



SharePoint 2013 Client Side Object Model Cookbook

101 Recipes

Vijai Anand

I would like to dedicate this eBook to my parents and my friends.

- Vijai Anand. R

TABLE OF CONTENTS

1	SharePoint 2013 .Net Client Side Object Model: An Overview	11
2	Create a console application using Visual Studio 2012	14
3	Perform SharePoint list tasks using CSOM.....	19
3.1	How to get all the lists from the website.....	19
3.2	How to create a new list in the website	20
3.3	How to delete a list from the website	22
3.4	How to update a list in the website	23
3.5	How to enable folder creation for the list in the website	24
3.6	How to disable attachments to list items in the website	26
3.7	How to display the list in the quick launch bar	27
3.8	How to enable versioning for the list.....	29
3.9	How to enable minor versions for the document library	31
3.10	How to enable force check out for the document library	33
3.11	How to enable content approval for the list.....	35
3.12	How to specify the permission required to view minor versions and drafts within the list.....	36
3.13	How to get all the list templates available for creating lists.....	38
4	Perform SharePoint website tasks using CSOM.....	41
4.1	How to get the properties of a website	41
4.2	How to update the properties of a website.....	42
4.3	How to get only specific properties of a website	43
4.4	How to delete a website	45
4.5	How to get all available web templates from website	46
4.6	How to get all the active features from website	48
5	Perform SharePoint list item tasks using CSOM	50
5.1	How to get all the items from the list	50
5.2	How to create a new item in the list.....	51
5.3	How to update an item in the list	53
5.4	How to delete an item in the list	54
5.5	How to get the items from a list folder.....	55

5.6	How to get the limited number of items from the list	57
5.7	How to get all the attachments for the list item	58
5.8	How to delete an attachment for the list item	60
6	Perform SharePoint content type tasks using CSOM	62
6.1	How to get all the content types from the website.....	62
6.2	How to create a site content type	63
6.3	How to delete the site content type	66
6.4	How to set the site content type read only	67
6.5	How to get all the content types from the list.....	68
6.6	How to delete the content type from the list.....	70
6.7	How to add existing content type to the list	71
7	Perform SharePoint field tasks using CSOM	73
7.1	How to get all the fields from the list	73
7.2	How to update a specific field available in the list	74
7.3	How to add a field in the list	75
7.4	How to add an existing field to the list	77
7.5	How to delete a field from the list	78
7.6	How to set the default value for the list field	79
7.7	How to get the calculated field formula	80
7.8	How to set the formula for the calculated field.....	82
8	Perform SharePoint list view tasks using CSOM	84
8.1	How to get all the views for the list	84
8.2	How to create a new list view	85
8.3	How to get all the fields available in the list view.....	86
8.4	How to set the default view in the list.....	88
8.5	How to add a field to the list view	89
8.6	How to delete a field from the list view.....	90
8.7	How to delete a list view.....	91
9	Perform SharePoint folder tasks using CSOM.....	94
9.1	How to get all the top level folders from the website	94
9.2	How to get all the top level folders from the list	95
9.3	How to get the subfolders from the list.....	97

9.4	How to delete a folder from the list	98
9.5	How to create a new folder in the document library	99
9.6	How to get the number of items inside the folder	100
10	Perform SharePoint file tasks using CSOM	103
10.1	How to get the major version of the file.....	103
10.2	How to get the minor version of the file.....	104
10.3	How to check out the file in the document library	106
10.4	How to get the user login name who has checked out the file	107
10.5	How to get the user login name who added the file	108
10.6	How to get the check out type associated with the file	110
10.7	How to check in the file	111
10.8	How to get the check in comment of the file	113
10.9	How to unpublish the major version of the file	114
10.10	How to discard check out of the file	115
10.11	How to delete the file from the document library	117
11	Perform SharePoint file version tasks using CSOM	119
11.1	How to get all the versions for the file	119
11.2	How to get the file version for the document by version Id.....	121
11.3	How to delete a file version by version ID for the document.....	122
11.4	How to delete a file version by version label for the document	124
11.5	How to restore a specific file version for the document	126
11.6	How to check if the file version is a current version for the document	127
11.7	How to delete all the file versions for the document.....	129
12	Perform SharePoint group tasks using CSOM.....	131
12.1	How to get all the site groups	131
12.2	How to create a new site group.....	132
12.3	How to set the user as owner for the site group.....	133
12.4	How to set the group as owner for the site group	135
12.5	How to get all the users from the site group.....	137
12.6	How to add a user to the site group	138
12.7	How to remove a user from the site group	140
12.8	How to delete a site group.....	141

13	Perform SharePoint role tasks using CSOM	143
13.1	How to get all the permission levels from the website	143
13.2	How to create a permission level in the website.....	144
13.3	How to update the permission level in the website	146
13.4	How to remove the permissions from the permission level.....	149
13.5	How to delete the permission level from the website	150
13.6	How to assign the role to a specific group	152
13.7	How to assign the role to a specific user	153
14	Perform SharePoint Taxonomy related tasks using CSOM	156
14.1	How to get all the Term Stores for the provided site	156
14.2	How to get the Default site collection Termstore	157
14.3	How to get the default keyword termstore	159
14.4	How to get all the groups for the termstore.....	160
14.5	How to create a new group for the term store	161
14.6	How to delete the group from the term store.....	163
14.7	How to get all the termsets for the taxonomy group	164
14.8	How to create a term set for the specified group	166
14.9	How to delete the term set from the specified group.....	167
14.10	How to get all the terms for the termset.....	169
14.11	How to create a new term for the termset.....	170
14.12	How to delete the term from the term set.....	172
14.13	How to create a copy of the term within the termset.....	173
14.14	How to move the specified term to different termset	175
14.15	How to deprecate the specified term	176
14.16	How to get all the labels for specified term.....	178
14.17	How to create a new label for specified term	180
14.18	How to delete the label for specified term.....	182
15	Perform SharePoint User Profile tasks using CSOM	184
15.1	How to get a particular user's properties	184
15.2	How to get the current user's properties	185
15.3	How to get the specific user profile property for a user	186
16	Perform SharePoint social tasks using CSOM	189

