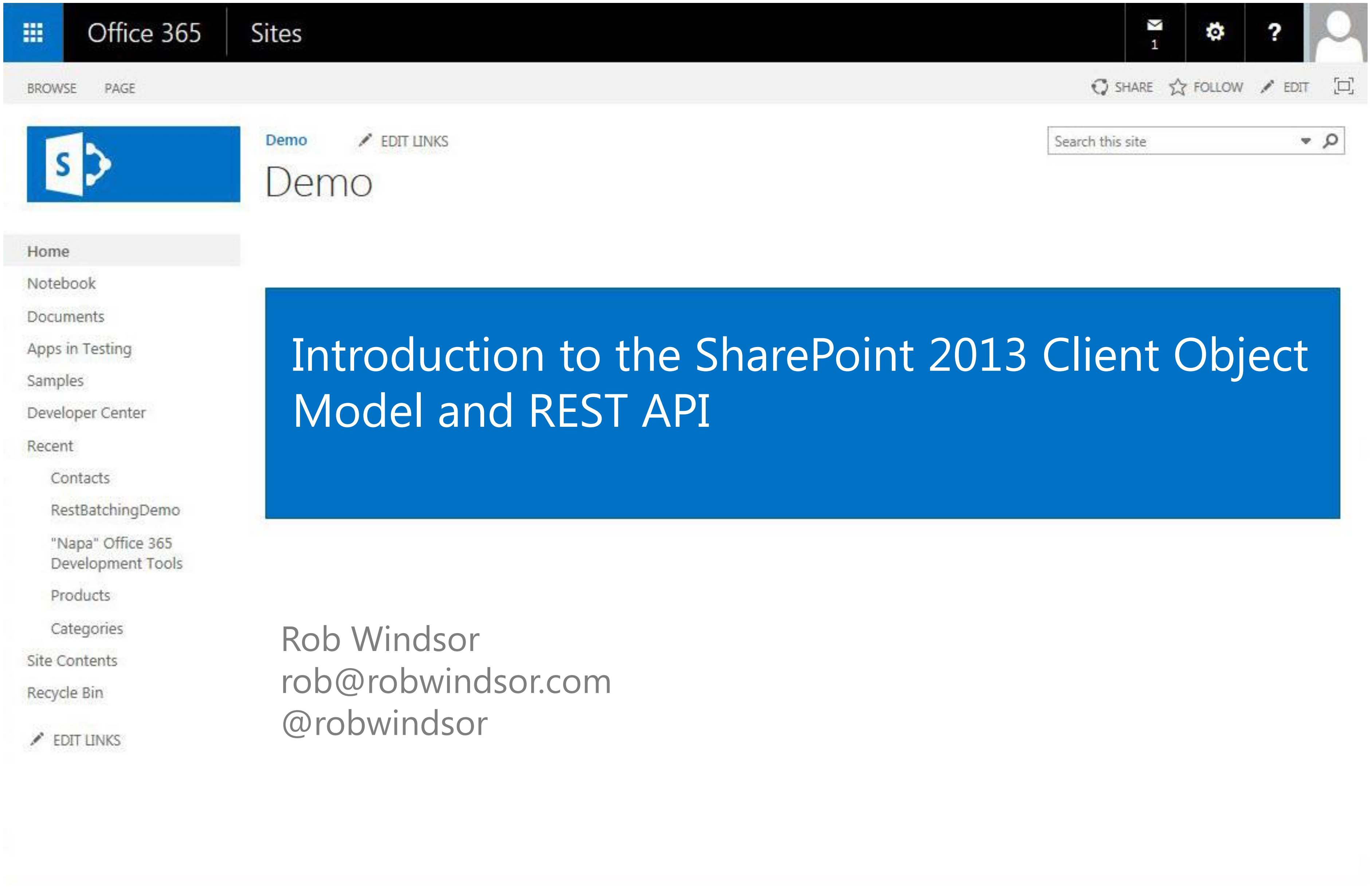
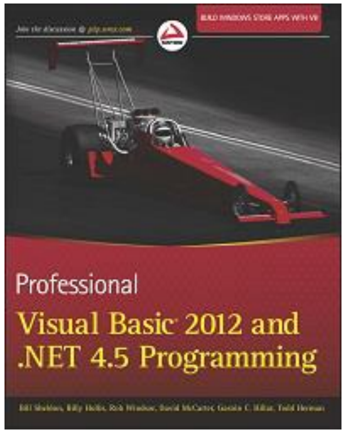


