FREE eBook

LEARNING sharepoint

Free unaffiliated eBook created from
Stack Overflow contributors.

#sharepoint

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sharepoint

It is an unofficial and free sharepoint ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sharepoint.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with sharepoint

Remarks

SharePoint can refer to one or more products from the Microsoft SharePoint family.

- **SharePoint Foundation**: This was the underlying technology for all SharePoint sites and is no longer available for SharePoint 2016
- **SharePoint Server**: This is the on-premises version of SharePoint. You can deploy one or more SharePoint servers. It offers additional features over SharePoint Foundation, such as BI capabilities, Enterprise Content Management and more
- **SharePoint Online**: Cloud-based version of SharePoint. The customer does not need to care about the server infrastructure or scalability.

Office 365 is a separate Microsoft offering that includes the SharePoint Online service, although not all plans support all SharePoint features.

The following links provide extensive feature comparisons between available SharePoint versions:

- SharePoint 2013 on-premises vs. SharePoint 2016 on-premises: Feature availability across SharePoint on-premises plans
- SharePoint Features in Office 365: Feature availability across SharePoint plans
- SharePoint Features in SharePoint Online (without Office 365): Feature availability across SharePoint standalone plans
- Merged comparison of features between SharePoint 2013 and SharePoint Online: http://www.buckleyplanet.com/2014/06/sharepoint-online-vs-onprem-feature-comparison.html

Versions

Version	Official Name	Release Date
Pre-2003	SharePoint Portal Server	2002-07-09
2003	SharePoint Portal Server 2003	2003-11-23
2007	SharePoint Server 2007	2007-01-27
2010	Microsoft SharePoint Server 2010	2010-07-15
2013	Microsoft SharePoint Server 2013	2013-01-09
2016	Microsoft SharePoint Server 2016	2016-05-01

Examples

Introduction

SharePoint 2016 is the version 16 release of the SharePoint product family. It was released on May 4, 2016. This example covers the installation of SharePoint 2016 using the Single Server Farm configuration. This configuration covers the basics of setting up a SharePoint farm without the need to have multiple servers. Note that the covered scenarios by a Single Server Farm are usually limited to development and very small production scenarios.

Requirements

Prior to installing SharePoint, the basic environment must be set up. SharePoint stores documents as well as metadata, logs, custom applications, customizations, and much more. Ensure that you have sufficient disk space and RAM available above the base line requirements.

- 4 Cores on a 64-bit compatible processors
- 12 - 24 GB of RAM (depending on test or prod deployment)
- 80GB hard drive for system
- 100GB hard drive as second drive
- Server with 64-bit Windows Server 2012 R2 or Technical Preview “Threshold”
- SQL Server 2014 or SQL Server 2016
- .NET Framework 4.5.2 or .NET Framework 4.6
- Domain joined computer and delegated farm service accounts

All other prerequisites can be installed manually or done using the SharePoint Prerequisite installer included with the SharePoint installation.

Installation

- Run the prerequisites installer; it may request to reboot the server before continuing
- Run Setup.exe from the SharePoint installation
- Enter the license key
- Accept the license agreement
- Select "Complete" on the Server Type Tab
- Setup should complete successfully
- On the complete page, leave the check box checked next to the Run Product Configuration Wizard and click Close

Configuration

If you are continuing from the previous step the SharePoint 2016 Product Configuration Wizard

should open automatically. if the box does not appear or you are running the configuration later, open the configuration wizard by going to Start -> SharePoint 2016 Products -> SharePoint 2016 Product Configuration Wizard.

- Click next on the welcome page
- A modal dialog will pop up saying some services my be restarted during the configuration; nothing has been installed yet, so click yes
- Add the database server for the farm
    - Enter the name of the machine running SQL Server; in this case, it is the local machine
    - Enter the name of the Configuration database or keep the default name SharePoint_Config
    - Enter the username of the domain service user who will be accessing the database (in the form of DOMAIN\user) *Enter the password for the domain user
    - Click next when done
- Enter the farm password; this will be used when joining additional servers to the new farm
- Select the Single Server Farm role
- Configure the Central Admin Web App (where SharePoint will be managed from by the farm administrators) select the port number and select the type of authentication federation (NTLM or Negotate (Kerberose))
- Review the settings on the final pages and make changes as necessary
- When ready, run the configuration which may take a few minutes
- On completion, you will open the wizard will allow you to open the Central Admin site
- On failure, you can investigate the logs in the %COMMONPROGRAMFILES%\Microsoft Shared\Web Server Extensions\16\LOG folder

Farm Configuration

Once the central web app, config database, and central admin are set up, you will be ready to configure the farm for use for users or development. You can bookmark the location of the Central Admin site or access it through a shortcut in the same location as the Product Configuration Wizard.

- If you are starting the configuration later, click on Quick Launch -> Configuration Wizards -> Farm Configuration Wizard
- If you are starting the Wizard from the installation step, click Start the Wizard
- Choose if you want to be part of the customer improvement program by clicking Yes or No
- On the farm configuration page, select the domain account that will run background services on the farm
    - While this account may be the same as the database account, they may also be different for separation of roles and privileges
    - Enter the account as DOMAIN\user
- Validate the services you want available on the farm on the Services page
- Create the first site collection on the farm (this step can be skipped and done at a later time)
    - Enter the site collection's title, description, web address (usually the first site is at the server root), and the template
    - Most things can be changed (title, description) can be changed easily, but others like

the web URL may take much more work to change; the template can also not be easily rolled back, but SharePoint allows a large amount of customizations that allows you to take any base template and convert the style and layout of the site

- When you are complete with the configuration, click finish

The farm and the first site collection are now configured for use.

Build a web part with the SharePoint Framework

dev.office.com/sharepoint is a great place to get started with the SharePoint Framework.

The SharePoint Framework is a modern, client side approach to SharePoint Development initially targeted at SharePoint Online in Office 365. Web parts created with the SharePoint Framework are a new type of web part and they can be made available to add on both existing SharePoint pages and new SharePoint pages.

There's a great hello world example for this process hosted at Build your first SharePoint client-side web part (Hello World part 1). All of the examples at dev.office.com are available for community contributions through github.

The basic steps of Hello World in the SharePoint Framework are:

1. Generate the skeleton of the project with the Yeoman SharePoint Generator.

   yo @microsoft/SharePoint

2. Edit the generated code in the editor of your choice. Support for Visual Studio Code is strong across platforms.

3. Preview the web part using gulp and the local SharePoint Workbench

   gulp serve

4. Preview in your SharePoint Online environment

   Go to the following URL: 'https://your-sharepoint-site/_layouts/workbench.aspx'

SharePoint ULS Logs and Logging

The SharePoint Unified Logging Service (ULS) provides support and debugging capabilities for both ops and developers. Understanding how to read the logs is an important first step to resolving issues.

Tooling

Microsoft provides the ULS Viewer to help read old logs and logs that are currently being written to as the farm is running. It can also filter and apply formatting to logs to help narrow down a problem.

Correlation Identifier

To isolate an issue, it is helpful to only look at a particular correlation id. Each correlation id is associated with a request or end to end action of the system (such as a time jobber). If there is a problem with a web page being rendered, locating the request in the ULS logs and isolating it to the specific correlation id removes all the noise from the other logs, helping to pinpoint the problem.

Adding SPMonitoredScope to My Code

One way to figure add logging and some performance monitoring is to add SPMonitoredScope to your code.

```
using (new SPMonitoredScope("Feature Monitor"))
{
    // My code here
}
```

This code will log the beginning and end of your requests as well as some performance data. Building your own custom monitor that implements ISPScopedPerformanceMonitor allows you to set the trace level or maximum execution time for a set of code.

Read Getting started with sharepoint online: https://riptutorial.com/sharepoint/topic/950/getting-started-with-sharepoint

Chapter 2: Creating a provider hosted App

Examples

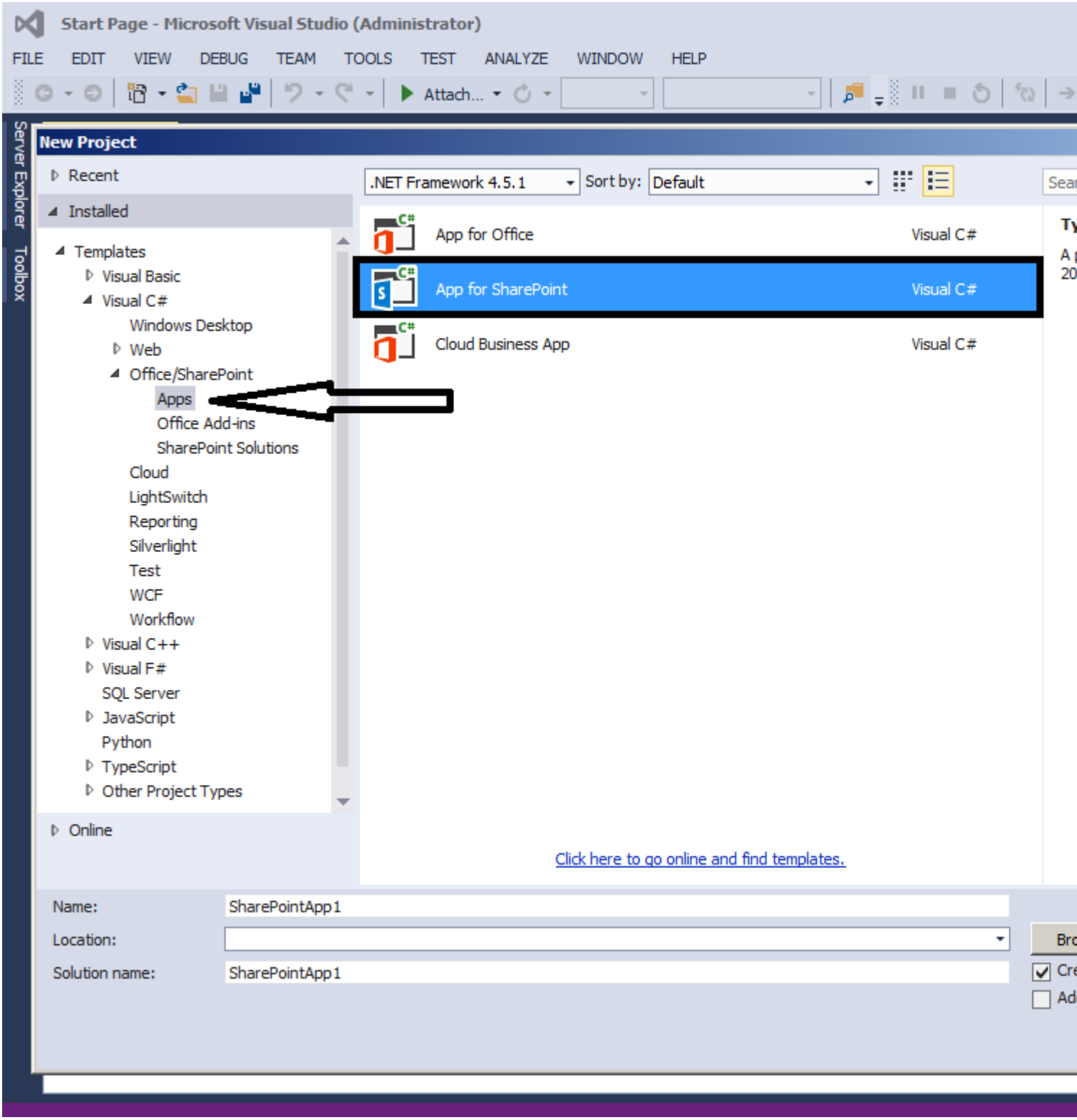
Setting development environment

To start with App Development we need Visual studio 2013 or higher version. Download latest community or expression edition from here > https://www.visualstudio.com/products/free-developer-offers-vs

Once it has been downloaded and installed

Open and **Click create new project**

expand Office/SharePoint section you should see an option for App as shown below.

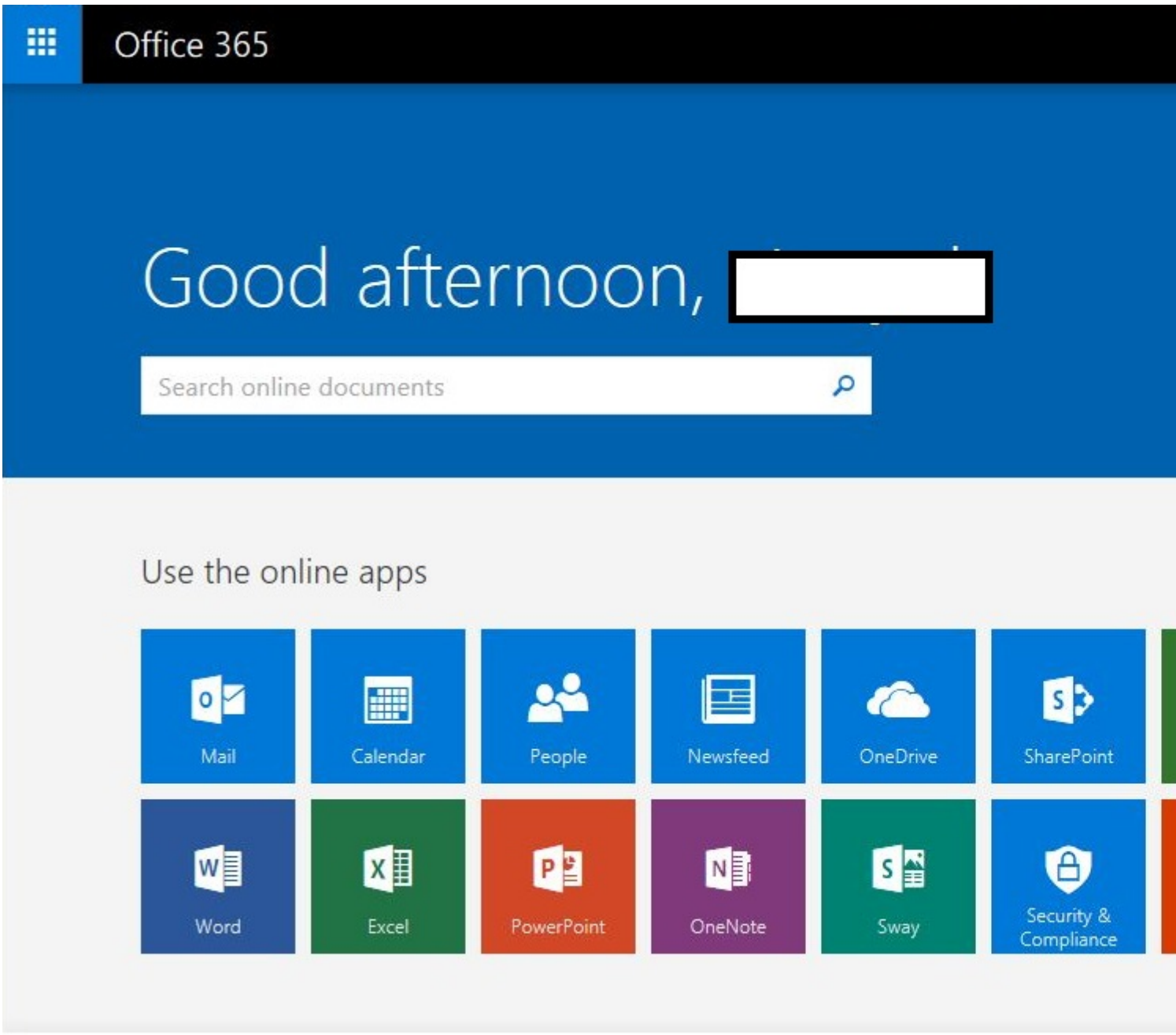


If App option not available Close the VS , download and install **Microsoft Office Developer Tools** https://www.visualstudio.com/en-us/features/office-tools-vs.aspx