16.1	How to check the specified user is following the target user.....	189
16.2	How to get the count of people the user is following	190
16.3	How to get the count of documents followed by the user.....	192
16.4	How to get the count of sites followed by the user.....	193
16.5	How to get the count of actors followed by the user	195
16.6	How to get all the followers for the user	196
16.7	How to get all the people followed by the user	198
16.8	How to get all the documents followed by the user.....	200
16.9	How to get all the sites followed by the user	201
16.10	How to follow the target user.....	203
16.11	How to stop following the target user	205
16.12	How to follow the site.....	206
16.13	How to stop following the site.....	209
16.14	How to follow the document.....	210
16.15	How to stop following the document	212
Summary:.....		215

About the Author:

Vijai Anand has been working in the IT industry for over 4 years. He holds a Bachelor's degree in Electronics and Communication Engineering. He works as a SharePoint Developer at Cognizant Technology Solutions, Bangalore. Vijai has worked on Microsoft Office SharePoint® Server 2007, Microsoft SharePoint® 2010 and Microsoft SharePoint® 2013.

Vijai is a frequent contributor on C# Corner (www.c-sharpcorner.com). He has authored around 400 articles and 360 blogs on www.csharpcorner.com regarding SharePoint 2013, SharePoint 2010, SharePoint Workspace, SharePoint Designer 2010, Powershell, C # and Silverlight. He currently holds **Microsoft Most Valuable Professional (MVP)** and **Mindcracker Most Valuable Professional** awards for his SharePoint Server contributions.

He has authored the following **eBooks**:

- *Getting Started with Managed Metadata Service in SharePoint 2010* published in CSHARPCORNER.com
- *Business Data Connectivity Services - Step by Step tutorial* published in ITFUNDA.com

He has accomplished the following **Microsoft Certifications**:

- Microsoft SharePoint® 2010, Application Development
- Microsoft SharePoint® 2010, Designing and Developing Microsoft SharePoint 2010 Applications.
- Microsoft Office SharePoint® Server 2007, Application Development
- Microsoft Office SharePoint® Server 2007, Configuration

Who can read this book?

SharePoint Developers who have some basic knowledge about SharePoint 2010 or 2013 will find this book helpful for understanding and working with the .Net Client Side Object Model. This book is mainly focused on beginner functionality and code samples to perform basic operations using the .Net Client Side Object Model. For advanced developers, sections 14, 15 and 16 will be more useful that explain the operations that can be performed by the new assemblies added in the SharePoint 2013 Client Side Object Model. With respect to the code samples in this book, you should be familiar with SharePoint Object Model and out-of-the-box features.

Acknowledgment:

I am really thankful to each and everyone who motivated me to write articles and publish my third eBook. I would like to express my special thanks to **Mahesh Chand (Microsoft MVP, Founder of Mindcracker Networks)** and to the entire CSharpcorner team for motivating me to publish my third eBook. I am really thankful to **Sam Hobbs (C# Corner Editor)** for reviewing and editing my eBook.

I would like to thank **Satveer Khurpa (SharePoint Lead Architect)** who did technical review for this eBook. Thanks to all the reviewers for reviewing my eBook.

I would like to express my special thanks to **M.Kavya** who designed the cover page of this eBook.

I would like to express my thanks to all my colleagues and Architects who supported me to write this book. Thanks to all my friends who helped me to publish this eBook.

1 SharePoint 2013 .Net Client Side Object Model: An Overview

The

Client Side Object Model (CSOM) was first introduced in SharePoint 2010. The Client Side Object Model is mainly used to build client applications and enable us to access SharePoint Sites that are hosted outside without using webservice. Prior to the CSOM, developers had only a few choices to build client applications. However, this has changed now. With the introduction of CSOM, developers now have more choices and will be able to access the core SharePoint functionalities to a wide extent. In SharePoint 2010, the CSOM exposed the core SharePoint functionalities only whereas in SharePoint 2013, the Microsoft SharePoint team has added a few more assemblies. Now we are able to access the Service Applications using the Client Side Object Model.

There are three programming models used for the Client Side Object Model; they are:

- .Net Client Side Object Model.
- Javascript Object Model.
- Silverlight Object Model.

Assemblies for the above models are located in the SharePoint file system. The .Net Client Side Object Model is located at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\15\ISAPI. The Silverlight Object Model is located at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\15\TEMPLATE\LAYOUTS\ClientBin. ECMAScripts are located at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\15\TEMPLATE\LAYOUTS.