Introduction to the SharePoint 2013 Client Object Model and REST API

Rob Windsor
rob@robwindsor.com
@robwindsor

About Me

- Senior SharePoint Consultant
- Technical Contributor to the Pluralsight On-Demand Library
- Microsoft MVP, MCPD, MCT
- Founder and Past-President of the North Toronto .NET UG
- Co-author of Prof. Visual Basic 2012 and .NET 4.5 (Wrox)


Professional
Visual Basic 2012 and .NET 4.5 Programming

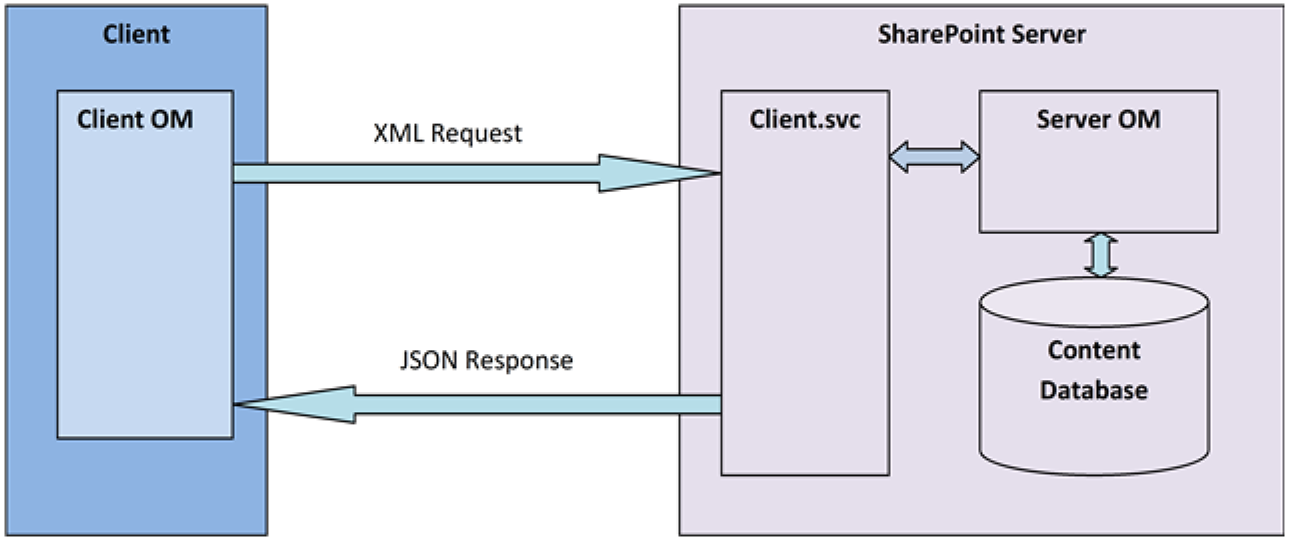
About the APIs

Client Object Model (CSOM)

- **API used when building remote applications**
  - Designed to be similar to the Server Object Model
  - Introduced in SharePoint 2010, expanded in SharePoint 2013
  - Slightly different versions for SP 2013 on-premises and SP Online
- **Three implementations**
  - .NET Managed, Silverlight (plus Mobile), JavaScript
  - Façades on top of /_vti_bin/Client.svc
- **Managed implementation has two versions**
  - Version 15 is for use against an on-premises farm
  - Version 16 is for use against SharePoint Online
- **Communication with SharePoint done in batches**