Preparing for developer site

Once we have visual studio, we need a developer site to deploy apps to SharePoint. Simplest way is to get is > Sign up for a free, one year Office 365 developer account https://profile.microsoft.com/RegSysProfileCenter/wizardnp.aspx?wizid=14b845d0-938c-45af-b061-f798fbb4d170&lcid=1033

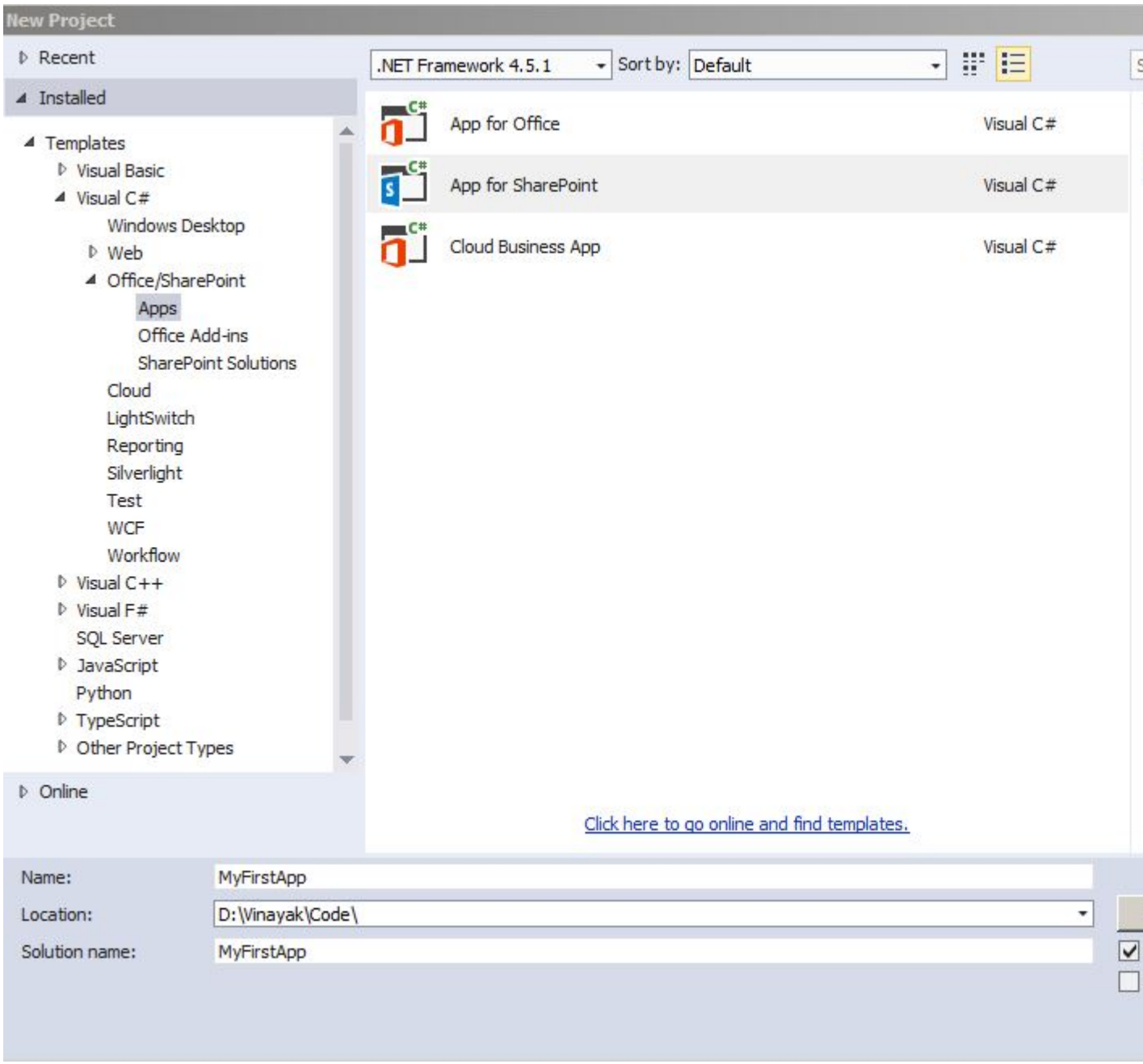
Once sign up process is finished https://www.office.com/ center URL for all your App



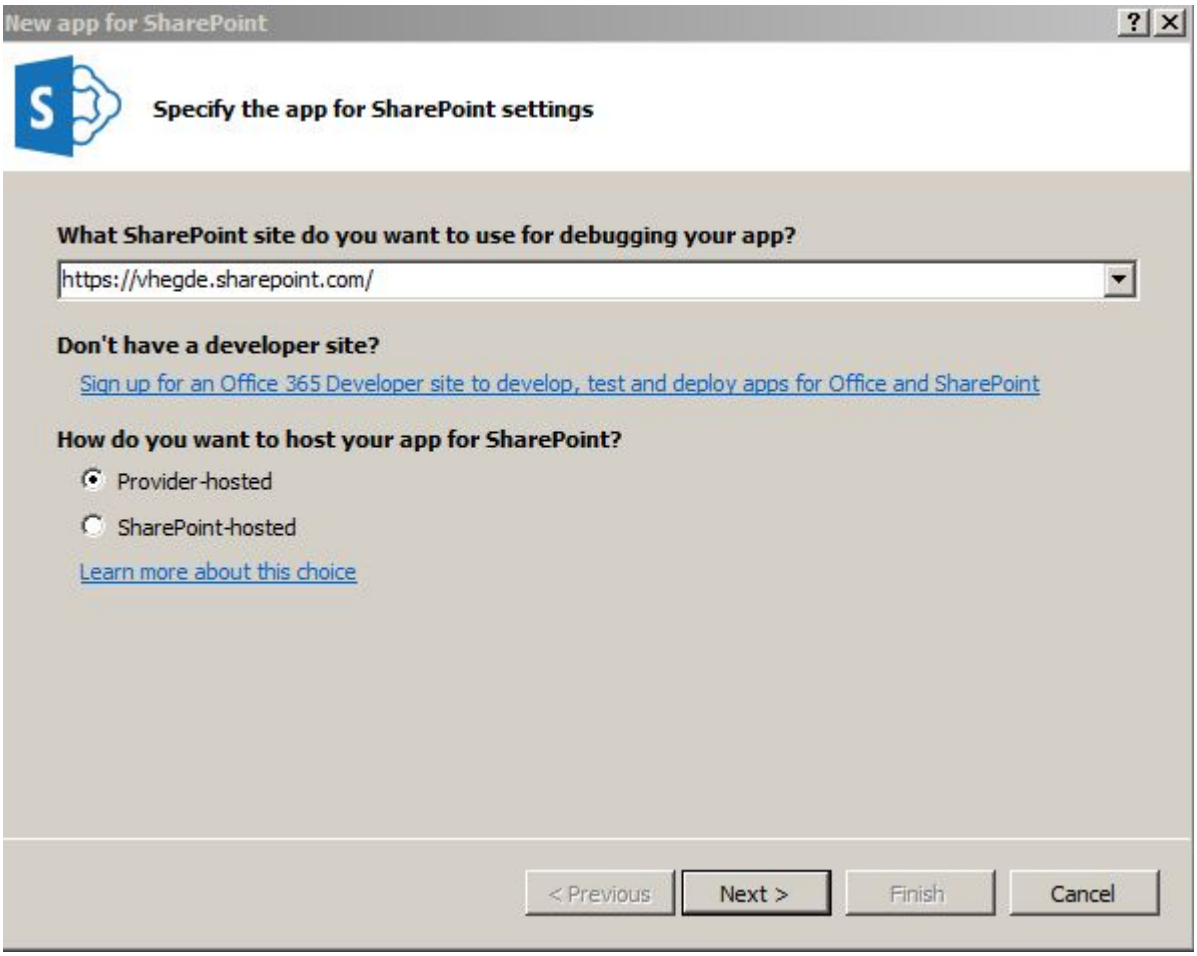
Create App in Visual studio

Lets start with creating our first app

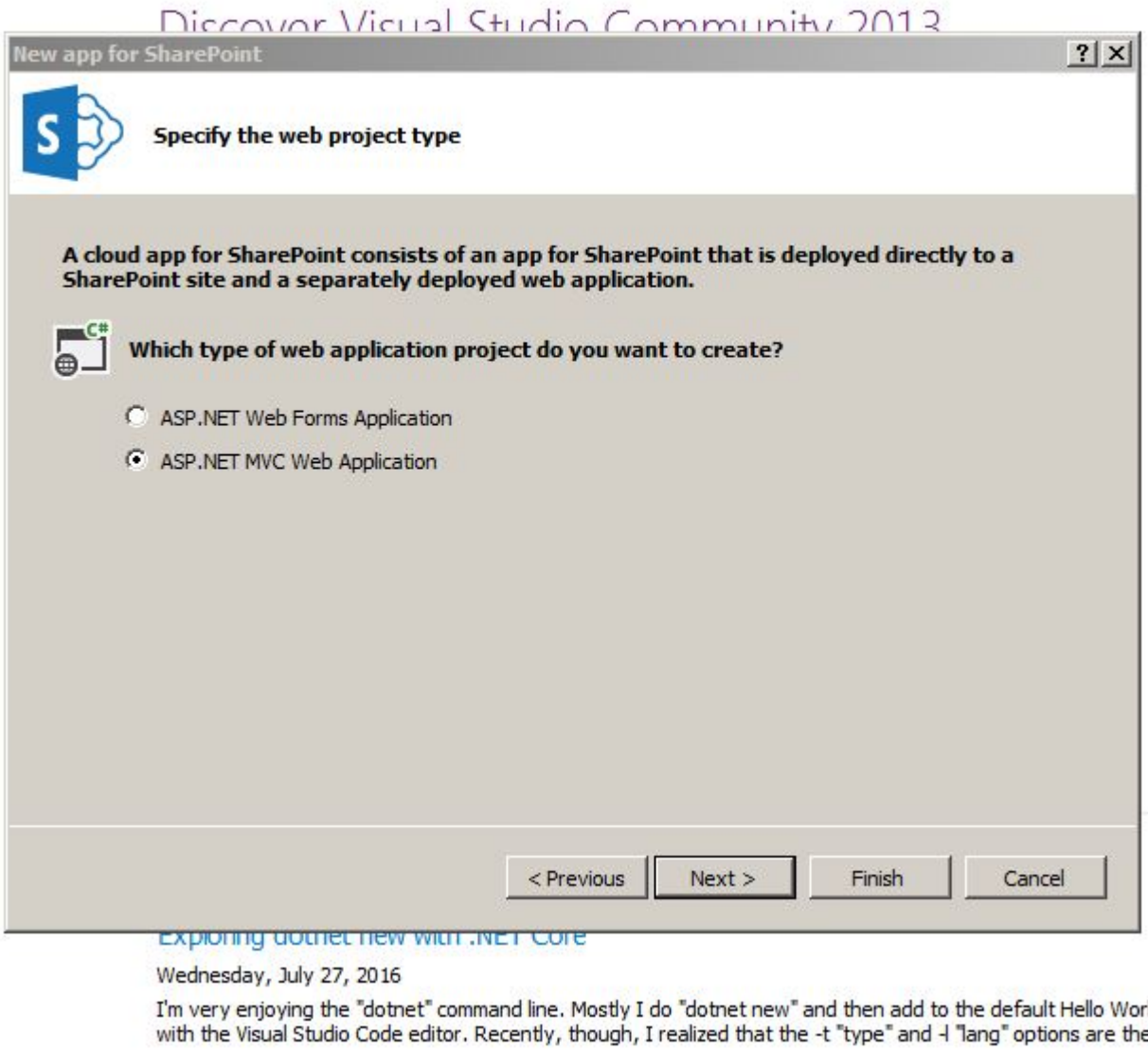
1. Open visual studio and > create new project
2. Enter Name and Location