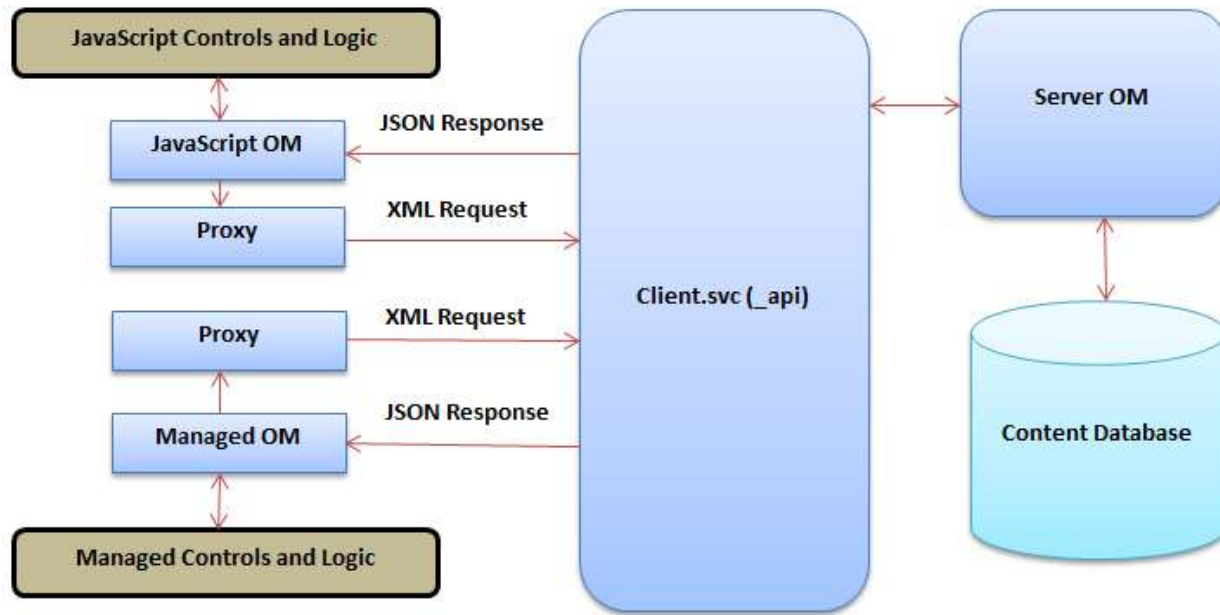


Figure1.1: Client Side Object Model Architecture

How does it work? If you use the Client Side API to perform a specific task then the SharePoint .Net client object model bundles up these uses of the API into XML and send them to the server. The server receives this request, and makes appropriate calls into the object model on the server, collects the responses, forms them into JavaScript Object Notation (JSON), and sends that JSON back to the SharePoint .Net client object model. The client object model parses the JSON and presents the results to the application.

Please find the below table to know few client-side classes and server-side equivalents

Client-side classes and server-side equivalents	
Client	Server
ClientContext	SPContext
Site	SPSite
Web	SPWeb
List	SPList
ListItem	SPLListItem
Field	SPField

To build a basic operation using the .Net Client Side Object Model there are some very essential classes and methods that need to be used that are explained below.

ClientContext class: the ClientContext class gets the current context by passing the site URL, which inherits from the ClientContextRuntime class.

Load() method: the Load() method does not actually load anything, only when the ExecuteQuery() method is called does it provide notification that these are the property values that should be loaded for the object. In the Load() method lambda expressions can be used to specify which property should be loaded instead of loading all the properties.

ExecuteQuery() method: the ExecuteQuery() method sends the request to the server. There is no network traffic until the application calls this method. Until the ExecuteQuery() method is called only the requests are registered by the application.

In the later sections you will see the code samples to perform the basic operations using the SharePoint 2013 Client Side Object Model.

2 Create a console application using Visual Studio 2012

In this section you will see how to create a console application using Visual Studio 2012 which will be used to explain all the examples for the .Net Client Side Object Model.

Steps Involved:

1. Open Visual Studio 2012 (Run as administrator).
2. Go to "File" => "New" => "Project...".

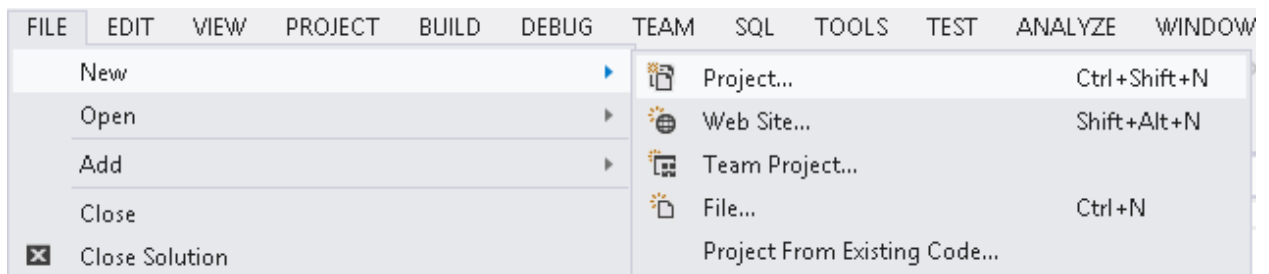


Figure2.1: Create a new project in Visual Studio 2012

3. Select "Console Application" in the Visual C# node from the installed templates.
4. Enter the Name as "CSOMCodeSamples" and click on "OK".