Client Object Model Batching

- **All CRUD operations are automatically batched**
- **Requests for resources batched using Load and LoadQuery methods**
- **Batches are executed using ExecuteQuery or ExecuteQueryAsync**
    - This triggers a POST request to Client.svc/ProcessQuery
    - Message body contains XML document with batched request information
    - Response contains requested resources in JSON format



Client Object Model Coverage

- Sites, Webs, Features, Event Receivers
- Lists, List Items, Fields, Content Types, Views, Forms
- Files, Folders
- Users, Roles, Groups, User Profiles, Feeds
- Web Parts
- Search
- Taxonomy
- Workflow
- IRM
- E-Discovery
- Analytics
- Business Data

Client Object Model Authentication

- **.NET Managed**
  - Windows credentials passed by default
  - ClientContext.AuthenticationMode
    - Default
    - Anonymous
    - FormsAuthentication
  - ClientContext.Credentials
    - Expects System.Net.ICredentials
    - NetworkCredential, SharePointOnlineCredentials, …
  - ClientContext.FormsAuthenticationLoginInfo
- **Silverlight and JavaScript**
  - Credentials of the hosting Web application are always used

REST API

- **API used when building remote applications**
- **What is the REST API in SharePoint**
  - Data-centric web services **based on** the Open Data Protocol (OData)
  - Each resource or set of resources is addressable
    - http://<site url>/_api/web
    - http://<site url>/_api/web/lists
    - http://<site url>/_api/web/lists/getByTitle(‘Customers’)
    - http://<site url>/_api/web/lists/getByTitle(‘Customers’)/items
  - Operations on resources map to HTTP Verbs
    - GET, PUT, POST, DELETE, …
  - Results from service returned in AtomPub (XML) or JavaScript Object Notation (JSON) format

REST API History

- **SharePoint 2010**
  - Initial REST API added
  - /_vti_bin/ListData.svc
  - Exposed CRUD operations on list data
- **SharePoint 2013**
  - REST API expands and evolves
  - ListData.svc deprecated
    - Still available for backwards compatibility
  - RESTful operations added to /_vti_bin/Client.svc
  - /_api added as an alias for /_vti_bin/Client.svc

REST API Coverage

- Sites, Webs, Features, Event Receivers
- Lists, List Items, Fields, Content Types, Views, Forms, IRM
- Files, Folders
- Users, Roles, Groups, User Profiles, Feeds
- Search

- No support for Managed Metadata
  - Term Store or Managed Metadata Fields
- No support for Workflow

Getting Data from SharePoint

CSOM - Retrieving Resources Using Load

- **Indicates object data should be included in next batch retrieval**
- **Not all property values are retrieved**
  - Example: collections of associated objects

Managed:

```
var context = new ClientContext(siteUrl)
var web = context.Web;
context.Load(web);
context.Load(web.Lists);
context.ExecuteQuery();
ResultsListBox.Items.Add(web.Title);
ResultsListBox.Items.Add(web.Lists.Count);
```

JavaScript:

```
var context = SP.ClientContext.get_current();
var web = context.get_web();
var lists = web.get_lists();
context.load(web);
context.load(lists);
context.executeQueryAsync(success, fail);

function success() {
    var div = jQuery("#message");
    div.text(web.get_title());
    div.append("<br />");
    div.append(lists.get_count());
}
```

Retrieving Resources Using LoadQuery (Managed Code)

- **Indicates result of query should be included in next batch retrieval**
- **Query executed on server**
- **Result returned from call**
  - Not loaded in-place as with Load

```
var web = context.Web;

var query = from list in web.Lists
            where list.Hidden == false &&
                    list.ItemCount > 0
            select list;
var lists = context.LoadQuery(query);
context.ExecuteQuery();

Console.WriteLine(lists.Count());
```