3. Enter your developer site url created in previous step and select Provider-hosted

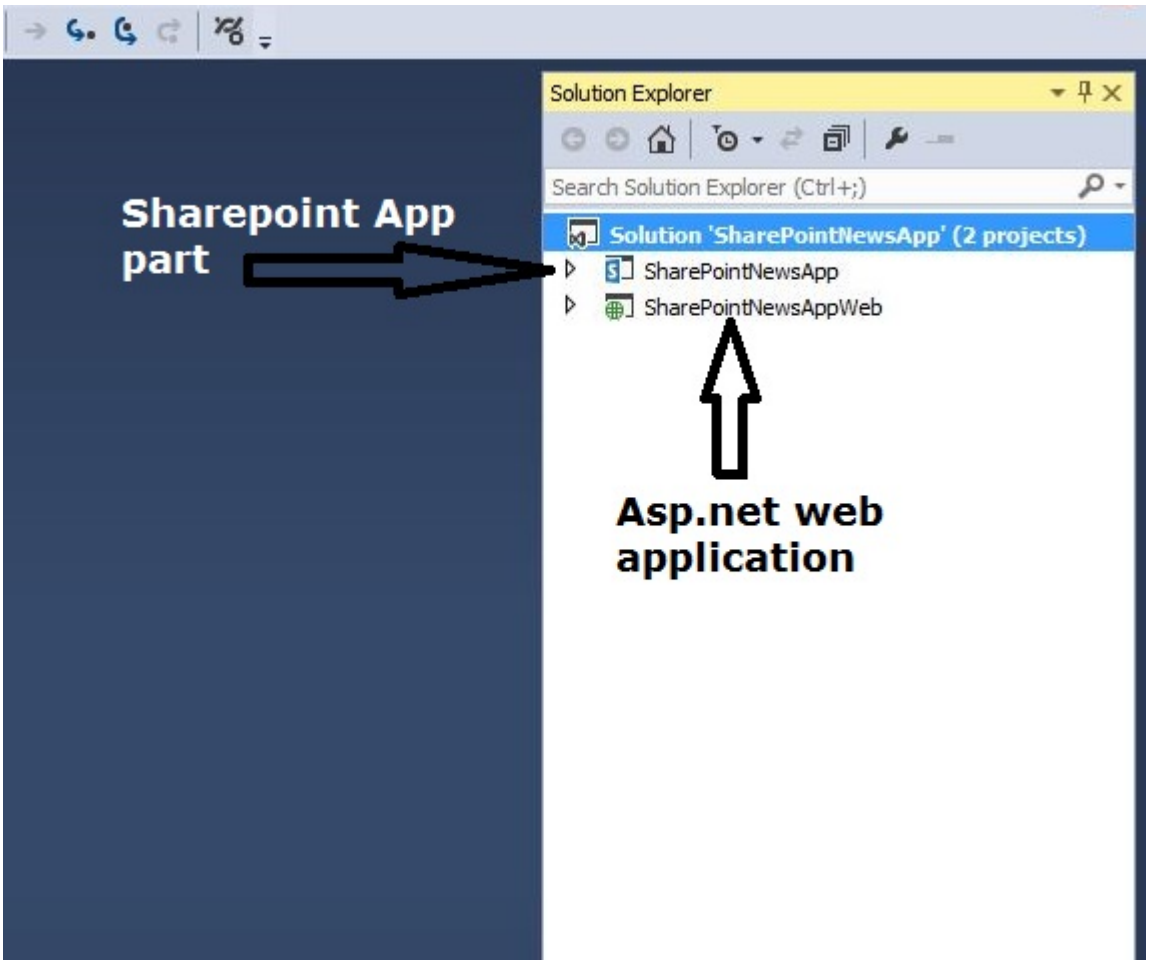


4. Popup will open which will as for login
5. Next step it will as for type of application, either select MVC or Webform. I'm selecting MCV here



6. Under How do you want your add-in to authenticate?, choose Use Windows Azure Access Control Service.and Click Finish

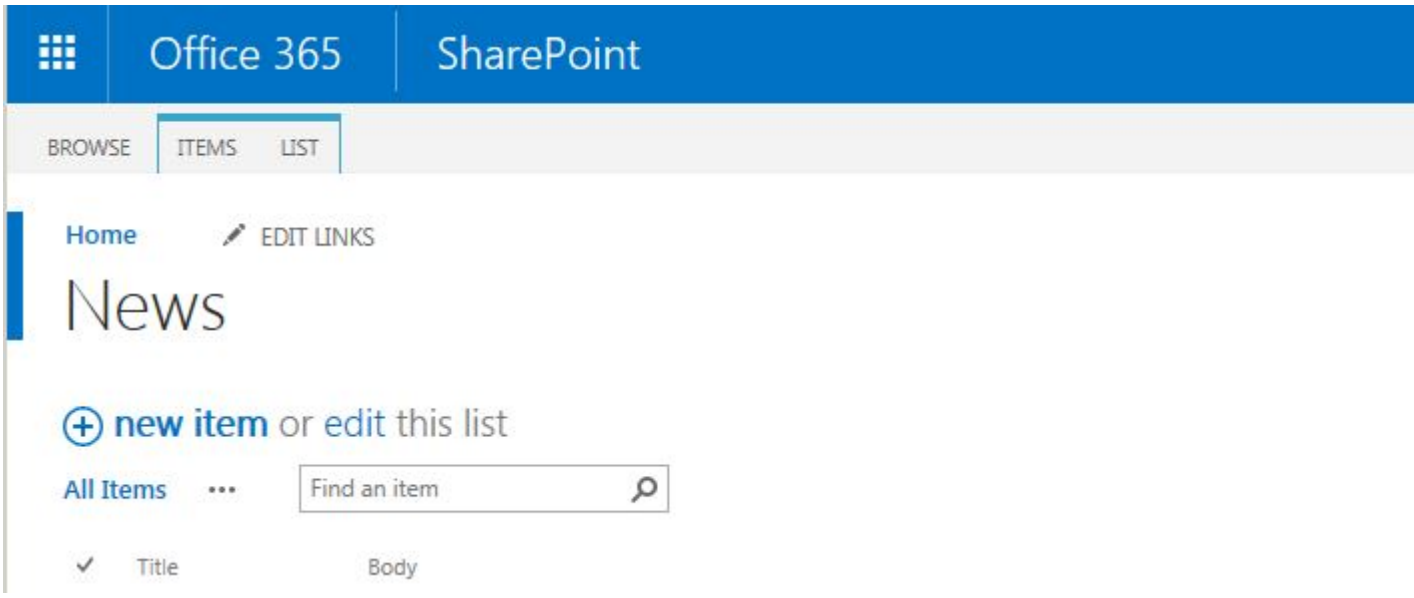
7. In solution explorer we can see 2 project has been created. One is SharePoint app-part and another is asp.net web app



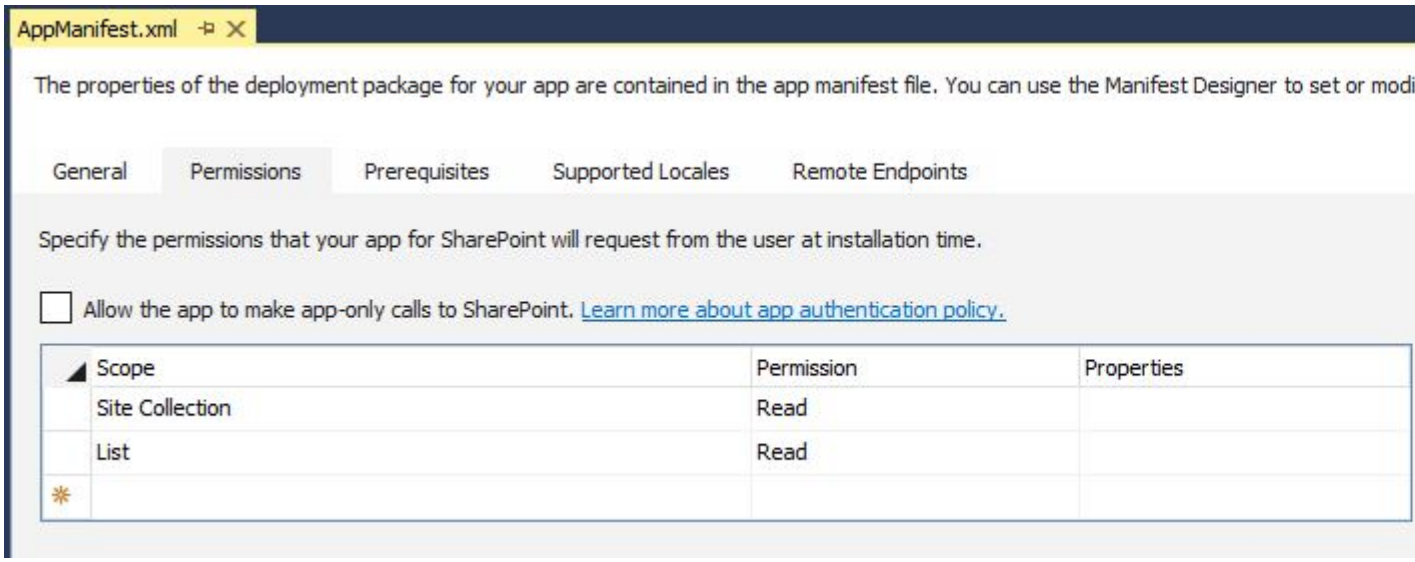
Lets start coding

Here I'm taking the example of a basic news app

1. Open the SharePoint developer site and create a list to store our news articles
2. Create a custom list and Add 3 more columns Body, Summery, ThumbnailImageUrl



3. Go back to our SharePoint app, Open AppManifest.xml file, click on permission Tab and give Read permission to the site collection and save it.



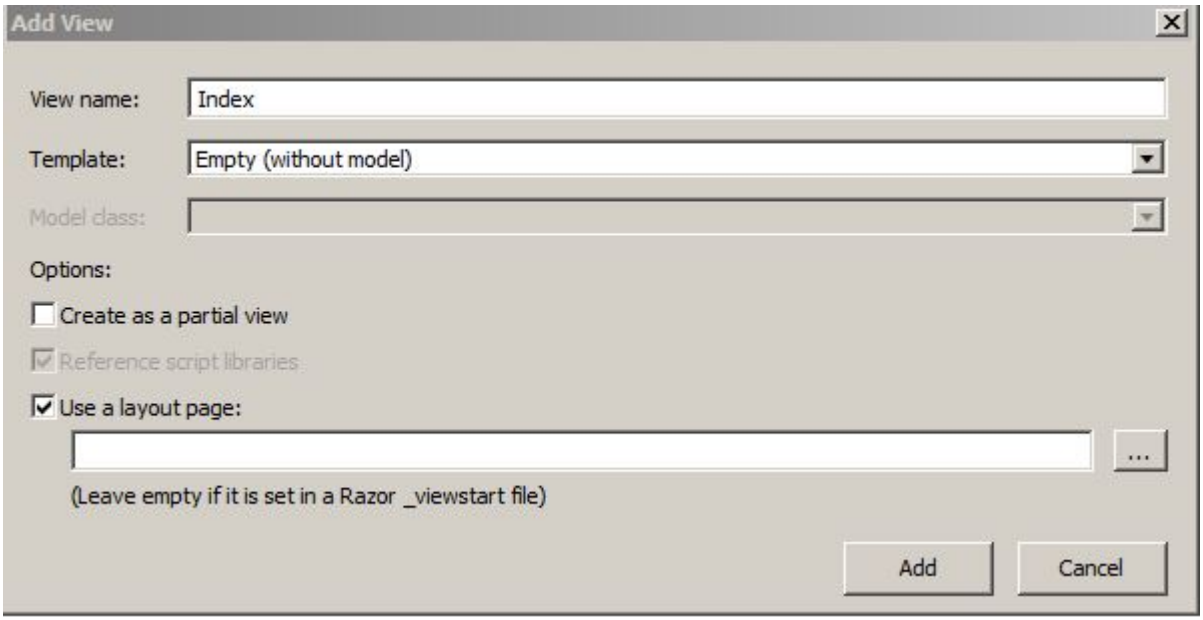
4. Open HomeController from web application, in my case its an MVC application. If you are creating an webform app then you code should be in default.aspx.cs page

5. Below is the code snippet to get latest news from the list. This how our index page should look like.

```
[SharePointContextFilter]
public ActionResult Index()
{
    User spUser = null;

    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);
    List<NewsList> newsList = new List<NewsList>();
    using (var clientContext = spContext.CreateUserClientContextForSPHost())
    {
        if (clientContext != null)
        {
            spUser = clientContext.Web.CurrentUser;

            clientContext.Load(spUser, user => user.Title);

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

            List lst = clientContext.Web.Lists.GetByTitle("News");
            CamlQuery queryNews = CamlQuery.CreateAllItemsQuery(10);
            ListItemCollection newsItems = lst.GetItems(queryNews);
            clientContext.Load(newsItems, includes => includes.Include(i => i.Id, i =>
i.DisplayName, i => i["ThumbnailImageUrl"], i => i["Summery"]));

            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
                foreach (var lstProductItem in newsItems)
                {
                    newsList.Add(
                        new NewsList
                        {
                            Id = Convert.ToInt32(lstProductItem.Id.ToString()),
```

```
                                Title = lstProductItem.DisplayName.ToString(),
                                Summery = lstProductItem["Summery"].ToString(),
                                Thumbnail = lstProductItem["ThumbnailImageUrl"].ToString()
                            });
                    }
                }
            }
        }

        return View(newsList);
}
```

6. Now Right click on **Index** and Click **Add View.** Then click on Add



7. Now open the **Index.cshtml** file From **Views>Home directory**

8. Below is the code snippet for index.cshtml file

```
@model List<SharePointNewsAppWeb.Models.NewsList>
 @{
 ViewBag.Title = "My News - browse latest news";
}
 <br />
 @foreach (var item in Model)
 {
 <div class="row panel panel-default">
  <div class="col-xs-3">
    <a href="/home/aticle?ArticleId=@item.Id">
        <img class="img-responsive" style="max-height:200px;max-width:100%;"
src="@item.Thumbnail" alt="@item.Title" />
    </a>
</div>
<div class="col-xs-9 panel-default">
    <div class="panel-heading">
        <h4><a href="/home/aticle?ArticleId=@item.Id">@item.Title.ToUpper()</a></h4>
    </div>
    <div class="panel-body">
        <p>@item.Summery</p>
    </div>
```

```
</div>
```

9. Right click on Model folder in your solution and Add a CS class file. Add below Model classes

```
 using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Web;

 namespace SharePointNewsAppWeb.Models
{
  public class NewsApp
   {

   }
 public class NewsList
  {

public int Id { get; set; }

public string Title { get; set; }

public string Summery { get; set; }

public string Thumbnail { get; set; }
 }
 public class FullArticle
 {

  public int Id { get; set; }

  public string Title { get; set; }

  public string Body { get; set; }

 }
}
```

10. Use the F5 key to deploy and run your add-in. If you see a Security Alert window that asks you to trust the self-signed Localhost certificate, choose Yes.