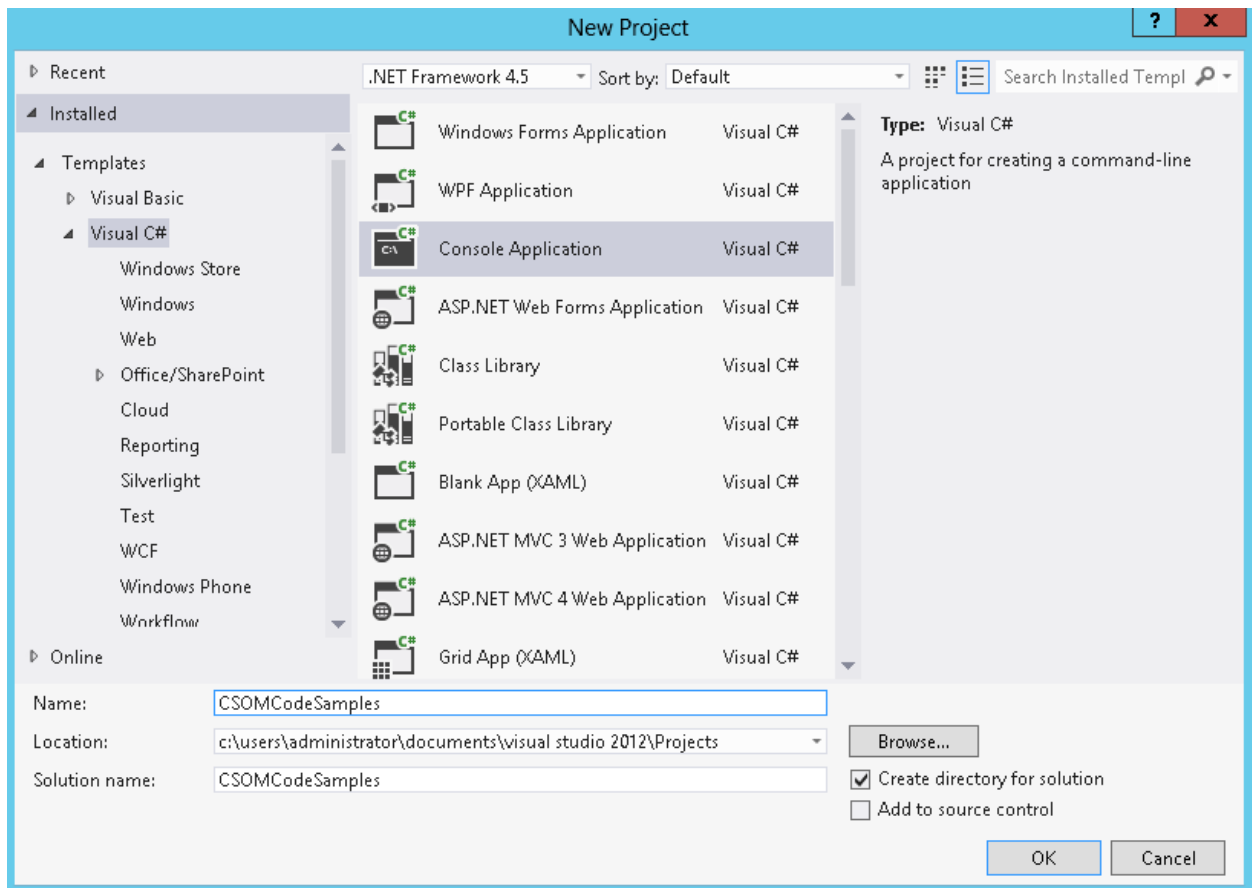


Figure2.2: Select the console application template

5. Right-click on the project and then click on "Properties". Click on the "Build" tab and ensure that the following options are selected properly.

