";
listInformation.Url = "Lists/Announcements";
listInformation.QuickLaunchOption = QuickLaunchOptions.On;
listInformation.TemplateType = (int)ListTemplateType.Announcements;

// Add ListCreationInformation to lists collection and return list object
List list = site.Lists.Add(listInformation);

// modify additional list properties and update
list.OnQuickLaunch = true;
list.EnableAttachments = false;
list.Update();

// send command to server to create list
clientContext.ExecuteQuery();
```



Checking Whether List Already Exists

```
Web site = clientContext.Web;
clientContext.Load(site);

string listTitle = "Announcements";

// delete list if it exists
ExceptionHandlingScope scope = new ExceptionHandlingScope(clientContext);
using (scope.StartScope()) {
    using (scope.StartTry()) {
        site.Lists.GetByTitle(listTitle).DeleteObject();
    }
    using (scope.StartCatch()) { }
}

// create and initialize ListCreationInformation object
ListCreationInformation listInformation = new ListCreationInformation();
listInformation.Title = listTitle;
listInformation.Url = "Lists/Announcements";
listInformation.QuickLaunchOption = QuickLaunchOptions.On;
listInformation.TemplateType = (int)ListTemplateType.Announcements;

// Add ListCreationInformation to lists collection and return list object
List list = site.Lists.Add(listInformation);

// modify additional list properties and update
list.OnQuickLaunch = true;
list.EnableAttachments = false;
list.Update();

// send command to server to create list
clientContext.ExecuteQuery();
```



Creating List Items

```
ListItemCreationInformation lici = new ListItemCreationInformation();

var item1 = list.AddItem(lici);
item1["Title"] = "SharePoint introduces new app model";
item1["Body"] = "<div>Developers wonder what happened to solutions.</div>";
item1["Expires"] = DateTime.Today.AddYears(10);
item1.Update();

var item2 = list.AddItem(lici);
item2["Title"] = "All SharePoint developers must now learn JavaScript";
item2["Body"] = "<div>Some developers are more excited then others.</div>";
item2["Expires"] = DateTime.Today.AddYears(1);
item2.Update();

var item3 = list.AddItem(lici);
item3["Title"] = "CSOM programming is super fun";
item3["Body"] = "<div>Just ask my mom.</div>";
item3["Expires"] = DateTime.Today.AddDays(7);
item3.Update();

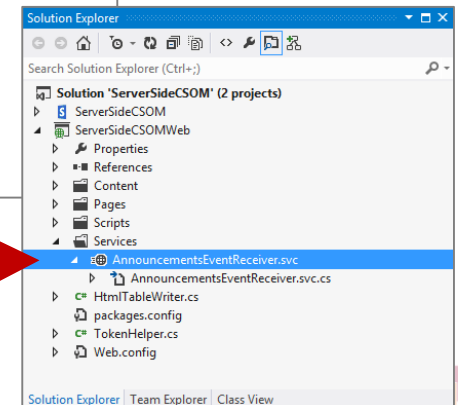
clientContext.ExecuteQuery();
```



Registering Remote Event Receivers

- Remote Event Receivers require registration
 - Declarative registration only possible in app web
 - Registration in host web requires CSOM code

```
string urlEventReceiver = Request.Url.GetLeftPart(UriPartial.Authority) +  
    @"/Services/AnnouncementsEventReceiver.svc";  
  
EventReceiverDefinitionCreationInformation erci1 = new EventReceiverDefinitionCreationInformation();  
erci1.ReceiverName = "ItemAdding";  
erci1.EventType = EventReceiverType.ItemAdding;  
erci1.ReceiverUrl = urlEventReceiver;  
erci1.SequenceNumber = 1000;  
EventReceiverDefinition er1 = list.EventReceivers.Add(erci1);  
er1.Update();  
  
EventReceiverDefinitionCreationInformation erci2 = new EventReceiverDefinitionCreationInformation();  
erci2.ReceiverName = "ItemUpdating";  
erci2.EventType = EventReceiverType.ItemUpdating;  
erci2.ReceiverUrl = urlEventReceiver;  
erci2.SequenceNumber = 1000;  
EventReceiverDefinition er2 = list.EventReceivers.Add(erci2);  
er2.Update();  
  
clientContext.ExecuteQuery();
```



Creating Site Columns - Part 1

```
static Field CreateSiteColumn(string fieldName, string fieldDisplayName, string fieldType) {  
    Console.WriteLine("Creating " + fieldName + " site column...");  
  