And now first App is ready

Creating Full article page

We have already created first page which will show all the news articles. This page will show Complete article.

1. Add One more Action Method to HomeController

```
[SharePointContextFilter]
public ActionResult Aticle(int ArticleId)
{
    User spUser = null;
```

```
    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);
    FullArticle article = new FullArticle();
    using (var clientContext = spContext.CreateUserClientContextForSPHost())
    {
        if (clientContext != null)
        {
            spUser = clientContext.Web.CurrentUser;

            clientContext.Load(spUser, user => user.Title);

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

            List lst = clientContext.Web.Lists.GetByTitle("News");
            CamlQuery queryNews = new CamlQuery();
            queryNews.ViewXml = @"<View><Query><Where><Eq><FieldRef Name='ID'/>" +
"<Value Type='Number'>" + ArticleId + "</Value></Eq></Where></Query>" +
                "<ViewFields><FieldRef Name='ID'/><FieldRef Name='Title'/><FieldRef
Name='Body'/></ViewFields></View>";//
            ListItemCollection newsItems = lst.GetItems(queryNews);
            clientContext.Load(newsItems, includes => includes.Include(i => i.Id, i =>
i.DisplayName, i => i["Body"]));

            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
                foreach (var lstProductItem in newsItems)
                {
                    article.Id = Convert.ToInt32(lstProductItem.Id.ToString());
                    article.Title = lstProductItem.DisplayName.ToString();
                    article.Body = lstProductItem["Body"].ToString();
                }
            }
        }
    }
    return View(article);
}
```

2. Again Right click on Action and create a View with same name Action method name. In My case View will be called **Aticle**

```
  @model SharePointNewsAppWeb.Models.FullArticle

@{
 ViewBag.Title = "Aticle";
}

<br />
<div class="panel panel-default">
 <div class="panel-heading"><a style="font-size:20px;" href="/"><i   class="glyphicon
glyphicon-chevron-left"></i> <i class="glyphicon glyphicon-home"></i>      </a></div>
 <div class="panel-heading"><h1 class="h2">@Model.Title.ToUpper()</h1></div>
 <div class="panel-body">@Html.Raw(@Model.Body)</div>
</div>
```

This is the code for Full article page which shows Body of the news article

Read Creating a provider hosted App online: https://riptutorial.com/sharepoint/topic/6301/creating-a-provider-hosted-app

Chapter 3: Major Releases

Examples

SharePoint 2016

Build number	Description	Product
16.0.4366.1000	Cumulative Update April 2016	SharePoint Server 2016
16.0.4336.1000	RTM	SharePoint Server 2016
16.0.4327.1000	Release Candidate	SharePoint Server 2016
16.0.4266.1001	16.0.4306.1002 Beta 2	SharePoint Server 2016

SharePoint 2013

Build number	Description
15.0.4623.1001	June 2014
15.0.4631.1001	July 2014
15.0.4641.1001	August 2014
15.0.4649.1001	September 2014
15.0.4659.1001	October 2014
15.0.4667.1000	November 2014
15.0.4675.1000	December 2014
15.0.4693.1001	February 2015
15.0.4701.1001	March 2015
15.0.4711.1000	April 2015 (SP1 REQ)
15.0.4719.1002	May 2015
15.0.4727.1001	June 2015
15.0.4737.1000	July 2015
15.0.4745.1000	August 2015

Build number	Description
15.0.4753.1003	September 2015
15.0.4763.1002	October 2015
15.0.4771.1000	November 2015
15.0.4779.1000	December 2015
15.0.4787.1000	MS16-004
15.0.4787.1000	Januar 2016
15.0.4797.1001	February 2016
15.0.4805.1000	March 2016
15.0.4815.1000	April 2016
15.0.4823.1003	May 2016
15.0.4833.1000	June 2016

Source: SharePoint 2013 Build Numbers and CU's

Read Major Releases online: https://riptutorial.com/sharepoint/topic/2737/major-releases

Chapter 4: REST Services

Remarks

REST Service Endpoint URLs

The REST client access API was first introduced in SharePoint 2010, but was greatly expanded in SharePoint 2013. The REST API in SharePoint 2010 is accessed through the ListData web service at the `/_vti_bin/ListData.svc` url. SharePoint 2013 introduced the `/_api/lists/` and `/_api/web` endpoint URLs, which behave slightly differently.

The above endpoint URLs should be preceded by `http://server/site` where `server` represents the name of the server, and `site` represents the name of, or path to, the specific site.

Example URL for...	SharePoint 2010	SharePoint 2013
Fetching a List:	`/_vti_bin/ListData.svc/ListName`	`/_api/lists('ListGuid')`
Fetching an Item:	`/_vti_bin/ListData.svc/ListName(1)`	`/_api/lists('ListGuid')/items(1)`
Fetching a Web:	(no equivalent)	`/_api/web`

Despite the differences in accessing lists and list items, working with those results is very similar in both versions.

Note that the `ListData.svc` service is still available in SharePoint 2013 for backwards compatibility.

Sending REST Requests

A REST request can be submitted via a native JavaScript XMLHttpRequest or via the jQuery AJAX wrapper construct.

XMLHttpRequest Syntax

```
var xhr = new XMLHttpRequest();
xhr.open(verb, url, true);
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);
```

jQuery AJAX Syntax

```
$.ajax({
```

```
    method: verb,
    url: url,
    headers: { "Content-Type":"application/json" },
    data: data
});
```

For more details on sending requests via AJAX, see the JavaScript AJAX documentation.

Examples

Working with Lists

Getting List Items

This example shows how to retrieve all list items and iterate through them. You can use the `top` parameter to request a certain number of results. You can also use the `select` parameter to select certain fields (`$select=id, Title, uri`).

JavaScript

```
function GetListItems(){
    $.ajax({
        url: "../_api/web/lists/getbytitle('List Title')/items?$top=50"
        contentType: "application/json;odata=verbose",
        method: "GET",
        headers: { "accept": "application/json;odata=verbose" },
        success: function (data) {
            $.each(data.d.results, function(index,item){
                //use item to access the individual list item
                console.log(item.Id);
            });
        },
        error: function(error){
            console.log(error);
        }
    });
    }
```

Getting an individual list item

JavaScript

```
function GetListItem(){
    $.ajax({
        url: "../_api/web/lists/getbytitle('List Title')/items(1)",
        contentType: "application/json;odata=verbose",
        method: "GET",
        headers: { "accept": "application/json;odata=verbose" },
        success: function (data) {
            console.log(data.d.Id);
        },
        error: function(error){
            console.log(error);
        }
```

```
    });
    }
```

Get List Items with Lookup Columns

Sometimes, you may have a list structure that looks like this:

Animal Listing Table

Name	Type	Description
Title	String (Text)	Name of the animal
Age	Number	How old the animal is
Value	Currency	Value of the animal
Type	*Lookup (Animal Types Table)*	Lookup Field (Gives dropdown of choices from Animal Types Table)

Animal Types Table

Name	Type	Description
Title	String (Text)	Name of the species/animal type (ex. Pig)
NumLegs	Number	Number of legs on the animal

Probably not the most serious example but the problem here is still valid. When you use the usual request in order to retrieve values from the SharePoint list, for the `Type` of the animal, you will only get back a field called `TypeId` in the JSON response. In order to expand these items out in just a single AJAX call, some extra markup is required in the URL parameters.

This example applies to more than just lookup columns too. When you are using `People/Groups` columns, they are essentially just lookups as well, so you can pull items such as `Title`, `EMail`, and others easily.

Example Code

Important Note: When you define the fields that you want to get back from the lookup columns, you must prefix the name of the field with the name of the lookup field in the original table. For example, if you want to get back the `NumLegs` attribute from the lookup column, you must type `Type/NumLegs`.

JavaScript

```
// webUrl: The url of the site (ex. https://www.contoso.com/sites/animals)
// listTitle: The name of the list you want to query
// selectFields: the specific fields you want to get back
// expandFields: the name of the fields that need to be pulled from lookup tables
// callback: the name of the callback function on success
function getItems(webUrl,listTitle,selectFields, expandFields, callback){
    var endpointUrl = webUrl + "/_api/web/lists/getbytitle('" + listTitle + "')/items";
    endpointUrl+= '?$select=' + selectFields.join(",");
    endpointUrl+= '&$expand=' + expandFields.join(",");
    return executeRequest(endpointUrl,'GET', callback);
}

function executeRequest(url,method,callback,headers,payload)
{
    if (typeof headers == 'undefined'){
        headers = {};
    }
    headers["Accept"] = "application/json;odata=verbose";
    if(method == "POST") {
        headers["X-RequestDigest"] = $("#__REQUESTDIGEST").val();
    }

    var ajaxOptions =
    {
    url: url,
    type: method,
    contentType: "application/json;odata=verbose",
    headers: headers,
    success: function (data) { callback(data) }
    };
    if(method == "POST") {
    ajaxOptions.data = JSON.stringify(payload);
    }

    return $.ajax(ajaxOptions);
}

// Setup the ajax request by setting all of the arguments to the getItems function
function getAnimals() {
    var url = "https://www.contoso.com/sites/animals";
    var listTitle = "AnimalListing";

    var selectFields = [
        "Title",
        "Age",
        "Value",
        "Type/Title",
        "Type/NumLegs"
    ];

    var expandFields = [
        "Type/Title",
        "Type/NumLegs"
    ];

    getItems(url, listTitle, selectFields, expandFields, processAnimals);
}

// Callback function
```

```
// data: returns the data given by SharePoint
function processAnimals(data) {
    console.log(data);
    // Process data here
}

// Start the entire process
getAnimals();
```

Adding selections to a multivalue lookup field

This example assumes that your lookup column is named `MultiLookupColumnName` and that you want to set your multi-lookup field to lookup to the items with IDs 1 and 2.

Using jQuery AJAX

2010

```
var listName = "YourListName";
var lookupList = "LookupListName";
var idOfItemToUpdate = 1;
var url = "/server/site/_vti_bin/ListData.svc/"+listName+"("+idOfItemToUpdate+")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(2)"}}
    ]
});
$.ajax({
    method: 'POST',
    url: url,
    contentType: 'application/json',
    headers: {
      "X-HTTP-Method" : "MERGE",
      "If-Match" : "*"
    },
    data: data
});
```

2013

```
var listGuid = "id-of-list-to-update"; // use list GUID here
var lookupGuid = "id-of-lookup-list"; // use lookup list GUID here
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('"+ listGuid + "')/items("+ idOfItemToUpdate + ")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
    ]
});
$.ajax({
    method: 'POST',
    url: url,
    contentType: 'application/json',
    headers: {
      "X-HTTP-Method" : "MERGE",
```

```
      "If-Match" : "*"
    },
    data: data
});
```

Using XMLHttpRequest

2010

```
var listName = "YourListName";
var lookupList = "LookupListName";
var idOfItemToUpdate = 1;
var url = "/server/site/_vti_bin/ListData.svc/YourListName("+idOfItemToUpdate+")";
var data = JSON.stringify({
    MultiLookupColumnName:[
       {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(2)"}}
    ]
});
var xhr = new XMLHttpRequest();
xhr.open("POST",url,true);
xhr.setRequestHeader("X-HTTP-Method", "MERGE");
xhr.setRequestHeader("If-Match", "*");
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);
```

2013

```
var listGuid = "id-of-list-to-update";
var lookupGuid = "id-of-lookup-list";
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('"+ listGuid + "')/items("+ idOfItemToUpdate + ")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
    ]
});
var xhr = new XMLHttpRequest();
xhr.open("POST",url,true);
xhr.setRequestHeader("X-HTTP-Method", "MERGE");
xhr.setRequestHeader("If-Match", "*");
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);
```

Paging list items returned from a query

To simulate paging using REST you can do the following:

1. Use the `$skip=n` parameter to skip the first `n` entries according to the `$orderby` parameter
2. Use the `$top=n` parameter to return the top `n` entries according to the `$orderby` and `$skip` parameters.

```
var endpointUrl = "/_api/lists('guid')/items"; // SP2010: "/_vti_bin/ListData.svc/ListName";
$.getJSON(
```

```
    endpointUrl + "?$orderby=Id&$top=1000",
    function(data){
        processData(data); // you can do something with the results here
        var count = data.d.results.length;
        getNextBatch(count, processData, onComplete); // fetch next page
    }
);

function getNextBatch(totalSoFar, processResults, onCompleteCallback){
    $.getJSON(
        endpointUrl + "?$orderby=Id&$skip="+totalSoFar+"&$top=1000",
        function(data){
            var count = data.d.results.length;
            if(count > 0){
                processResults(data); // do something with results
                getNextBatch(totalSoFar+count, callback); // fetch next page
            }else{
                onCompleteCallback();
            }
        }
    );
}
```

Retrieve an ID of newly created item in SharePoint list

This example shows how to retrieve an ID of a newly created item using SharePoint REST API.