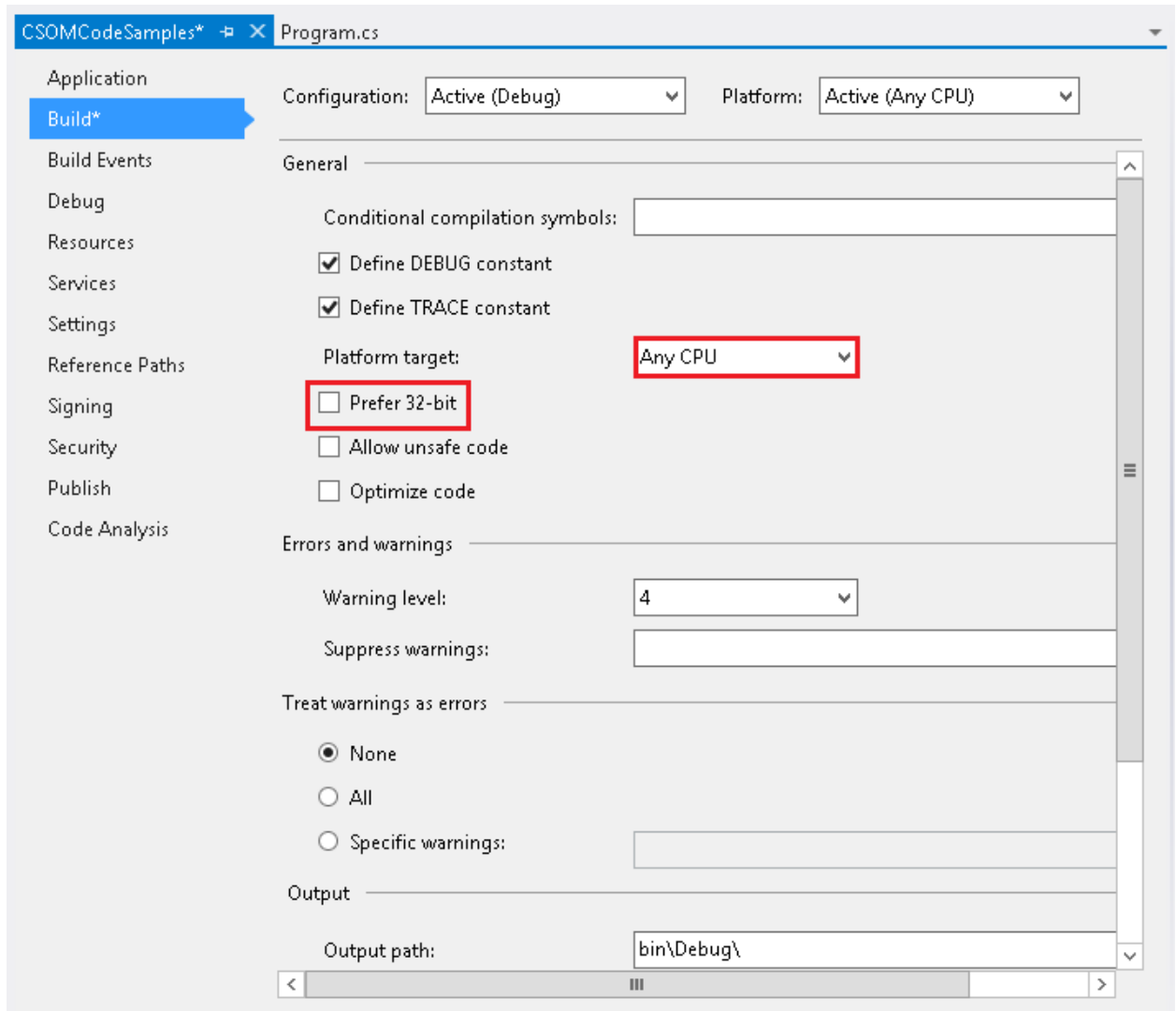


Figure2.3: Project properties

6. In the Solution Explorer, right-click on the "References" folder and then click on "Add Reference".

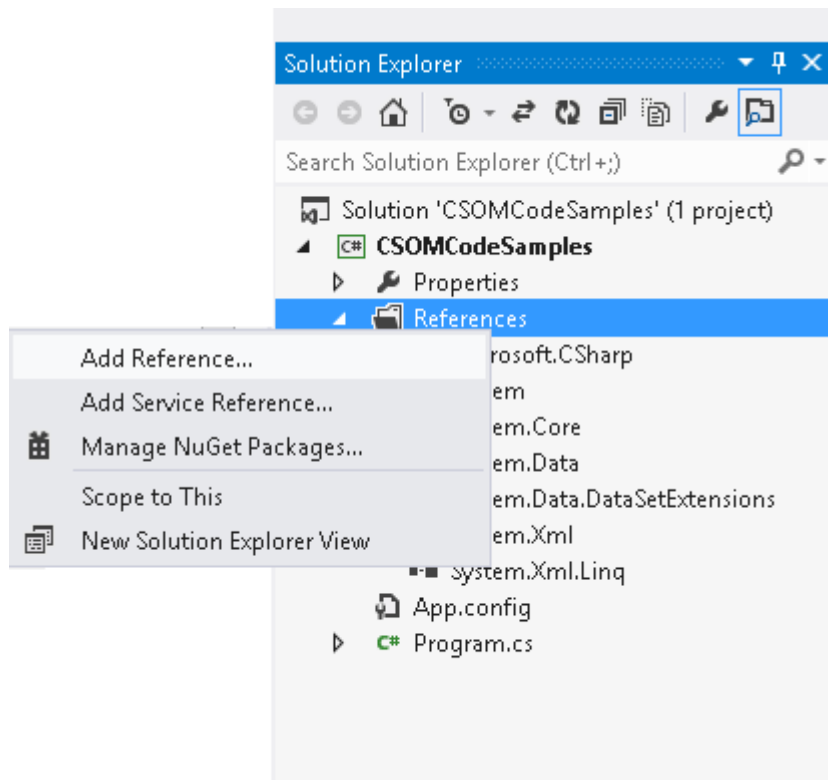


Figure2.4: Add reference

7. Add the following assemblies from hive 15 (C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\ISAPI).

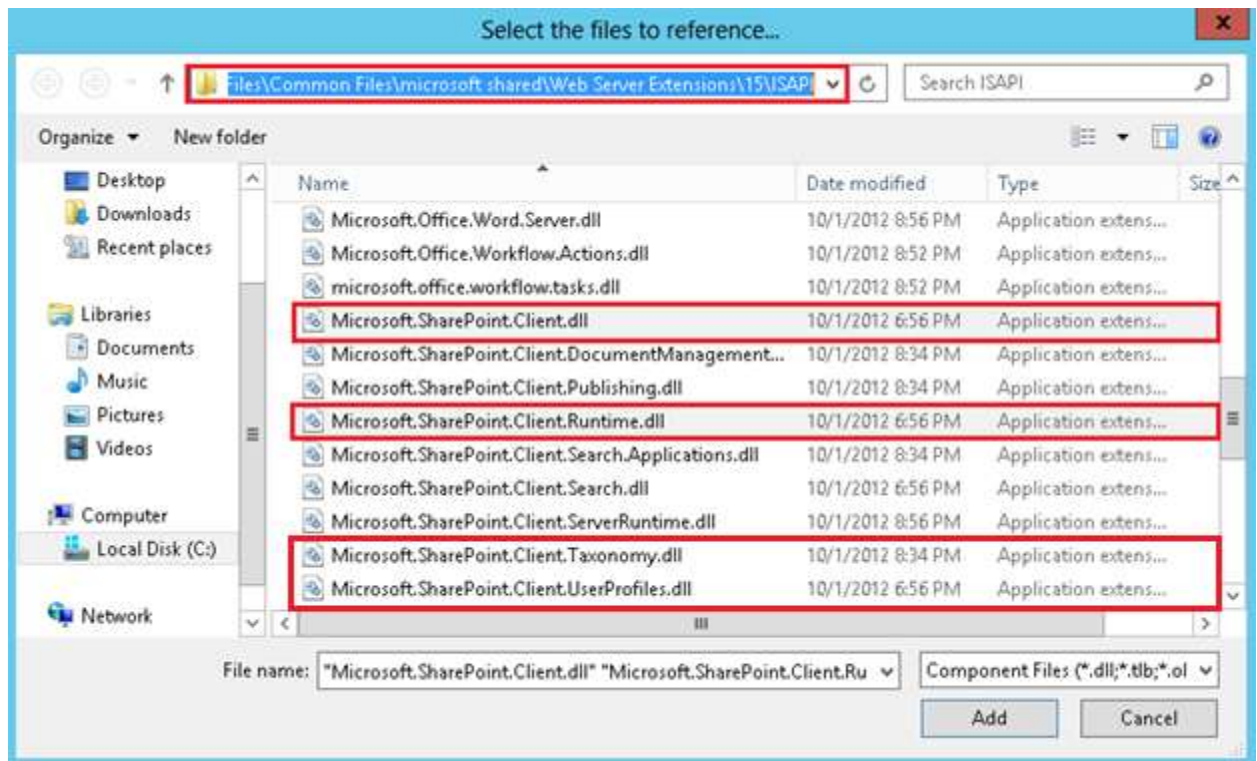


Figure2.5: Add the references

Microsoft.SharePoint.Client.Taxonomy.dll should be added to perform taxonomy related tasks and Microsoft.SharePoint.Client.UserProfiles.dll should be added to perform user profile related tasks.

Thus in this section you have seen how to create a console application using Visual Studio 2012.

3 Perform SharePoint list tasks using CSOM

In this section you will see how to perform list related tasks using the SharePoint 2013 .Net Client Side Object Model.

3.1 How to get all the lists from the website

In this example you will see how to get all the lists from the website using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list collection for the web
            ListCollection listColl = web.Lists;

            // Retrieve the list collection properties
            clientContext.Load(listColl);
        }
    }
}
```

```

// Execute the query to the server.
clientContext.ExecuteQuery();

// Loop through all the list
foreach (List list in listColl)
{
    // Display the list title and ID
    Console.WriteLine("List Name: " + list.Title + "; ID: " + list.Id);
}
Console.ReadLine();
}
}
}

```

Output

```

file:///c:/users/administrator/documents/visual studio 2012/Projects/CSOMCode...
List Name: appdata; ID: fe281284-293b-41f8-b63c-00a64c76269a
List Name: Composed Looks; ID: 6d39dc78-3125-44c0-be35-54cb80198ac5
List Name: Content type publishing error log; ID: b88686ba-f19d-4cd0-8358-8a07dc
d5dc65
List Name: Converted Forms; ID: fa388ec0-f309-4e7e-b8e4-1c19ad9ca265
List Name: Custom List; ID: a5070347-0ec5-4fbb-8343-b538befa74e0
List Name: Documents; ID: 35595a27-1f7d-42d9-9720-6478446c7d88
List Name: Form Templates; ID: 42b43a19-ec31-49de-b4d5-35395bf732e7
List Name: List Template Gallery; ID: f0c0284d-6ec3-4628-94b6-97ef9e108bd3