Retrieving Resources Using loadQuery (JavaScript)

- No LINQ in JavaScript
- loadQuery very similar to load
    - Returns new object
    - Returns array for collections

load:

```
var context = SP.ClientContext.get_current();
var lists = context.get_web().get_lists();
context.load(lists);
context.executeQueryAsync(success, fail);

function success() {
    var div = jQuery("#message");
    div.text(lists.get_count());
}
```

loadQuery:

```
var context = SP.ClientContext.get_current();
var lists = context.get_web().get_lists();
var myLists = context.loadQuery(lists);
context.executeQueryAsync(success, fail);

function success() {
    var div = jQuery("#message");
    div.text(myLists.length);
}
```

REST – Managed Code Service Proxy

- **Service metadata for /_api was added around April 2013**
  - You can add a service reference in Visual Studio
  - Tooling to add a service reference for SharePoint Online does not work
- **Service proxy contains two context classes**
  - SP.Data.ListData – access to list data
  - SP.ApiData – access to everything else
- **Generated proxy classes do not natively support updates**
  - SharePoint REST API works differently than OData
  - Need to use POST tunneling and client hooks to get updates to work
  - Too much work – better off using CSOM
- **For more detail see white paper by Paul Schaelein**
  - SharePoint 2013 REST and WCF Data Services
  - http://www.schaeflein.net/Pages/SharePoint-2013-REST-and-WCF-Data-Services.aspx
- **Still have the option of using the 2010 version of REST API**

Retrieving Data (Managed)

```
var svcUri = new Uri(siteUrl + "/_api");
var context = new SP2013Proxy.SP.ApiData(svcUri);
context.Credentials = System.Net.CredentialCache.DefaultCredentials;

var resourceUri = new Uri("/web", UriKind.Relative);
var webs = context.Execute<SP2013Proxy.SP.Web>(resourceUri);
var web = webs.First();

ResultsListBox.Items.Add(web.Title);
```

Retrieving Data (Managed)

- Have option of doing HTTP requests
- Need to work with raw XML or JSON

```
var url = siteUrl + "/_api/Web/";
var client = new WebClient();
client.UseDefaultCredentials = true;
client.Headers[HttpRequestHeader.Accept] = "application/json;odata=verbose";
var json = client.DownloadString(url);

var ser = new JavaScriptSerializer();
dynamic item = ser.Deserialize<object>(json);

ResultsListBox.Items.Add(item["d"]["Title"]);
```

Retrieving Data (JavaScript)

- **Use jQuery to make service call**
- **Use _spPageContextInfo to get site URL**
- **Use Accept header to request JSON response**

```
var call = jQuery.ajax({
    url: _spPageContextInfo.webAbsoluteUrl + "/_api/Web/",
    type: "GET",
    dataType: "json",
    headers: {
        Accept: "application/json;odata=verbose"
    }
});
call.done(function (data, textStatus, jqXHR) {
    var div = jQuery("#message");
    div.text(data.d.Title);
});
call.fail(function (jqXHR, textStatus, errorThrown) {
    alert("Call failed. Error: " + errorThrown);
});
```

Reducing Network Traffic

CSOM - Selecting Fields to Retrieve

- Limit fields returned to reduce network traffic
- Use parameter array in Load and LoadQuery
- Use Include for collections

Managed:

```
var web = context.Web;
context.Load(web, w => w.Title, w => w.Description);

var query = from list in web.Lists.Include(l => l.Title)
            where list.Hidden == false &&
                list.ItemCount > 0
            select list;
var lists = context.LoadQuery(query);
context.ExecuteQuery();
```

JavaScript:

```
var web = context.get_web();
var lists = web.get_lists();
context.load(web, "Title", "Description");
context.load(lists, "Include(Title)");
context.executeQueryAsync(success, fail);
```