Note :

listName - This variable contains name of you list.

newItemBody - This will be your request body for adding new item in list.

e.g. var newItemBody = { __metadata: { 'type': 'SP.Data.MyListNameItem' }, Title: 'Some title value' };

```
function CreateListItemWithDetails(listName, newItemBody) {

    var item = newItemBody;
    return $.ajax({
        url: _spPageContextInfo.siteAbsoluteUrl + "/_api/web/lists/getbytitle('" + listName +
"')/items",
        type: "POST",
        contentType: "application/json;odata=verbose",
        data: JSON.stringify(item),
        headers: {
            "Accept": "application/json;odata=verbose",
            "X-RequestDigest": $("#__REQUESTDIGEST").val(),
            "content-Type": "application/json;odata=verbose"
        }

    });
}

CreateListItemWithDetails(listName, newItemBody)
    .then(function(data){
        //success callback
```

```
        var NewlyCreatedItemId = data.d.ID;
    }, function(data){
        //failure callback
    });
```

How to perform CRUD operations using SharePoint 2010 REST Interface

Create

In order to perform a Create operation via REST, you must perform the following actions:

Create an HTTP request using the `POST` verb. Use the service URL of the list to which you want to add an entity as the target for the POST. Set the content type to `application/json`. Serialize the JSON objects that represent your new list items as a string, and add this value to the request body JavaScript example:

```
function createListItem(webUrl,listName, itemProperties, success, failure) {

    $.ajax({
        url: webUrl + "/_vti_bin/listdata.svc/" + listName,
        type: "POST",
        processData: false,
        contentType: "application/json;odata=verbose",
        data: JSON.stringify(itemProperties),
        headers: {
            "Accept": "application/json;odata=verbose"
        },
        success: function (data) {
            success(data.d);
        },
        error: function (data) {
            failure(data.responseJSON.error);
        }
    });
}
```

Usage

```
var taskProperties = {
    'TaskName': 'Order Approval',
    'AssignedToId': 12
};

createListItem('https://contoso.sharepoint.com/project/','Tasks',taskProperties,function(task){

    console.log('Task' + task.TaskName + ' has been created');
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Read

In order to perform a Read operation via REST, you must perform the following actions:

Create an HTTP request using the `GET` verb. Use the service URL of the list item to which you want to add an entity as the target for the GET. Set the content type to `application/json`. JavaScript example:

```
function getListItemById(webUrl,listName, itemId, success, failure) {
    var url = webUrl + "/_vti_bin/listdata.svc/" + listName + "(" + itemId + ")";
    $.ajax({
        url: url,
        method: "GET",
        headers: { "Accept": "application/json; odata=verbose" },
        success: function (data) {
            success(data.d);
        },
        error: function (data) {
            failure(data.responseJSON.error);
        }
    });
}
```

Usage

```
getListItemById('https://contoso.sharepoint.com/project/','Tasks',2,function(taskItem){
    console.log(taskItem.TaskName);
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Update

To update an existing entity, you must perform the following actions:

Create an HTTP request using the `POST` verb. Add an `X-HTTP-Method` header with a value of `MERGE`. Use the service URL of the list item you want to update as the target for the POST Add an `If-Match` header with a value of the entity's original ETag, or *. JavaScript example:

```
function updateListItem(webUrl,listName,itemId,itemProperties,success, failure)
{
   getListItemById(webUrl,listName,itemId,function(item){

      $.ajax({
         type: 'POST',
         url: item.__metadata.uri,
         contentType: 'application/json',
         processData: false,
         headers: {
                "Accept": "application/json;odata=verbose",
                "X-HTTP-Method": "MERGE",
                "If-Match": item.__metadata.etag
         },
         data: Sys.Serialization.JavaScriptSerializer.serialize(itemProperties),
         success: function (data) {
                success(data);
         },
         error: function (data) {
```

```
                failure(data);
            }
        });

    },
    function(error){
        failure(error);
    });

}
```

Usage

```
var taskProperties = {
    'TaskName': 'Approval',
    'AssignedToId': 12
};


updateListItem('https://contoso.sharepoint.com/project/','Tasks',2,taskProperties,function(item){

    console.log('Task has been updated');
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Delete

To delete an entity, you must perform the following actions:

Create an HTTP request using the `POST` verb. Add an `X-HTTP-Method` header with a value of `DELETE`. Use the service URL of the list item you want to update as the target for the POST Add an `If-Match` header with a value of the entity's original ETag. JavaScript example:

```
function deleteListItem(webUrl, listName, itemId, success, failure) {
    getListItemById(webUrl,listName,itemId,function(item){
        $.ajax({
            url: item.__metadata.uri,
            type: "POST",
            headers: {
                "Accept": "application/json;odata=verbose",
                "X-Http-Method": "DELETE",
                "If-Match": item.__metadata.etag
            },
            success: function (data) {
                success();
            },
            error: function (data) {
                failure(data.responseJSON.error);
            }
        });
    },
   function (error) {
       failure(error);
   });
```

```
}
```

Usage

```
deleteListItem('https://contoso.sharepoint.com/project/','Tasks',3,function(){
    console.log('Task has been deleted');
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Read REST Services online: https://riptutorial.com/sharepoint/topic/3045/rest-services

Chapter 5: SharePoint 2013 Client Side Rendering

Introduction

Client Side Rendering (CSR) is a new concept that is introduced in SharePoint 2013. It provides you with a mechanism that allow you to use your own output render for a set of controls that are hosted in a SharePoint page (list views, list forms and search results). Client Site Rendering is simply when the data is transformed using the client rather than the server. This means using client-side technologies, such as HTML and JavaScript rather than having to write XSLT.

Examples

Change hyperlink of fields/columns inside the list view using CSR

Below example shows how to change the hyperlink for "**ID**" and "**Title(LinkTitle)**" field inside the list view using CSR.

Step1 : Create a JS file and paste below code

```
(function () {

    function registerRenderer() {
        var ctxForm = {};
        ctxForm.Templates = {};

        ctxForm.Templates = {
            Fields : {
                'LinkTitle': { //------ Change Hyperlink of LinkTitle
                    View : function (ctx) {
                        var url = String.format('{0}?ID={1}',
"/sites/Lists/testlist/EditItem.aspx", ctx.CurrentItem.ID);
                        return String.format('<a href="{0}" onclick="EditItem2(event,
\'{0}\');return false;">{1}</a>', url, ctx.CurrentItem.Title);
                    }
                },
                'ID' : { //------ Change Hyperlink from ID field
                    View : function (ctx) {
                        var url = String.format('{0}?ID={1}',
"/IssueTracker/Lists/testlist/DisplayItem.aspx", ctx.CurrentItem.ID);
                        return String.format('<a href="{0}" onclick="EditItem2(event,
\'{0}\');return false;">{1}</a>', url, ctx.CurrentItem.ID);
                    }
                },

            }
        };
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');
```

```
})();
```

Step 2 : GoTo web part properties of List View and add JS Link reference to this newly created js file (e.g. ~sitecollection/SiteAssets/CSRCodeFile.js)

(Note : Refer your JSlink in this format only. "~sitecollection/YourJSfFilePath".)

Step 3 : Appy and Done

Hide column from SharePoint list view using CSR.

This example shows how to hide a "Date" field from the SharePoint list view using CSR.

```
(function () {

    function RemoveFields(ctx) {
        var fieldName = "Date"; // here Date is field or column name to be hide
        var header = document.querySelectorAll("[displayname=" + fieldName +
"]")[0].parentNode;
        var index = [].slice.call(header.parentNode.children).indexOf(header) + 1;
        header.style.display = "none";
        for (var i = 0, cells = document.querySelectorAll("td:nth-child(" + index + ")"); i <
cells.length; i++) {
            cells[i].style.display = "none";
        }
    }

    function registerRenderer() {
        var ctxForm = {};
        ctxForm.Templates = {};
        ctxForm.OnPostRender = RemoveFields;
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');

})();
```

Apply validations on New/Edit Item Form using CSR

Suppose we have a SharePoint list and it has four fields viz. Title, Full Name, Email, Mobile Number etc. Now if you want to apply custom validation in New/Edit Item form, you can easily do it with CSR code. The below mentioned can validate following conditions in forms:

- Blank values in fields
- Email id format check with regular expression
- Mobile Number format Check with regular expression
- Full Name field should not contain numeric values

Step : 1 Create a JS file, say `CSRValidations.js` and copy paste following code in the JS file

```
        (function () {

        // Create object that have the context information about the field that we want to
```

```
change it's output render
        var fieldContext = {};
        fieldContext.Templates = {};
        fieldContext.Templates.Fields = {
            // Apply the new rendering for Email field on New and Edit Forms
            "Title": {
                "NewForm": titleFieldTemplate,
                "EditForm":  titleFieldTemplate
            },
            "Full_x0020_Name": {
                "NewForm": fullNameFieldTemplate,
                "EditForm":  fullNameFieldTemplate
            },
            "Email": {
                "NewForm": emailFieldTemplate,
                "EditForm":  emailFieldTemplate
            },
            "Mobile_x0020_Phone": {
                "NewForm": mobilePhoneFieldTemplate,
                "EditForm":  mobilePhoneFieldTemplate
            }
        };

        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(fieldContext);

    })();

    // This function provides the rendering logic
    function emailFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

        // Register a callback just before submit.
        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpEmail').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new emailValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, emailOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpEmail' class='ms-long'> \ <br><span id='spnEmailError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate email format
    emailValidator = function () {
        emailValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            //Email format Regex expression
            //var emailRejex = /\S+@\S+\.\S+/;
            var emailRejex = /^(([^<>()[\]HYPERLINK
"\\.,;:\s@\"\\.,;:\s@\"]+(\.[^<>()[\]HYPERLINK "\\.,;:\s@\"\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
```

```
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!emailRejex.test(value) && value.trim()) {
                isError = true;
                errorMessage = "Please enter valid email address";
            }

            //Send error message to error callback function (emailOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
    };

    // Add error message to spnError element under the input field element
    function emailOnError(error) {
        document.getElementById("spnEmailError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }

    // This function provides the rendering logic
    function titleFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);
        // Register a callback just before submit.

        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpTitle').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new titleValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, titleOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpTitle' class='ms-long'> \ <br><span id='spnTitleError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate title format
    titleValidator = function () {
        titleValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }

            //Send error message to error callback function (titleOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
```

```
    };

    // Add error message to spnError element under the input field element
    function titleOnError(error) {
        document.getElementById("spnTitleError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }

    // This function provides the rendering logic
    function mobilePhoneFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

        // Register a callback just before submit.
        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpMobilePhone').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new mobilePhoneValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, mobilePhoneOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpMobilePhone' class='ms-long'> \ <br><span id='spnMobilePhoneError'
class='ms-formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate mobilePhone format
    mobilePhoneValidator = function () {
        mobilePhoneValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            //MobilePhone format Regex expression
            //var mobilePhoneRejex = /\S+@\S+\.\S+/;
            var mobilePhoneRejex = /^[0-9]+$/;

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!mobilePhoneRejex.test(value) && value.trim()) {
                isError = true;
                errorMessage = "Please enter valid mobile phone number";
            }

            //Send error message to error callback function (mobilePhoneOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
    };

    // Add error message to spnError element under the input field element
    function mobilePhoneOnError(error) {
        document.getElementById("spnMobilePhoneError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }
```

```
    // This function provides the rendering logic
    function fullNameFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

        // Register a callback just before submit.
        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpFullName').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new fullNameValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, fullNameOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpFullName' class='ms-long'> \ <br><span id='spnFullNameError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate fullName format
    fullNameValidator = function () {
        fullNameValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            //FullName format Regex expression
            var fullNameRejex = /^[a-z ,.'-]+$/i;

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!fullNameRejex.test(value) && value.trim()) {
                isError = true;
                errorMessage = "Please enter valid name";
            }

            //Send error message to error callback function (fullNameOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
    };

    // Add error message to spnError element under the input field element
    function fullNameOnError(error) {
        document.getElementById("spnFullNameError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }
```

Step : 2 Open New Item Form in browser. Edit page and edit web part.

Step : 3 In Web part properties, Go to Miscellaneous --> JS link --> paste the path of your js file (e.g. ~sitecollection/SiteAssets/CSRValidations.js)

Step : 4 Save Web part properties and page.

Change column display name in list view using CSR

There are cases when you need to change Display Name of column in a list view

e.g. Column Name showing in the view is "IsApprovalNeeded" and you want to appear as "Is Approval Needed?".

You can, of course change the display name of a column by changing the column title in list settings, but if you want to keep it as it is in the list settings and only modify it on the page preview then you can do it by using CSR(Client-Side-Rendering).

Here is the code...

```
(function () {

    function preTaskFormRenderer(renderCtx) {
       modifyColumns(renderCtx);
    }

    function modifyColumns(renderCtx)
    {
      var arrayLength= renderCtx.ListSchema.Field.length;
        for (var i=0; i < arrayLength;i++)
        {
           if(renderCtx.ListSchema.Field[i].DisplayName == 'IsApprovalNeeded')
             {
               var newTitle= "Is Approval Needed?";
               var linkTitleField = renderCtx.ListSchema.Field[i];
               linkTitleField.DisplayName = newTitle;
             }
         }
    }

    function registerRenderer()
    {
      var ctxForm = {};
      ctxForm.Templates = {};
      ctxForm.OnPreRender = preTaskFormRenderer;
      SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }

    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');

})();
```

Read SharePoint 2013 Client Side Rendering online:
https://riptutorial.com/sharepoint/topic/8317/sharepoint-2013-client-side-rendering

Chapter 6: SharePoint App

Introduction

SharePoint Hosted App

Remarks

Reference required from site: http://www.letsharepoint.com/what-is-user-information-list-in-sharepoint-2013/

Examples

SharePoint 2013: Access User Profile Service Data using JSOM in SharePoint 2013

SharePoint 2013: Access User Profile Service Data using JSOM in SharePoint 2013

In this article, we will learn to manage or access User Profile Service(UPS) Application using JSOM (Javascript Object Model) and create a basic App. Before we start, lets go through basic UPS terminology first.

User Profile – It has all the information of people in an organization in an organized manner. It displays all properties like AccountName, FirstName, LastName, WorkEmail etc. related to a user.

User Profile Service Application – It is considered as centralized location to store all user profiles and also allows the administrators to configure or manage profiles, profile synchronization, My Site, Social Tags etc. It can also pull information from Directory Services like Active Directory.

My Site – A personalized site for individual user to manage their information and store documents, links etc. It provides rich networking and social features by enabling users to share information about themselves or their activities. My Site is accessible by clicking on User Name on top right corner of SharePoint page.

Manage and Access User Profile Data

Since we are going to work using JSOM, we can perform only 'Read ' operations with an exception that Profile picture can be changed using JSOM (or CSOM or REST)

*Server Side Code allows Read/Write both operations.