List Name: Master Page Gallery; ID: 26e7f9c7-5d08-48e0-953f-70b17cc10aba
List Name: MicroFeed; ID: f5c416dc-3c9d-42d9-b215-f9eb83a312a7
List Name: Project Policy Item List; ID: 24e63912-320a-4be7-8178-5918d0fdd8da
List Name: Site Assets; ID: fa4f8238-c5f5-408e-b076-c159b992ad78
List Name: Site Pages; ID: 593f5446-4924-400e-b33f-c26972a0f776
List Name: Solution Gallery; ID: 3edf8f54-d02d-454d-b3fd-4b22df59d542
List Name: Style Library; ID: 9d9c2efa-24e6-45f0-bc9a-2c73b2c707b6
List Name: TaxonomyHiddenList; ID: e2b64098-9a72-4664-bbfa-cdac9b487e06
List Name: Theme Gallery; ID: 1ec42f24-db7d-4f32-a694-abce390c7038
List Name: User Information List; ID: 7917c76f-4006-4544-8f2d-ef5bb5583ad5
List Name: Web Part Gallery; ID: 96b7550b-5d1c-4edf-ab58-5d4dd6663afc
List Name: wfpub; ID: 17b8d646-75be-47b0-9181-5e93999c1950

```

Figure3.1.1: Get all the lists from the list

3.2 How to create a new list in the website

In this example you will see how to create a new list in the website using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Specifies the properties of the new custom list
            ListCreationInformation creationInfo = new ListCreationInformation();
            creationInfo.Title = "Custom List";
            creationInfo.Description = "Custom list created using CSOM";
            creationInfo.TemplateType = (int)ListTemplateType.GenericList;

            // Create a new custom list
            List newList=clientContext.Web.Lists.Add(creationInfo);

            // Retrieve the custom list properties
            clientContext.Load(newList);

            // Execute the query to the server.
            clientContext.ExecuteQuery();

            // Display the custom list Title property
            Console.WriteLine(newList.Title);
            Console.ReadLine();
        }
    }
}

```

Output

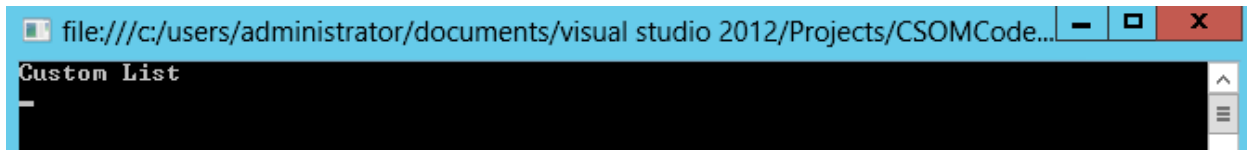


Figure 3.2.1: Create a new list

3.3 How to delete a list from the website

In this example you will see how to delete a list from the website using the Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by Title
            List list = web.Lists.GetByTitle("Custom");