    // delete existing field if it exists  
    try {  
        Field fld = site.Fields.GetByInternalNameOrTitle(fieldName);  
        fld.DeleteObject();  
        clientContext.ExecuteQuery();  
    }  
    catch { }  
  
    string fieldXML = @"<Field Name='" + fieldName + "' " +  
        "DisplayName='" + fieldDisplayName + "' " +  
        "Type='" + fieldType + "' " +  
        "Group='wingtip' > " +  
        "</Field>";  
  
    Field field = site.Fields.AddFieldAsXml(fieldXML, true, AddFieldOptions.DefaultValue);  
    clientContext.Load(field);  
    clientContext.ExecuteQuery();  
    return field;  
}
```



Creating Site Columns - Part 2

```
fieldProductCode = CreateSiteColumn("ProductCode", "Product Code", "Text");
fieldProductCode.EnforceUniqueValues = true;
fieldProductCode.Indexed = true;
fieldProductCode.Required = true;
fieldProductCode.Update();
clientContext.ExecuteQuery();
clientContext.Load(fieldProductCode);
clientContext.ExecuteQuery();

fieldProductDescription =
    clientContext.CastTo<FieldMultiLineText>(CreateSiteColumn("ProductDescription", "Product Description", "Note"));
fieldProductDescription.NumberOfLines = 4;
fieldProductDescription.RichText = false;
fieldProductDescription.Update();
clientContext.ExecuteQuery();

fieldProductListPrice =
    clientContext.CastTo<FieldCurrency>(CreateSiteColumn("ProductListPrice", "List Price", "Currency"));
fieldProductListPrice.MinimumValue = 0;
fieldProductListPrice.Update();
clientContext.ExecuteQuery();

fieldProductCategory =
    clientContext.CastTo<TaxonomyField>(CreateSiteColumn("ProductCategory", "Product Category", "TaxonomyFieldType"));
fieldProductCategory.SspId = localTermStoreId;
fieldProductCategory.TermSetId = termSetId;
fieldProductCategory.AllowMultipleValues = false;
fieldProductCategory.Update();
clientContext.ExecuteQuery();

fieldProductColor =
    clientContext.CastTo<FieldMultiChoice>(CreateSiteColumn("ProductColor", "Product Color", "MultiChoice"));
string[] choicesProductColor = { "White", "Black", "Grey", "Blue", "Red", "Green", "Yellow" };
fieldProductColor.Choices = choicesProductColor;
fieldProductColor.Update();
clientContext.ExecuteQuery();
```



Creating Content Types - Part 1

```
static ContentType CreateContentType(string contentTypeName, string baseContentType) {  
    DeleteContentType(contentTypeName);  
  
    ContentTypeCreationInformation contentTypeCreateInfo = new ContentTypeCreationInformation();  
    contentTypeCreateInfo.Name = contentTypeName;  
    contentTypeCreateInfo.ParentContentType = site.ContentTypes.GetById(baseContentType); ;  
    contentTypeCreateInfo.Group = "wingtip";  
    ContentType ctype = site.ContentTypes.Add(contentTypeCreateInfo);  
    clientContext.ExecuteQuery();  
    return ctype;  
}  
  
static void DeleteContentType(string contentTypeName) {  
    try {  
        foreach (var ct in site.ContentTypes) {  
            if (ct.Name.Equals(contentTypeName)) {  
                ct.DeleteObject();  
                Console.WriteLine("Deleting existing " + ct.Name + " content type...");  
                clientContext.ExecuteQuery();  
                break;  
            }  
        }  
    }  
    catch { }  
}
```



Creating Content Types - Part 2

```
ctypeProduct = CreateContentType("Product", "0x01");

// add site columns
FieldLinkCreationInformation fieldLinkProductCode = new FieldLinkCreationInformation();
fieldLinkProductCode.Field = fieldProductCode;
ctypeProduct.FieldLinks.Add(fieldLinkProductCode);
ctypeProduct.Update(true);

FieldLinkCreationInformation fieldLinkProductDescription = new FieldLinkCreationInformation();
fieldLinkProductDescription.Field = fieldProductDescription;
ctypeProduct.FieldLinks.Add(fieldLinkProductDescription);
ctypeProduct.Update(true);