Retrieve User Profile Properties using JSOM

Lets create a SharePoint Hosted App and retrieve user information in that app –

Launch Visual Studio 2013 and select "App for SharePoint 2013" from New Project. After selecting

above project type, you are presented with a window to connect to SharePoint site and select type of app to be deployed (see below screenshot) Here I have provided mu SharePoint Online developer site URL and selected SharePoint Hosted App. Click Finish.

3.) After the project is created, you will see a set of folders/filers added in Solution Explorer added by default to the project.

4.) If you open "Default.aspx" page, you will find some JavaScript libraries already added to page.

Here we need to add one more library to start working with User Profiles

Read SharePoint App online: https://riptutorial.com/sharepoint/topic/9876/sharepoint-app

Chapter 7: Working with JavaScript Client Object Model (JSOM)

Remarks

Background

The JavaScript Object Model was introduced in SharePoint 2010. It exposes on the client side many of the objects that were previously only accessible through server-side code or through dedicated web services.

Embedding JavaScript in SharePoint Pages

In SharePoint 2013 you can put your JavaScript in a Script Editor web part.

In SharePoint 2010 you can use the "content link" property of a Content Editor web part to link to an HTML file that contains your embedded script.

Object Reference

The constructors, methods, and properties of all objects found in the `SP` namespace are documented in the SharePoint 2013 client object model reference here.

The SharePoint 2010 JavaScript client object model reference is available here.

JSOM's Asynchronous Programming Pattern

When using the JavaScript client object model, code generally takes the following pattern:

1. Obtain a `ClientContext` object.
2. Use the `ClientContext` object to retrieve objects representing entities in the SharePoint object model, such as lists, folder, views.
3. Queue up instructions to be performed against the objects. These instructions are not transmitted to the server yet.
4. Use the `load` function to tell the `ClientContext` what information you want to receive back from the server.
5. Invoke the `ClientContext` object's `executeQueryAsync` function to send the queued instructions to the server, passing two callback functions to run on success or failure.
6. In the callback function, work with the results returned from the server.

Alternatives

Client-side alternatives to the JSOM include SharePoint's web services, REST endpoints, and the .NET client object model.

Examples

Getting library content types using the library name

```
function getContentTypes(site_url,name_of_the_library){
    var ctx = new SP.ClientContext(site_url);
    var web = ctx.get_web();
    list = web.get_lists().getByTitle(name_of_the_library);

    // You can include any property of the SP.ContentType object (sp.js), for this example we
are just getting the name
    ctx.load(list,'ContentTypes.Include(Name)');
    ctx.executeQueryAsync(onQuerySucceeded, onQueryFailed);
}

function onQuerySucceeded(sender, args) {
    // var list is the one that we used in function "getContentTypes"
    var contentTypesEnumerator = (list.get_contentTypes()).getEnumerator();

    while (contentTypesEnumerator.moveNext()) {
        var contentType = contentTypesEnumerator.get_current();
        alert(contentType.get_name());
    }
}

function onQueryFailed(sender, args) {
    alert('Request failed. ' + args.get_message() + '\n' + args.get_stackTrace());
}
```

Delete an item in a list

```
SP.SOD.executeOrDelayUntilScriptLoaded( function(){ deleteItem(1); }, "sp.js");

function deleteItem(id){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var item = list.getItemById(id);
    item.deleteObject();
    clientContext.executeQueryAsync(function(){
        alert("Item #"+id+" deleted successfully!");
    },function(sender,args){alert(args.get_message());});
}
```

Creating Items or Folders

Creating List Items

```
SP.SOD.executeOrDelayUntilScriptLoaded(createItem,"sp.js");

function createItem(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var newItem = list.addItem();
    newItem.set_item("Title","Example Title");
    newItem.update();
    clientContext.load(newItem); // only needed to retrieve info from newly created item
```

```
    clientContext.executeQueryAsync(function(){
        var itemId = newItem.get_item("ID");
        alert("Item #"+itemId+" Created Successfully!");
    },function(sender,args){
        alert(args.get_message());
    });
}
```

The example above demonstrates that a list item is created by performing the following:

1. Call the `addItem` method of a list object to get an item object
2. Call the `set_item` method on the resulting list item object to set each field value as desired
3. Call the `update` method on the list item object to indicate that the changes are to be committed
4. Call the `executeQueryAsync` method of the client context object to execute the queued instructions

Note that you do **not** need to pass the new item object to the client context's `load` method to create the item. That step is only necessary if you wish to retrieve any of the item's field values from the server.

Creating Folders

Creating a folder is similar to adding an item to a list. The difference is that one must first create a `ListItemCreationInformation` object and set its `underlyingObjectType` property to `SP.FileSystemObjectType.folder`, and its `leafName` property to the desired name of the new folder.

The object is then passed as a parameter in the `addItem` method on the library to create the folder.

```
// ...
var itemCreateInfo = new SP.ListItemCreationInformation();
itemCreateInfo.set_underlyingObjectType(SP.FileSystemObjectType.folder);
itemCreateInfo.set_leafName(folderName);
var newItem = list.addItem(itemCreateInfo);
// ...
```

To commit the change, invoke the `executeQueryAsync` method of the `ClientContext` object through which the library was accessed.

The full example below creates a folder with a name based on the current timestamp, then opens that folder in a modal dialog.

```
SP.SOD.executeOrDelayUntilScriptLoaded(createFolder,"sp.js");

function createFolder(){
    var now = new Date();
    var timeStamp = now.getYear() + "-" + (now.getMonth()+1) + "-" + now.getDate()
        + "T" + now.getHours()+"_"+now.getMinutes()+"
"+now.getSeconds()+"_"+now.getMilliseconds();
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("Library Title");
```

```
    var itemCreateInfo = new SP.ListItemCreationInformation();
    itemCreateInfo.set_underlyingObjectType(SP.FileSystemObjectType.folder);
    itemCreateInfo.set_leafName(timeStamp);
    var newItem = list.addItem(itemCreateInfo);
    newItem.update();
    clientContext.load(newItem);
    var rootFolder = list.get_rootFolder(); // Note: use a list's root folder to determine its
server relative URL
    clientContext.load(rootFolder);
    clientContext.executeQueryAsync(function(){
        var itemId = newItem.get_item("ID");
        var name = newItem.get_item("FileLeafRef");
        SP.UI.ModalDialog.showModalDialog(
            {
                title: "Folder \""+name+"\" (#"+itemId+") Created Successfully!",
                url: rootFolder.get_serverRelativeUrl() + "/" + name
            }
        );
    },function(sender,args){alert(args.get_message());});
}
```

Get Current User Information

```
SP.SOD.executeOrDelayUntilScriptLoaded(showUserInfo,"sp.js");

function showUserInfo(){
    var clientContext = new SP.ClientContext();
    var user = clientContext.get_web().get_currentUser();
    clientContext.load(user);
    clientContext.executeQueryAsync(function(){
            var details = "ID: "+user.get_id()+"\n"+
                "Title: "+user.get_title()+"\n"+
                "Login: "+user.get_loginName()+"\n"+
                "Email: "+user.get_email();
            alert(details);
        },function(sender,args){alert(args.get_message());})
}
```

Get a List Item by ID

```
SP.SOD.executeOrDelayUntilScriptLoaded(myFunction,"sp.js");

function myFunction(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var item = list.getItemById(1); // get item with ID == 1
    clientContext.load(item);
    clientContext.executeQueryAsync(
        function(){ // onSuccess
            var title = item.get_item("Title");
            alert(title);
        },
        function(sender,args){ // onError
            alert(args.get_message());
        }
    );
}
```

Basic Example

Use the `set_viewXml` method of the SP.CamlQuery object to specify a CAML query to retrieve items.

```
SP.SOD.executeOrDelayUntilScriptLoaded(showListItems,"core.js");

function showListItems(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml(
        "<View><Query>" +
            "<Where>" +
                "<Eq><FieldRef Name=\"Title\"/><Value Type=\"Text\">Value</Value></Eq>" +
            "</Where>" +
            "<OrderBy><FieldRef Name=\"Modified\" Ascending=\"FALSE\"/></OrderBy>" +
        "</Query>"+
        //"<RowLimit>5000</RowLimit>" +
        "</View>");
    var items = list.getItems(camlQuery);
    clientContext.load(items);
    clientContext.executeQueryAsync(function(){
        var itemArray = [];
        var itemEnumerator = items.getEnumerator();
        while(itemEnumerator.moveNext()){
            var item = itemEnumerator.get_current();
            var id = item.get_item("ID");
            var title = item.get_item("Title");
            itemArray.push(id + ": " + title);
        }
        alert("ID: Title\n"+itemArray.join("\n"));
    },function(sender,args){alert(args.get_message());});
}
```

Paging the results of a CAML query

You can take advantage of the `RowLimit` element in a CAML query to retrieve only a subset of results with each query.

Use the `get_listItemCollectionPosition` method of a list item collection to retrieve the current position, then use that value as the parameter in an SP.CamlQuery object's `set_listItemCollectionPosition` method to retrieve the next batch of results.

```
SP.SOD.executeOrDelayUntilScriptLoaded(showListItems,"sp.js");

function showListItems(){
    var itemArray = [];
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
```

```
    var viewXml =
        "<View><Query>" +
            "<OrderBy><FieldRef Name=\"Modified\" Ascending=\"FALSE\"/></OrderBy>" +
        "</Query>"+
          "<RowLimit>1</RowLimit>" +
        "</View>";
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml(viewXml);
    var items = list.getItems(camlQuery);
    clientContext.load(items);
    clientContext.executeQueryAsync(loadResults,showError);

    function loadResults(){
        var resultsFound = false;
        var itemEnumerator = items.getEnumerator();
        while(itemEnumerator.moveNext()){
            var item = itemEnumerator.get_current();
            var id = item.get_item("ID");
            var title = item.get_item("Title");
            itemArray.push(id + ": " + title);
        }
        var pos = items.get_listItemCollectionPosition();// <- get position
        if(pos !== null){ // <-- position is null when no more results are returned
            if(confirm("Results so far: \nID: Title\n"+itemArray.join("\n"))){
                camlQuery = new SP.CamlQuery();
                camlQuery.set_listItemCollectionPosition(pos);// <- set position for next
batch
                camlQuery.set_viewXml(viewXml);
                items = list.getItems(camlQuery);
                clientContext.load(items);
                clientContext.executeQueryAsync(loadResults,showError);
            }
        }else{
            alert("Total Results: \nID: Title\n"+itemArray.join("\n")); // <- display when no
more results
        }
    }
    function showError(sender,args){
        alert(args.get_message());
    }
}
```

Read Working with JavaScript Client Object Model (JSOM) online:
https://riptutorial.com/sharepoint/topic/1316/working-with-javascript-client-object-model--jsom-

Chapter 8: Working with Managed Client Side Object Model (CSOM)

Remarks

- Most examples are from MSDN.
- To create a .NET managed client application that uses the client object model, you must set references to two client library DLLs: Microsoft.SharePoint.Client.dll and Microsoft.SharePoint.Client.Runtime.dll. You can find it in %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\16\ISAPI folder or your SharePoint server.
- or Install the Microsoft.SharePointOnline.CSOM NuGet Package, which will work "on prem" as well as in SP O365.
- Most properties are value properties and before accessing them you need to explicitly call clientContext.Load() and clientContext.ExecuteQuery(). More info here: Call Load and ExecuteQuery Before Accessing Value Properties

Examples

Hello world (getting site title)

All versions of SharePoint are based around Sites (SPSite (SSOM) or Site (CSOM)) and Webs (SPWeb(SSOM) or Web(CSOM)). A site is not rendered in the UI although it does contain metadata and features that are applied to its children. A web is the basic building block that renders a UI to the user accessing the site. All Sites have a root web that holds information and/or metadata like Document Libraries. This example shows a basic call to fetch the web located on the server `MyServer` under the virtual path `sites`.

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

namespace Microsoft.SDK.SharePointServices.Samples
{
    class RetrieveWebsite
    {
        static void Main()
        {
            // This is the URL of the target web we are interested in.
            string siteUrl = "http://MyServer/sites/MySiteCollection";
            // The client context is allows us to queue up requests for the server
            // Note that the context can only ask questions about the site it is created for
            using (ClientContext clientContext = new ClientContext(siteUrl))
            {
                // To make it easier to read the code, pull the target web
                // context off of the client context and store in a variable
                Web oWebsite = clientContext.Web;
                // Tell the client context we want to request information about the
                // Web from the server
                clientContext.Load(oWebsite);
```

```
                // After we are done creating the batch of information we need from the sever,
                // request the data from SharePoint
                clientContext.ExecuteQuery();
                // Print the results of the query
                Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title,
oWebsite.Description);
            }
        }
    }
}
```

Web. Retrieving the properties of a Web site

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
clientContext.Load(oWebsite);
clientContext.ExecuteQuery();
Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title, oWebsite.Description);
```

Web. Retrieving only specified properties of a Web site

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
clientContext.Load(
                oWebsite,
                website => website.Title,
                website => website.Created);
clientContext.ExecuteQuery();
Console.WriteLine("Title: {0} Created: {1}", oWebsite.Title, oWebsite.Created);
```

Web. Updating the title and description of a Web site

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = context.Web;
oWebsite.Title = "Updated Web Site";
oWebsite.Description = "This is an updated Web site.";
oWebsite.Update();
clientContext.ExecuteQuery();
```

Web. Creating a Web site

```
string siteUrl = "http://MyServer/sites/MySiteCollection";
string blogDescription = "A new blog Web site.";
int blogLanguage = 1033;
string blogTitle = "Blog Web Site";
string blogUrl = "blogwebsite";
bool blogPermissions = false;
string webTemplate = "BLOG#0";

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

WebCreationInformation webCreateInfo = new WebCreationInformation();
webCreateInfo.Description = blogDescription;
```

```
webCreateInfo.Language = blogLanguage;
webCreateInfo.Title = blogTitle;
webCreateInfo.Url = blogUrl;
webCreateInfo.UseSamePermissionsAsParentSite = blogPermissions;
webCreateInfo.WebTemplate = webTemplate;

Web oNewWebsite = oWebsite.Webs.Add(webCreateInfo);

clientContext.Load(
    oNewWebsite,
    website => website.ServerRelativeUrl,
    website => website.Created);

clientContext.ExecuteQuery();

Console.WriteLine("Server-relative Url: {0} Created: {1}", oNewWebsite.ServerRelativeUrl,
oNewWebsite.Created);
```