            // Delete the list object
            list.DeleteObject();
        }
    }
}
```



```

        // Execute the query to the server.
        clientContext.ExecuteQuery();
    }
}
}

```

Output

The Custom list is deleted successfully.

3.4 How to update a list in the website

In this example you will see how to update a list in the website using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

```

```

// Update the description for the custom list
list.Description = "Custom list description updated using CSOM";

// Update the list
list.Update();

// Retrieve the list properties
clientContext.Load(list);

// Execute the query to the server.
clientContext.ExecuteQuery();

// Display the list title and description
Console.WriteLine("List Title: " + list.Title + "; Description: " +
list.Description);
Console.ReadLine();
    }
}
}

```

Output

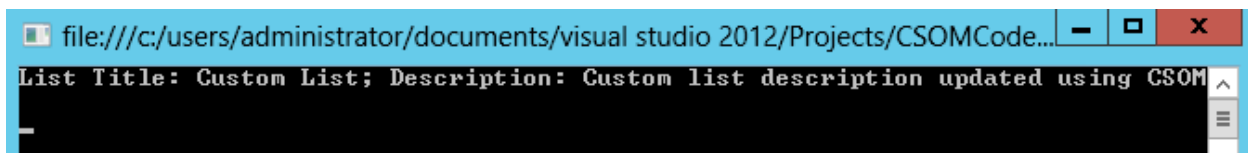


Figure 3.4.1: Update the list

3.5 How to enable folder creation for the list in the website

In this example you will see how to enable folder creation for the list in the website using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Enable folder creation for the custom list
            list.EnableFolderCreation = true;

            // Update the list
            list.Update();

            // Execute the query to the server.
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Custom List** link in the quick launch bar. Click on the **List Settings** button in the ribbon interface. Click on the **Advanced Settings** link which is available under the **General Settings** section. You will be able to see that the folder creation for the list has enabled successfully.

Folders

Specify whether the "New Folder" command is available. Changing this setting does not affect existing folders.

Make "New Folder" command available?

☒ Yes ☐ No

Figure3.5.1: Enable folder creation for the list**Reference**

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.list.enablefoldercreation.aspx>

3.6 How to disable attachments to list items in the website

In this example you will see how to disable attachments to list items in the website using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Disable attachments to list items
            list.EnableAttachments = false;
        }
    }
}
```

```

        // Update the list
        list.Update();

        // Execute the query to the server.
        clientContext.ExecuteQuery();
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Custom List** link in the quick launch bar. Click on the **List Settings** button in the ribbon interface. Click on the **Advance Settings** link which is available under the **General Settings** section. You will be able to see that the attachments to list items are disabled successfully.



Figure 3.6.1: Disable attachments to list items

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.list.enableattachments.aspx>

3.7 How to display the list in the quick launch bar

In this example you will see how to display the list in the quick launch bar using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;

```



```
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Display the list in the quick launch bar
            list.OnQuickLaunch = true;

            // Update the list
            list.Update();

            // Execute the query to the server.
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

Navigate to the SharePoint site. Click on the **Custom List** link in the quick launch bar. Click on the **List Settings** button in the ribbon interface. Click on the **List name, description and navigation** link which is available under the **General Settings** section. You will be able to see that the option to display the list in the quick launch has been enabled successfully.

CSOM

 EDIT LINKS

Settings ▸ General Settings

Name and Description

Type a new name as you want it to appear in headings and links throughout the site. Type descriptive text that will help site visitors use this list.

Name:

Description:

Navigation

Specify whether a link to this list appears in the Quick Launch. Note: it only appears if Quick Launch is used for navigation on your site.



Display this list on the Quick Launch?

☒ Yes ☐ No

Figure 3.7.1: Display the list on the quick launch

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.list.onquicklaunch.aspx>

3.8 How to enable versioning for the list

In this example you will see how to enable versioning for the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Enable versioning for the list
            list.EnableVersioning = true;

            // Update the list
            list.Update();

            // Execute the query to the server.
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

Navigate to the SharePoint site. Click on the **Custom List** link in the quick launch bar. Click on the **List Settings** button in the ribbon interface. Click on the **Version Settings** link which is available under the **Settings** section. You will be able to see that versioning the settings for the list has been enabled successfully.

Item Version History

Specify whether a version is created each time you edit an item in this list.

[Learn about versions.](#)

Create a version each time you edit an item in this list?

☒ Yes ☐ No

Optionally limit the number of versions to retain:

☐ Keep the following number of versions:

☐ Keep drafts for the following number of approved versions:

Figure 3.8.1: Enable versioning for the list

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.list.enableversioning.aspx>

3.9 How to enable minor versions for the document library

In this example you will see how to enable minor versions for the document library using the .Net Client Side Object Model.

Build the project using the following:

- [Create a console application.](#)
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
```

```

ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

// Get the SharePoint web
Web web = clientContext.Web;

// Get the SharePoint list by title
List list = web.Lists.GetByTitle("Documents");

// Enable minor versions for the document library
list.EnableMinorVersions = true;

// Update the list
list.Update();