FieldLinkCreationInformation fieldLinkProductListPrice = new FieldLinkCreationInformation();
fieldLinkProductListPrice.Field = fieldProductListPrice;
ctypeProduct.FieldLinks.Add(fieldLinkProductListPrice);
ctypeProduct.Update(true);

FieldLinkCreationInformation fieldLinkProductCategory = new FieldLinkCreationInformation();
fieldLinkProductCategory.Field = fieldProductCategory;
ctypeProduct.FieldLinks.Add(fieldLinkProductCategory);
ctypeProduct.Update(true);

FieldLinkCreationInformation fieldLinkProductColor = new FieldLinkCreationInformation();
fieldLinkProductColor.Field = fieldProductColor;
ctypeProduct.FieldLinks.Add(fieldLinkProductColor);
ctypeProduct.Update(true);

clientContext.ExecuteQuery();
```



Creating List with Content Type

```
ListCreationInformation listInformationProducts = new ListCreationInformation();
listInformationProducts.Title = "Products";
listInformationProducts.Url = "Lists/Products";
listInformationProducts.QuickLaunchOption = QuickLaunchOptions.On;
listInformationProducts.TemplateType = (int)ListTemplateType.GenericList;
listProducts = site.Lists.Add(listInformationProducts);
listProducts.OnQuickLaunch = true;
listProducts.Update();

clientContext.Load(listProducts);
clientContext.Load(listProducts.ContentTypes);
clientContext.ExecuteQuery();

// configure list to use custom content type
listProducts.ContentTypesEnabled = true;
listProducts.ContentTypes.AddExistingContentType(ctypeProduct);
ContentType existing = listProducts.ContentTypes[0]; ;
existing.DeleteObject();
listProducts.Update();
clientContext.ExecuteQuery();

// add custom site columns to default view of list
View viewProducts = listProducts.DefaultView;
viewProducts.ViewFields.Add("ProductCode");
viewProducts.ViewFields.Add("ProductListPrice");
viewProducts.ViewFields.Add("ProductCategory");
viewProducts.ViewFields.Add("ProductColor");
viewProducts.Update();

clientContext.ExecuteQuery();
```



Creating a Document Library

```
ListCreationInformation listInformationProductImages = new ListCreationInformation();
listInformationProductImages.Title = "Product Images";
// make sure to set URL to root of site - not in /Lists folder
listInformationProductImages.Url = "ProductImages";
listInformationProductImages.QuickLaunchOption = QuickLaunchOptions.On;
listInformationProductImages.TemplateType = (int)ListTemplateType.PictureLibrary;
listProductImages = site.Lists.Add(listInformationProductImages);
listProductImages.OnQuickLaunch = true;
listProductImages.Update();

clientContext.ExecuteQuery();
```



Uploading Files to a Library

- Create a utility upload function with common CSOM code

```
static void UploadProductImage(byte[] imageContent, string imageFileName) {  
    Console.WriteLine("  uploading " + imageFileName);  
    FileCreationInformation fileInfo = new FileCreationInformation();  
    fileInfo.Content = imageContent;  
    fileInfo.Overwrite = true;  
    fileInfo.Url = listProductImagesUrl + imageFileName;  
    File newFile = listProductImages.RootFolder.Files.Add(fileInfo);  
    clientContext.ExecuteQuery();  
}
```

- Call function passing file name and byte array

```
UploadProductImage(Properties.Resources.WP0001, "WP0001.jpg");  
UploadProductImage(Properties.Resources.WP0002, "WP0002.jpg");  
UploadProductImage(Properties.Resources.WP0003, "WP0003.jpg");  
UploadProductImage(Properties.Resources.WP0004, "WP0004.jpg");  
UploadProductImage(Properties.Resources.WP0005, "WP0005.jpg");  
UploadProductImage(Properties.Resources.WP0006, "WP0006.jpg");  
UploadProductImage(Properties.Resources.WP0007, "WP0007.jpg");  
UploadProductImage(Properties.Resources.WP0008, "WP0008.jpg");  
UploadProductImage(Properties.Resources.WP0009, "WP0009.jpg");  
UploadProductImage(Properties.Resources.WP0010, "WP0010.jpg");
```



Summary

- ✓ CSOM Fundamentals
- ✓ User and App Authentication
- ✓ CSOM Code Optimization
- ✓ Remote Exception Handling
- ✓ Creating Content Types and Lists