List. Retrieving all properties of all lists in a Web site

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

clientContext.Load(collList);

clientContext.ExecuteQuery();

foreach (List oList in collList)
{
    Console.WriteLine("Title: {0} Created: {1}", oList.Title, oList.Created.ToString());
}
```

List. Retrieving only specified properties of lists

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

clientContext.Load(
    collList,
    lists => lists.Include(
        list => list.Title,
        list => list.Id));

clientContext.ExecuteQuery();

foreach (List oList in collList)
{
    Console.WriteLine("Title: {0} ID: {1}", oList.Title, oList.Id.ToString("D"));
}
```

List. Storing retrieved lists in a collection

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
```

```
ListCollection collList = oWebsite.Lists;

IEnumerable<List> resultCollection = clientContext.LoadQuery(
    collList.Include(
        list=>list.Title,
        list=>list.Id));

clientContext.ExecuteQuery();

foreach (List oList in resultCollection)
{
    Console.WriteLine("Title: {0} ID: {1}", oList.Title, oList.Id.ToString("D"));
}
```

List. Retrieving list fields from a Web site

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

IEnumerable<SP.List> listInfo = clientContext.LoadQuery(
    collList.Include(
        list => list.Title,
        list => list.Fields.Include(
            field => field.Title,
            field => field.InternalName)));

 clientContext.ExecuteQuery();

foreach (SP.List oList in listInfo)
{
    FieldCollection collField = oList.Fields;

    foreach (SP.Field oField in collField)
    {
        Regex regEx = new Regex("name", RegexOptions.IgnoreCase);

        if (regEx.IsMatch(oField.InternalName))
        {
            Console.WriteLine("List: {0} \n\t Field Title: {1} \n\t Field Internal Name: {2}",

                oList.Title, oField.Title, oField.InternalName);
        }
    }
}
```

List. Creating and updating a list

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

ListCreationInformation listCreationInfo = new ListCreationInformation();
listCreationInfo.Title = "My Announcements List";
listCreationInfo.TemplateType = (int)ListTemplateType.Announcements;

List oList = oWebsite.Lists.Add(listCreationInfo);
```

```
clientContext.ExecuteQuery();
```

List. Adding a field to a list

```
ClientContext clientContext = new ClientContext(siteUrl);

SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

SP.Field oField = oList.Fields.AddFieldAsXml("<Field DisplayName='MyField' Type='Number' />",

    true, AddFieldOptions.DefaultValue);

SP.FieldNumber fieldNumber = clientContext.CastTo<FieldNumber>(oField);
fieldNumber.MaximumValue = 100;
fieldNumber.MinimumValue = 35;

fieldNumber.Update();

clientContext.ExecuteQuery();
```

List. Deleting a list

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My Announcements List");

oList.DeleteObject();

clientContext.ExecuteQuery();
```

Item. Retrieving items from a list

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
camlQuery.ViewXml = "<View><Query><Where><Geq><FieldRef Name='ID'/>" +
    "<Value Type='Number'>10</Value></Geq></Where></Query><RowLimit>100</RowLimit></View>";
ListItemCollection collListItem = oList.GetItems(camlQuery);

clientContext.Load(collListItem);

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("ID: {0} \nTitle: {1} \nBody: {2}", oListItem.Id, oListItem["Title"],
oListItem["Body"]);
}
```

Item. Retrieving items (using the Include method)

This example shows how to retrieve items from the server as well as get deeper properties of each

list item. By default, the server will only return the minimum amount of data to represent the object. It is up to the caller to request additional information from the server.

```
ClientContext clientContext = new ClientContext(siteUrl);
List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
camlQuery.ViewXml = "<View><RowLimit>100</RowLimit></View>";

ListItemCollection collListItem = oList.GetItems(camlQuery);

// The first line of this request indicates the list item collection to load from the server
// The second line uses a lambda to request that from the server
// also include additional properties in the response
// The third though fifth lines are the properties being requested from the server
clientContext.Load(collListItem,
     items => items.Include(
        item => item.Id,
        item => item.DisplayName,
        item => item.HasUniqueRoleAssignments));

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("ID: {0} \nDisplay name: {1} \nUnique role assignments: {2}",
        oListItem.Id, oListItem.DisplayName, oListItem.HasUniqueRoleAssignments);
}
```

Item. Retrieving specific fields from a specified number of items

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
ListItemCollection collListItem = oList.GetItems(camlQuery);

clientContext.Load(
    collListItem,
    items => items.Take(5).Include(
    item => item["Title"],
    item => item["Body"]));

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("Title: {0} \nBody: {1}\n", oListItem["Title"], oListItem["Body"]);
}
```

Item. Retrieving items from all the lists in a Web site

```
ClientContext clientContext = new ClientContext(siteUrl);
ListCollection collList = clientContext.Web.Lists;

clientContext.Load(
    collList,
```

```
    lists => lists.Where(
        list => list.Hidden == false).Include(
        list => list.Title,
        list => list.Items.Take(10)));

clientContext.ExecuteQuery();

foreach (SP.List oList in clientContext.Web.Lists)
{
    string listTitle = oList.Title;
    int itemCount = oList.Items.Count;

    Console.WriteLine("List {0} returned with {1} items", listTitle, itemCount);
}
```

Item. Retrieving items using list item collection position

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

ListItemCollectionPosition itemPosition = null;

while (true)
{
    CamlQuery camlQuery = new CamlQuery();

    camlQuery.ListItemCollectionPosition = itemPosition;

    camlQuery.ViewXml = "<View><ViewFields><FieldRef Name='ID'/>" +
        "<FieldRef Name='Title'/><FieldRef Name='Body'/>" +
        "</ViewFields><RowLimit>5</RowLimit></View>";

    ListItemCollection collListItem = oList.GetItems(camlQuery);

    clientContext.Load(collListItem);

    clientContext.ExecuteQuery();

    itemPosition = collListItem.ListItemCollectionPosition;

    foreach (ListItem oListItem in collListItem)
    {
        Console.WriteLine("Title: {0}: \nBody: {1}", oListItem["Title"], oListItem["Body"]);
    }

    if (itemPosition == null)
    {
        break;
    }

    Console.WriteLine("\n" + itemPosition.PagingInfo + "\n");
}
```

Item. Creating a list item

When creating a new list item, its fields can be set using syntax similar to string arrays. Note that these fields are not created on the fly and are defined by the schema of the list. These fields (or

columns) must exist on the server otherwise the create will fail. All list items will have the Title field. Some lists may have required fields that must be filled out before the item will be published in the list.

In this example, the list is using the Announcements template. In addition to the title field, the list includes the Body field that will display the contents of the announcement on the list.

```
ClientContext clientContext = new ClientContext(siteUrl);
List oList = clientContext.Web.Lists.GetByTitle("Announcements");

ListItemCreationInformation itemCreateInfo = new ListItemCreationInformation();
ListItem oListItem = oList.AddItem(itemCreateInfo);
oListItem["Title"] = "My New Item!";
oListItem["Body"] = "Hello World!";

oListItem.Update();

clientContext.ExecuteQuery();
```

Item. Updating a list item

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");
ListItem oListItem = oList.Items.GetById(3);

oListItem["Title"] = "My Updated Title.";

oListItem.Update();

clientContext.ExecuteQuery();
```

Item. Deleting a list item

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");
ListItem oListItem = oList.GetItemById(2);

oListItem.DeleteObject();

clientContext.ExecuteQuery();
```

Groups. Retrieving all users from a SharePoint group

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(7);
UserCollection collUser = oGroup.Users;

clientContext.Load(collUser);

clientContext.ExecuteQuery();

foreach (User oUser in collUser)
{
```

```
    Console.WriteLine("User: {0}  ID: {1} Email: {2} Login Name: {3}",
        oUser.Title, oUser.Id, oUser.Email, oUser.LoginName);
}
```

Groups. Retrieving specific properties of users

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(7);
UserCollection collUser = oGroup.Users;

clientContext.Load(collUser,
    users => users.Include(
        user => user.Title,
        user => user.LoginName,
        user => user.Email));

clientContext.ExecuteQuery();

foreach (User oUser in collUser)
{
    Console.WriteLine("User: {0} Login name: {1} Email: {2}",
        oUser.Title, oUser.LoginName, oUser.Email);
}
```

Groups. Retrieving all users in all groups of a site collection

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;

clientContext.Load(collGroup);

clientContext.Load(collGroup,
    groups => groups.Include(
        group => group.Users));

clientContext.ExecuteQuery();

foreach (Group oGroup in collGroup)
{
    UserCollection collUser = oGroup.Users;

    foreach (User oUser in collUser)
    {
        Console.WriteLine("Group ID: {0} Group Title: {1} User: {2} Login Name: {3}",
            oGroup.Id, oGroup.Title, oUser.Title, oUser.LoginName);
    }
}
```

Groups. Adding a user to a SharePoint group

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection ");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(6);

UserCreationInformation userCreationInfo = new UserCreationInformation();
```

```
userCreationInfo.Email = "alias@somewhere.com";
userCreationInfo.LoginName = @"DOMAIN\alias";
userCreationInfo.Title = "John";

User oUser = oGroup.Users.Add(userCreationInfo);

clientContext.ExecuteQuery();
```

Roles. Creating a role definition

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");

Web oWebsite = clientContext.Web;

BasePermissions permissions = new BasePermissions();
permissions.Set(PermissionKind.CreateAlerts);
permissions.Set(PermissionKind.ManageAlerts);

RoleDefinitionCreationInformation roleCreationInfo = new RoleDefinitionCreationInformation();

roleCreationInfo.BasePermissions = permissions;
roleCreationInfo.Description = "A new role with create and manage alerts permission";
roleCreationInfo.Name = "Create and Manage Alerts";
roleCreationInfo.Order = 4;

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.Add(roleCreationInfo);

clientContext.ExecuteQuery();

Console.WriteLine("{0} role created.", oRoleDefinition.Name);
```

Roles. Assigning a user to a role on a Web site

```
ClientContext oClientContext = new
ClientContext("http://MyServer/sites/MySiteCollection/MyWebSite");
Web oWebsite = clientContext.Web;

Principal oUser = oWebsite.SiteUsers.GetByLoginName(@"DOMAIN\alias");

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.GetByName("Create and Manage
Alerts");
RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);
collRoleDefinitionBinding.Add(oRoleDefinition);

RoleAssignment oRoleAssignment = oWebsite.RoleAssignments.Add(oUser,
collRoleDefinitionBinding);

clientContext.Load(oUser,
    user => user.Title);

clientContext.Load(oRoleDefinition,
    role => role.Name);

clientContext.ExecuteQuery();

Console.WriteLine("{0} added with {1} role.", oUser.Title, oRoleDefinition.Name);
```

Roles. Creating a SharePoint group and adding the group to a role

```
ClientContext oClientContext = new
ClientContext("http://MyServer/sites/MySiteCollection/MyWebSite");
Web oWebsite = clientContext.Web;

GroupCreationInformation groupCreationInfo = new GroupCreationInformation();
groupCreationInfo.Title = "My New Group";
groupCreationInfo.Description = "Description of new group.";
Group oGroup = oWebsite.SiteGroups.Add(groupCreationInfo);

RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.GetByType(RoleType.Contributor);

collRoleDefinitionBinding.Add(oRoleDefinition);

oWebsite.RoleAssignments.Add(oGroup, collRoleDefinitionBinding);

clientContext.Load(oGroup,
    group => group.Title);

clientContext.Load(oRoleDefinition,
    role => role.Name);

clientContext.ExecuteQuery();

Console.WriteLine("{0} created and assigned {1} role.", oGroup.Title, oRoleDefinition.Name);
}
```

Permissions. Breaking the security inheritance of a list

```
string siteUrl = "http://MyServer/sites/MySiteCollection";
ClientContext oContext = new ClientContext(siteUrl);
SP.List oList = oContext.Web.Lists.GetByTitle("Announcements");

oList.BreakRoleInheritance(true, false);

oContext.ExecuteQuery();
```

Permissions. Breaking the security inheritance of a document and adding a user as reader

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("MyList");

int itemId = 3;
ListItem oListItem = oList.Items.GetById(itemId);

oListItem.BreakRoleInheritance(false);

User oUser = clientContext.Web.SiteUsers.GetByLoginName(@"DOMAIN\alias");

RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);
```

```
collRoleDefinitionBinding.Add(clientContext.Web.RoleDefinitions.GetByType(RoleType.Reader));

oListItem.RoleAssignments.Add(oUser, collRoleDefinitionBinding);

clientContext.ExecuteQuery();
```

Permissions. Breaking the security inheritance of a document and changing the permissions of a user

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("MyList");

int itemId = 2;
ListItem oListItem = oList.Items.GetById(itemId);

oListItem.BreakRoleInheritance(true);

User oUser = clientContext.Web.SiteUsers.GetByLoginName(@"DOMAIN\alias");
oListItem.RoleAssignments.GetByPrincipal(oUser).DeleteObject();

RoleDefinitionBindingCollection collRollDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

collRollDefinitionBinding.Add(clientContext.Web.RoleDefinitions.GetByType(RoleType.Reader));

oListItem.RoleAssignments.Add(oUser, collRollDefinitionBinding);

clientContext.ExecuteQuery();
```

Custom action. Adding a user custom action for list items

```
string urlWebsite = "http://MyServer/sites/MySiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My List");
UserCustomActionCollection collUserCustomAction = oList.UserCustomActions;

UserCustomAction oUserCustomAction = collUserCustomAction.Add();
oUserCustomAction.Location = "EditControlBlock";
oUserCustomAction.Sequence = 100;
oUserCustomAction.Title = "My First User Custom Action";
oUserCustomAction.Url = urlWebsite + @"/_layouts/MyPage.aspx";
oUserCustomAction.Update();

clientContext.Load(oList,
    list => list.UserCustomActions);

clientContext.ExecuteQuery();
```