REST - OData Queries

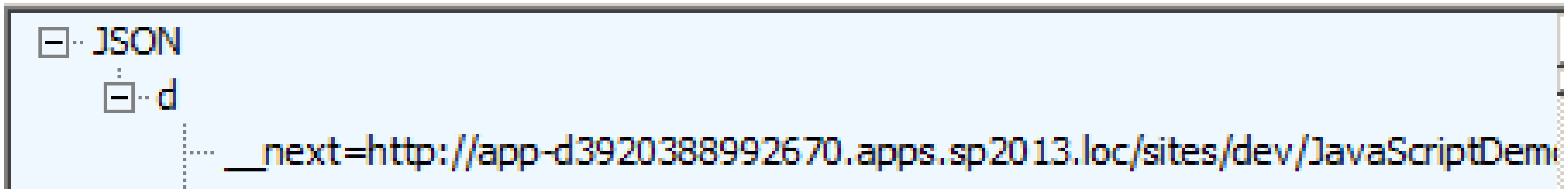
- **Queries represented by query strings added to resource URL**

Option	Example
$select	_api/Web/Lists?$select=Title,ItemCount
$filter	_api/Web/Lists?$filter=(Hidden eq false)
$orderby	_api/Web/Lists?$orderby=ItemCount desc
$skip, $top	_api/Web/Lists?$skip=25&$top=10
$expand	_api/Web/Lists?$expand=Fields

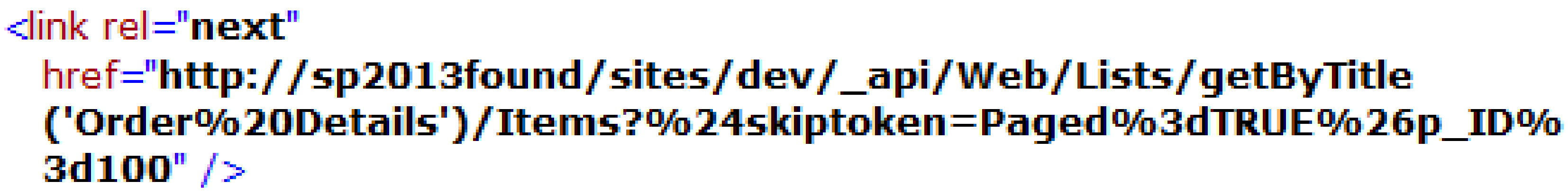
Full documentation: http://www.odata.org/documentation/odata-v2-documentation/uri-conventions/#4_Query_String_Options (http://bit.ly/10dqevp)

REST - OData Continuations

- OData provider may limit number of item in response
- Need to check for __next (JSON) or link element (AtomPub)
- Use URL to get next set of results



```
JSON
  d
    __next=http://app-d3920388992670.apps.sp2013.loc/sites/dev/JavaScriptDem
```



```
<link rel="next"
  href="http://sp2013found/sites/dev/_api/Web/Lists/getByTitle
  ('Order%20Details')/Items?%24skiptoken=Paged%3dTRUE%26p_ID%
  3d100" />
```

CAML Queries

CSOM - Retrieving List Items

- **Somewhat different than Server OM**

Task	Server OM	Managed Client OM
Get list	web.Lists[“Products”]	web.Lists.GetByTitle(“Products”)
Get items	list.Items	list.GetItems(query)
Get item title	item.Title	item[“Title”]
Query type	SPQuery	CamlQuery

- **Set of items accessed by List.GetItems method**
  - Forces use of CAML query to encourage reduced result sets
- **Selecting fields to be returned**
  - Can use ViewFields in query
  - Can use Include with Load or LoadQuery
- **CSOM does not support cross-list CAML queries**
  - Can use KeywordQuery with Search API for similar results