Custom action. Modifying a user custom action

```
string urlWebsite = "http://MyServer/sites/SiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);
```

```
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My List");
UserCustomActionCollection collUserCustomAction = oList.UserCustomActions;

clientContext.Load(collUserCustomAction,
    userCustomActions => userCustomActions.Include(
        userCustomAction => userCustomAction.Title));

clientContext.ExecuteQuery();

foreach (UserCustomAction oUserCustomAction in collUserCustomAction)
{
    if (oUserCustomAction.Title == "My First User Custom Action")
    {
        oUserCustomAction.ImageUrl = "http://MyServer/_layouts/images/MyIcon.png";
        oUserCustomAction.Update();

        clientContext.ExecuteQuery();
    }
}
```

Custom action. Adding a user custom action to the site actions of a Web site

```
string urlWebsite = "http://MyServer/sites/MySiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);

Web oWebsite = clientContext.Web;
UserCustomActionCollection collUserCustomAction = oWebsite.UserCustomActions;

UserCustomAction oUserCustomAction = collUserCustomAction.Add();

oUserCustomAction.Location = "Microsoft.SharePoint.StandardMenu";
oUserCustomAction.Group = "SiteActions";
oUserCustomAction.Sequence = 101;
oUserCustomAction.Title = "Website User Custom Action";
oUserCustomAction.Description = "This description appears on the Site Actions menu.";
oUserCustomAction.Url = urlWebsite + @"/_layouts/MyPage.aspx";

oUserCustomAction.Update();

clientContext.Load(oWebsite,
    webSite => webSite.UserCustomActions);

clientContext.ExecuteQuery();
```

Web part. Updating the title of a Web Part

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile = oClientContext.Web.GetFileByServerRelativeUrl("Default.aspx");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

oClientContext.Load(limitedWebPartManager.WebParts,
    wps => wps.Include(
    wp => wp.WebPart.Title));
```

```
oClientContext.ExecuteQuery();

if (limitedWebPartManager.WebParts.Count == 0)
{
    throw new Exception("No Web Parts on this page.");
}

WebPartDefinition oWebPartDefinition = limitedWebPartManager.WebParts[1];
WebPart oWebPart = oWebPartDefinition.WebPart;
oWebPart.Title = "My New Web Part Title";

oWebPartDefinition.SaveWebPartChanges();

oClientContext.ExecuteQuery();
```

Web part. Adding a Web Part to a page

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile = oClientContext.Web.GetFileByServerRelativeUrl("Default.aspx");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

string xmlWebPart = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<WebPart xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"" +
    " xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"" +
    " xmlns=\"http://schemas.microsoft.com/WebPart/v2\">" +
    "<Title>My Web Part</Title><FrameType>Default</FrameType>" +
    "<Description>Use for formatted text, tables, and images.</Description>" +
    "<IsIncluded>true</IsIncluded><ZoneID></ZoneID><PartOrder>0</PartOrder>" +
    "<FrameState>Normal</FrameState><Height /><Width /><AllowRemove>true</AllowRemove>" +
    "<AllowZoneChange>true</AllowZoneChange><AllowMinimize>true</AllowMinimize>" +
    "<AllowConnect>true</AllowConnect><AllowEdit>true</AllowEdit>" +
    "<AllowHide>true</AllowHide><IsVisible>true</IsVisible><DetailLink /><HelpLink />" +
    "<HelpMode>Modeless</HelpMode><Dir>Default</Dir><PartImageSmall />" +
    "<MissingAssembly>Cannot import this Web Part.</MissingAssembly>" +
    "<PartImageLarge>/_layouts/images/mscontl.gif</PartImageLarge><IsIncludedFilter />" +
    "<Assembly>Microsoft.SharePoint, Version=13.0.0.0, Culture=neutral, " +
    "PublicKeyToken=94de0004b6e3fcc5</Assembly>" +
    "<TypeName>Microsoft.SharePoint.WebPartPages.ContentEditorWebPart</TypeName>" +
    "<ContentLink xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\" />" +
    "<Content xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\">" +
    "<![CDATA[This is a first paragraph!<DIV> </DIV>And this is a second
paragraph.]]></Content>" +
    "<PartStorage xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\"
/></WebPart>";

WebPartDefinition oWebPartDefinition = limitedWebPartManager.ImportWebPart(xmlWebPart);

limitedWebPartManager.AddWebPart(oWebPartDefinition.WebPart, "Left", 1);

oClientContext.ExecuteQuery();
```

Web part. Deleting a Web Part from a page

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile =
oClientContext.Web.GetFileByServerRelativeUrl("/sites/MySiteCollection/SitePages/Home.aspx ");
```

```
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

oClientContext.Load(limitedWebPartManager.WebParts);

oClientContext.ExecuteQuery();

if (limitedWebPartManager.WebParts.Count == 0)
{
    throw new Exception("No Web Parts to delete.");
}

WebPartDefinition webPartDefinition = limitedWebPartManager.WebParts[0];

webPartDefinition.DeleteWebPart();

oClientContext.ExecuteQuery();
```

Context. Using a credential cache for elevated execution of code

While server side-code can run with elevated privileges, there is not an equivalent method to elevate privileges in client-side code (for obvious security reasons). As an alternative, you can specify credentials to emulate the access of a specific user or service account.

To specify credentials, build and populate a **`CredentialCache`** object, then assign it to your `ClientContext` object's `Credentials` property.

The example below emulates the application pool account, and assumes an on-premises SharePoint 2013 environment with NTLM.

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

using (ClientContext ctx = new ClientContext("https://onpremises.local/sites/demo/"))
{
    // need the web object
    ctx.Load(ctx.Web);
    ctx.ExecuteQuery();

    // here the default network credentials relate to the identity of the account
    // running the App Pool of your web application.
    CredentialCache credCache = new CredentialCache();
    cc.Add(new Uri(ctx.Web.Url), "NTLM", CredentialCache.DefaultNetworkCredentials);

    ctx.Credentials = credCache;
    ctx.AuthenticationMode = ClientAuthentication.Default;
    ctx.ExecuteQuery();

    // do stuff as elevated app pool account
}
```

Note that granting the application pool account elevated privileges in SharePoint is against best practice, but that any relevant network credentials could be used in its place.

Read Working with Managed Client Side Object Model (CSOM) online:

https://riptutorial.com/sharepoint/topic/2679/working-with-managed-client-side-object-model--csom-

Chapter 9: Working with Managed Server Side Object Model (full-trust)

Remarks

Conceptual Hierarchy

In the SharePoint conceptual hierarchy, **site collections** contain **sites**, which in turn contain **lists**. A site collection (`SPSite`) has no explicit UI but always contains one root level site (accessible through the `RootWeb` property) and possibly additional subsites under that root site. A site or web (`SPWeb`) has a UI and contains lists/document libraries (`SPList`), pages with webparts, and items/documents (`SPListItem`).

Server-Side Caveats

- To create an application that uses the SharePoint server-side object model, in your Visual Studio project you must add a reference to the Microsoft.SharePoint assembly which is listed under Framework Assemblies.
- Applications using the Server Side Object Model (full-trust) can run only on a Windows Server that is hosting SharePoint.
- You cannot connect to a SharePoint server other than the one the application is running on.

Examples

Hello World (getting site title)

2013

SharePoint 2013 and newer versions are 64-bit only and so the assembly/program needs to be also built for 64-bit processor.

Right after your project is created it is necessary to swich the **Platform target** from **Any CPU** to **x64** otherwise error will occure.

```
using System;
using Microsoft.SharePoint;

namespace StackOverflow
{
    class Samples
    {
        static void Main()
        {
            using (SPSite site = new SPSite("http://server/sites/siteCollection"))
```

```
            using (SPWeb web = site.OpenWeb())
            {
                Console.WriteLine("Title: {0} Description: {1}", web.Title, web.Description);
            }
        }
    }
}
```

Looping through entire SharePoint farm

Using PowerShell executed from a SharePoint Web Server:

```
$wacoll = get-spwebapplication
foreach($wa in $wacoll){
    if($wa.IsAdministrationWebApplication -eq $false){
        foreach($site in $wa.Sites){
            foreach($web in $site.AllWebs){
                # your code here
                $web.Dispose()
            }
            $site.Dispose()
        }
    }
}
```

Retrieve list items

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Some list"];

    // It is always better and faster to query list items with GetItems method with
    // empty SPQuery object than to use Items property
    SPListItemCollection items = list.GetItems(new SPQuery());
    foreach (SPListItem item in items)
    {
        // Do some operation with item
    }
}
```

Retrieve items using paging

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Some list"];
    SPQuery query = new SPQuery()
    {
        RowLimit = 100
    };

    do
    {
        SPListItemCollection items = list.GetItems(query);
```

```
        foreach (SPListItem item in items)
        {
            // Do some operation with item
        }

        // Assign current position to SPQuery object
        query.ListItemCollectionPosition = items.ListItemCollectionPosition;
    } while (query.ListItemCollectionPosition != null);
}
```

Get list by url

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    string listUrl = string.Format("{0}{1}", web.ServerRelativeUrl, "Lists/SomeList");
    SPList list = web.GetList(listUrl);
}
```

Creating a list item

When creating a new list item, its fields can be set using syntax similar to string arrays. Note that these fields are not created on the fly and are defined by the schema of the list. These fields (or columns) must exist on the server otherwise the create will fail. All list items will have the Title field. Some lists may have required fields that must be filled out before the item will be published in the list.

In this example, the list is using the Announcements template. In addition to the title field, the list includes the Body field that will display the contents of the announcement on the list.

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Announcements"];

    SPListItem item = list.AddItem();
    item[SPBuiltInFieldId.Title] = "My new item";
    item[SPBuiltInFieldId.Body] = "Hello World!";
    item.Update();
}
```

Read Working with Managed Server Side Object Model (full-trust) online: https://riptutorial.com/sharepoint/topic/7543/working-with-managed-server-side-object-model--full-trust-

Chapter 10: Working with Modal Dialog Boxes with JavaScript

Syntax

- var options = SP.UI.$create_DialogOptions();
- var modalDialog = SP.UI.ModalDialog.showModalDialog(options);

Parameters

options Property	Description
title	A string that contains the title of the dialog
url	A string that contains the URL of the page that appears in the dialog. Either **url** or **html** must be specified. **url** takes precedence over **html**.
html	An HTML element to display within the dialog.
x	The x-offset of the dialog as an integer value.
y	The y-offset of the dialog as an integer value.
width	The width of the dialog as an integer value. If unspecified and **autosize** is **false** the width is set to 768px
height	The height of the dialog as an integer value. If unspecified and **autosize** is **false** the height is set to 576px
allowMaximize	A Boolean value specifying whether the **Maximize** button should be shown.
showMaximized	A Boolean value specifying whether the dialog opens maximized.
showClose	A Boolean value specifying whether the **Close** button appears on the dialog.
autoSize	A Boolean value that specifies whether the dialog platform handles dialog sizing automatically.
dialogReturnValueCallback	A function pointer that specifies the return callback function. Function takes two parameters: a *dialogResult* of type SP.UI.DialogResult Enumeration, and a *returnValue* object that

options Property	Description
	contains any data returned by the dialog.
args	An object that contains data that are passed to the dialog.

Remarks

The `SP.UI.ModalDialog` namespace was introduced to the JavaScript Object Model with SharePoint 2010, and is available in subsequent SharePoint versions 2013, Office365, and 2016.

Additional reference materials:

- MSDN Reference for SP.UI.ModalDialog.showModalDialog(options)
- MSDN Reference for SP.UI.DialogResult Enumeration

Examples

Perform an Action when a Dialog Box is Closed

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog,"sp.js");

function showDialog(){
    var options = SP.UI.$create_DialogOptions();
    options.url = "/mySite/lists/myList/NewForm.aspx";
    options.dialogReturnValueCallback = myCallBackFunction;
    SP.UI.ModalDialog.showModalDialog(options);
    function myCallBackFunction(result,data){
        switch(result){
            case SP.UI.DialogResult.invalid:
                alert("The dialog result was invalid");
                break;
            case SP.UI.DialogResult.cancel:
                alert("You clicked cancel or close");
                break;
            case SP.UI.DialogResult.OK:
                alert("You clicked OK, creating an item in the list.");
                break;
        }
    }
}
```

Show an Existing Page in a Dialog

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog,"sp.js");

function showDialog(){
    SP.UI.ModalDialog.showModalDialog(
        { url: "/org/it/web/wik/Lists/ExampleCode/DispForm.aspx?ID=6" }
    );
}
```

Show a Custom Dialog

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog,"sp.js");

function showDialog(){
    var dialogOptions = SP.UI.$create_DialogOptions();
    dialogOptions.title = "Your Title Here!";
    var dummyElement = document.createElement("div");
    dummyElement.style.textAlign = "center";
    dummyElement.appendChild(document.createElement("br"));
    dummyElement.appendChild(document.createTextNode("Some beautifully crafted text."));
    dummyElement.appendChild(document.createElement("br"));
    dialogOptions.html = dummyElement;
    SP.UI.ModalDialog.showModalDialog(dialogOptions);
}
```

Read Working with Modal Dialog Boxes with JavaScript online:
https://riptutorial.com/sharepoint/topic/6868/working-with-modal-dialog-boxes-with-javascript

Credits

S. No	Chapters	Contributors
1	Getting started with sharepoint	Community, Marco, Ryan Gregg, Thriggle, Tom Resing, Zach Koehne
2	Creating a provider hosted App	vinayak hegde
3	Major Releases	jjr2527, MikhailSP
4	REST Services	Aaron, Brock Davis, ocelotsloth, R4mbi, Rohit Waghela, Thriggle
5	SharePoint 2013 Client Side Rendering	Rohit Waghela, Yayati
6	SharePoint App	Sunil sahu
7	Working with JavaScript Client Object Model (JSOM)	Thriggle, yngrdyn
8	Working with Managed Client Side Object Model (CSOM)	InvoiceGuy, Lukáš Nešpor, MikhailSP, RamenChef, Thriggle, Zach Koehne
9	Working with Managed Server Side Object Model (full-trust)	Lukáš Nešpor, Thriggle
10	Working with Modal Dialog Boxes with JavaScript	Thriggle