CSOM - Using CAML Queries

Managed:

```
var web = context.Web;
var list = web.Lists.GetByTitle("Products");
var query = new CamlQuery();
query.ViewXml = "<View>" +
                "<Query>" +
                "<Where><Eq>" +
                "<FieldRef Name='Category' " +
                    "LookupId='True' />" +
                "<Value Type='Lookup'>1</Value>" +
                "</Eq></Where>" +
                "</Query>" +
                "</View>";
var items = list.GetItems(query);
context.Load(items,
    c => c.Include(li => li["ID"], li => li["Title"]));
context.ExecuteQuery();
```

JavaScript:

```
var context = SP.ClientContext.get_current();
var web = context.get_web();
var list = web.get_lists().getByTitle("Products");
var query = new SP.CamlQuery();
query.set_viewXml("<View>" +
                  "<Query>" +
                  "<Where><Eq>" +
                  "<FieldRef Name='Category' " +
                      "LookupId='True' />" +
                  "<Value Type='Lookup'>1</Value>" +
                  "</Eq></Where>" +
                  "</Query>" +
                  "<RowLimit>5</RowLimit>" +
                  "</View>");
var items = list.getItems(query);
context.load(web, "Title");
context.load(items, "Include(ID, Title)");
context.executeQueryAsync(success, fail);
```

REST - CAML Queries

- **Must be executed using a POST**
- **Headers must include Form Digest**

```
var viewXml = { ViewXml: "<View>" +
    "<Query>" +
    "<Where><Eq>" +
    "<FieldRef Name='Category' LookupId='True' />" +
    "<Value Type='Lookup'>1</Value>" +
    "</Eq></Where>" +
    "</Query>" +
    "</View>"
}

var call = jQuery.ajax({
    url: _spPageContextInfo.webAbsoluteUrl +
        "/_api/Web/Lists/getByTitle('Products')/GetItems(query=@v1)?" +
        @v1=" + JSON.stringify(viewXml),
    type: "POST",
    dataType: "json",
    headers: {
        Accept: "application/json;odata=verbose",
        "X-RequestDigest": jQuery("#__REQUESTDIGEST").val()
    }
});
```

REST - Form Digest

- **Protects against replay attacks**
- **Value available in hidden field on SharePoint page**
- **Unique to user and site**
- **Only valid for limited time**
- **Use UpdateFormDigest() function to refresh value in hidden field**
  - Service call only make if form digest has expired

- **For more details see blog post by Wictor Wilen**
  - How to refresh the Request Digest value in JavaScript
  - http://www.wictorwilen.se/sharepoint-2013-how-to-refresh-the-request-digest-value-in-javascript

CRUD Operations

CSOM - Creating a List

- **Moderately different than code for Server Object Model**
- **Adding the list**
  - Web.Lists.Add(creationInformation)
  - Parameter is type ListCreationInformation

Managed:

```
var web = context.Web;
var lci = new ListCreationInformation();
lci.Title = "Tasks";
lci.QuickLaunchOption = QuickLaunchOptions.On;
lci.TemplateType = (int)ListTemplateType.Tasks;
var list = web.Lists.Add(lci);
```

JavaScript:

```
var web = context.get_web();
var lci = new SP.ListCreationInformation();
lci.set_title("Tasks");
lci.set_quickLaunchOption(SP.QuickLaunchOptions.on);
lci.set_templateType(SP.ListTemplateType.tasks);
var list = web.get_lists().add(lci);
```

REST - Creating a List (JavaScript)

- **Send POST to /_api/Web/Lists**
- **Message body has SP.List object with properties**
  - Fills same role as SP.ListCreationInformation object in CSOM
- **Must include Form Digest in headers**

```
var call = jQuery.ajax({
    url: _spPageContextInfo.webAbsoluteUrl + "/_api/Web/Lists",
    type: "POST",
    data: JSON.stringify({
        "__metadata": { type: "SP.List" },
        BaseTemplate: SP.ListTemplateType.tasks,
        Title: "Tasks"
    }),
    headers: {
        Accept: "application/json;odata=verbose",
        "Content-Type": "application/json;odata=verbose",
        "X-RequestDigest": jQuery("#__REQUESTDIGEST").val()
    }
});
```

REST – Creating a List (Managed)

- **Must be executed using a POST**
- **Headers must include Form Digest**

```
var digest = GetFormDigest();

var url = siteUrl + "/_api/Web/Lists";
var body = "{'__metadata': { type: 'SP.List' }, " +
    "BaseTemplate: 107, " +
    "Title: 'Tasks2'}";

var client = new WebClient();
client.UseDefaultCredentials = true;
client.Headers[HttpRequestHeader.Accept] = "application/json;odata=verbose";
client.Headers[HttpRequestHeader.ContentType] = "application/json;odata=verbose";
client.Headers["X-RequestDigest"] = digest;
var json = client.UploadString(url, body);

ResultsListBox.Items.Add("List added");
```

REST – Getting the Form Digest (Managed)

- **Make a POST request to /_api/contextinfo**
- **Headers must include Form Digest**

```
var url = siteUrl + "/_api/contextinfo";
var client = new WebClient();
client.UseDefaultCredentials = true;
client.Headers[HttpRequestHeader.Accept] =
    "application/json;odata=verbose";
var json = client.UploadString(url, "");

var ser = new JavaScriptSerializer();
dynamic item = ser.Deserialize<object>(json);

var digest = item["d"]["GetContextWebInformation"]["FormDigestValue"];

return digest;
```

CSOM - Creating and Updating List Items

- **Virtually the same as code for Server Object Model**
- **Adding a list item**
  - List.AddItem(creationInformation)
  - Parameter is type ListItemCreationInformation
- **Updating field values**
  - Exactly the same as Server Object Model code

Managed:

```
var web = context.Web;
var list = web.Lists.GetByTitle("Tasks");

var ici = new ListItemCreationInformation();
var item = list.AddItem(ici);
item["Title"] = "Sample Task";
item["AssignedTo"] = web.CurrentUser;
item["DueDate"] = DateTime.Now.AddDays(7);
item.Update();
```

JavaScript:

```
var web = context.get_web();
var list = web.get_lists().getByTitle("Tasks");

var ici = new SP.ListItemCreationInformation();
var item = list.addItem(ici);
item.set_item("Title", "Sample Task");
item.set_item("AssignedTo", web.get_currentUser());
var due = new Date();
due.setDate(due.getDate() + 7);
item.set_item("DueDate", due);
item.update();
```

REST - Creating List Items (JavaScript)

- **Post to /_api/Web/Lists/getByTitle(‘<List Name>’)/Items**
- **Type name is SP.Data.<List Name>ListItem**

```
var call = jQuery.ajax({
    url: _spPageContextInfo.webAbsoluteUrl + "/_api/Web/Lists/getByTitle('Tasks')/Items",
    type: "POST",
    data: JSON.stringify({
        "__metadata": { type: "SP.Data.TasksListItem" },
        Title: "Sample Task",
        AssignedToId: userId,
        DueDate: due
    }),
    headers: {
        Accept: "application/json;odata=verbose",
        "Content-Type": "application/json;odata=verbose",
        "X-RequestDigest": jQuery("#__REQUESTDIGEST").val()
    }
});
```

REST - Creating List Items (Managed)

```
var digest = GetFormDigest();
var userId = GetCurrentUserId();
var dueDate = DateTime.UtcNow.AddDays(7);
var dueDateString = dueDate.ToString("o");

var url = siteUrl + "/_api/Web/Lists/getByTitle('Tasks2')/Items";
var body = "{'__metadata': { type: 'SP.Data.Tasks2ListItem' }, " +
    "Title: 'Sample Task', " +
    "AssignedToId: " + userId + ", " +
    "DueDate: '" + dueDateString + "'}";

var client = new WebClient();
client.UseDefaultCredentials = true;
client.Headers[HttpRequestHeader.Accept] = "application/json;odata=verbose";
client.Headers[HttpRequestHeader.ContentType] = "application/json;odata=verbose";
client.Headers["X-RequestDigest"] = digest;
var json = client.UploadString(url, body);
```

REST - Creating List Items (Managed Proxy)

- **Create instance of list item type**
  - One of the generated types in the service proxy
- **Set property values**
- **Add to list using the AddTo<list name> method on context**

```
var svcUri = new Uri(siteUrl + "/_vti_bin/ListData.svc");
var context = new SP2010Proxy.DemoDataContext(svcUri);
context.Credentials = System.Net.CredentialCache.DefaultCredentials;

var item = new SP2010Proxy.ProductsItem();
item.Title = "Test Product";
item.ProductID = 999;
item.CategoryId = 1;
item.UnitPrice = 9.99;
item.UnitsInStock = 99;
context.AddToProducts(item);
context.SaveChanges();
```

REST - Updating List Items (JavaScript)

- **Send to /_api/Web/Lists/getByTitle(‘<List>')/Items(<Item Id>)**
- **Request type (X-Http-Method)**
  - Can update by sending PUT
    - All writable field values must be specified
  - Can update by sending POST
    - Set X-Http-Method to PATCH or MERGE
    - Only send field values that are changing
- **Concurrency (IF-MATCH)**
  - Item metadata includes etag which represents the version
  - Set IF-MATCH in header to etag value
    - Update will fail if item has been updated since read
  - SET IF-MATCH in header to *
    - Update will overwrite changes (if any)

REST - Updating List Items (JavaScript)

```
var call = jQuery.ajax({
    url: _spPageContextInfo.webAbsoluteUrl +
        "/_api/Web/Lists/getByTitle('Tasks')/Items(" + item.Id + ")",
    type: "POST",
    data: JSON.stringify({
        "__metadata": { type: "SP.Data.TasksListItem" },
        Status: "In Progress",
        PercentComplete: 0.10
    }),
    headers: {
        Accept: "application/json;odata=verbose",
        "Content-Type": "application/json;odata=verbose",
        "X-RequestDigest": jQuery("#__REQUESTDIGEST").val(),
        "IF-MATCH": item.__metadata.etag,
        "X-Http-Method": "PATCH"
    }
});
```

REST - Updating List Items (Managed)

```
var digest = GetFormDigest();
var itemId = GetListFirstItemId("Tasks2");

var url = siteUrl + "/_api/Web/Lists/getByTitle('Tasks2')/Items(" + itemId + ")";
var body = "{'__metadata': { type: 'SP.Data.Tasks2ListItem' }, " +
    "Status: 'In Progress', " +
    "PercentComplete: 0.10 }";

var client = new WebClient();
client.UseDefaultCredentials = true;
client.Headers[HttpRequestHeader.Accept] = "application/json;odata=verbose";
client.Headers[HttpRequestHeader.ContentType] = "application/json;odata=verbose";
client.Headers["X-RequestDigest"] = digest;
client.Headers["IF-MATCH"] = "*";
client.Headers["X-Http-Method"] = "PATCH";
var json = client.UploadString(url, body);
```

REST - Updating List Items (Managed Proxy)

- **Get list item**
- **Update property values**
- **Call UpdateObject on context**

```
var svcUri = new Uri(siteUrl + "/_vti_bin/ListData.svc");
var context = new SP2010Proxy.DemoDataContext(svcUri);
context.Credentials = System.Net.CredentialCache.DefaultCredentials;

var query = from product in context.Products
            where product.ProductID == 999
            select product;
var item = query.FirstOrDefault();

if (item != null)
{
    item.UnitPrice = 4.44;
    item.UnitsInStock = 44;
    context.UpdateObject(item);
    context.SaveChanges();
}
```

Thank You

- Big thanks to the organizers, sponsors and you for making this event possible
- Please fill out your evaluation
- Please keep in touch

rob@robwindsor.com

@robwindsor

blogs.msmvps.com/windsor