// Execute the query to the server.
clientContext.ExecuteQuery();
    }
}
}

```

Output

Navigate to the SharePoint site. Click on the **Documents** link in the quick launch bar. Click on the **Library Settings** button in the ribbon interface. Click on the **Versioning Settings** link which is available under the **Settings** section. You will be able to see that the minor versions for the document library has been enabled successfully.

Document Version History

Specify whether a version is created each time you edit a file in this document library. [Learn about versions.](#)

Create a version each time you edit a file in this document library?

- ☐ No versioning
- ☐ Create major versions
Example: 1, 2, 3, 4
- ☒ Create major and minor (draft) versions
Example: 1.0, 1.1, 1.2, 2.0

Optionally limit the number of versions to retain:

- ☐ Keep the following number of major versions:
- ☐ Keep drafts for the following number of major versions:

Figure 3.9.1: Enable minor versions for the list

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.list.enableminorversions.aspx>

Note

If you try to run the same code for a custom list then you will get the error as shown in Figure 3.9.2. This is because there is no option to enable minor versions in the custom list.

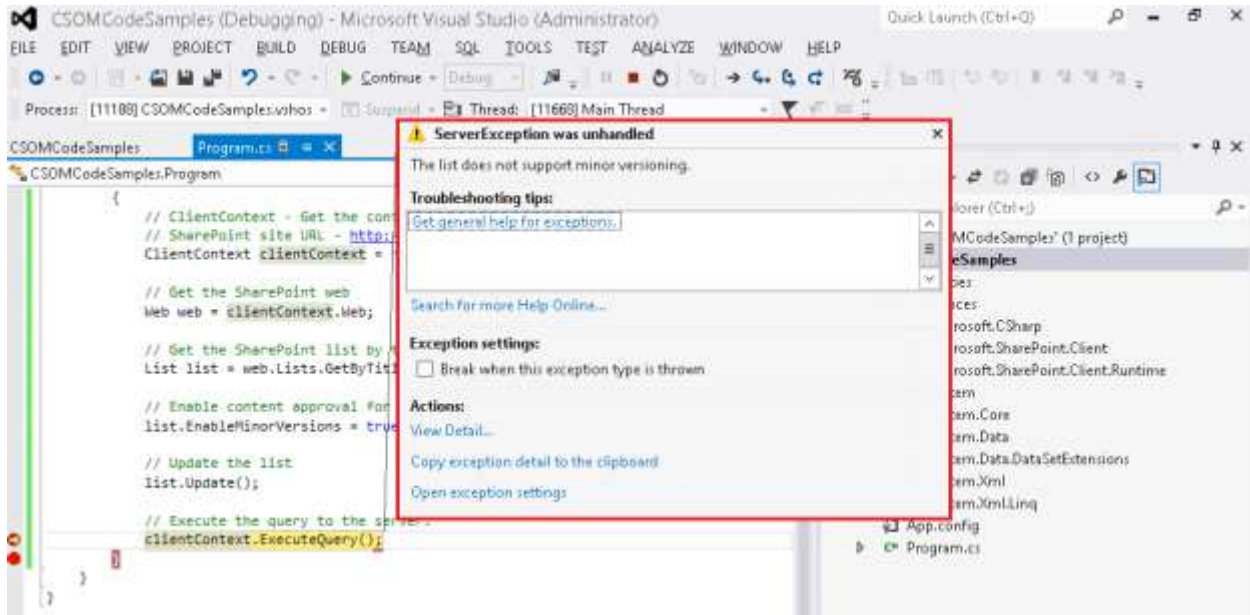


Figure 3.9.2: Error message

3.10 How to enable force check out for the document library

In this example you will see how to enable forced check out for the document library using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;

```



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Documents");

            // Enable force check out for document library
            list.ForceCheckout = true;

            // Update the list
            list.Update();

            // Execute the query to the server.
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Documents** link in the quick launch bar. Click on the **Library Settings** button in the ribbon interface. Click on the **Versioning Settings** link which is available under the **Settings** section. You will be able to see that the forced check out for the document library has been enabled successfully.

Require Check Out

Specify whether users must check out documents before making changes in this document library.
[Learn about requiring check out.](#)

Require documents to be checked out before they can be edited?

☒ Yes ☐ No

Figure3.10.1: Enable force check out for the list**Reference**

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.list.forcecheckout%28v=office.14%29.aspx>

3.11 How to enable content approval for the list

In this example you will see how to enable content approval for the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Enable content approval for the list
            list.EnableModeration = true;
        }
    }
}
```

```

        // Update the list
        list.Update();

        // Execute the query to the server.
        clientContext.ExecuteQuery();
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Custom List** link in the quick launch bar. Click on the **Library Settings** button in the ribbon interface. Click on the **Versioning Settings** link which is available under the **General Settings** section.

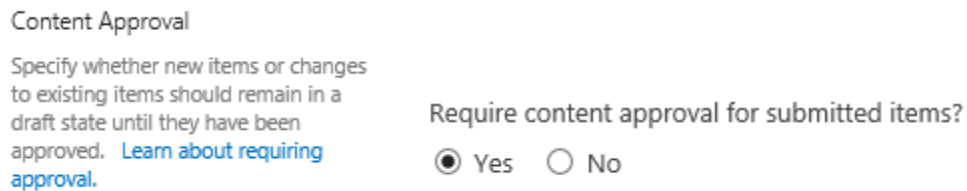


Figure3.11.1: Enable content approval for the list

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.list.draftversionvisibility.aspx>

3.12 How to specify the permission required to view minor versions and drafts within the list

In this example you will see how to specify the permission required viewing minor versions and drafts within the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the SharePoint list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Specify the permission required to view minor versions and drafts within
the list.
            // You can select the Draft Visibility Type as either Author, Approve or Read
            list.DraftVersionVisibility = DraftVisibilityType.Approver;

            // Update the list
            list.Update();

            // Execute the query to the server.
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Custom List** link in the quick launch bar. Click on the **Library Settings** button in the ribbon interface. Click on the **Versioning Settings** link which is available under **General Settings** section.

Draft Item Security

Drafts are minor versions or items which have not been approved. Specify which users should be able to view drafts in this list. [Learn about specifying who can view and edit drafts.](#)

Who should see draft items in this list?

- ☐ Any user who can read items
☐ Only users who can edit items
☒ Only users who can approve items (and the author of the item)

Figure 3.13.1: Specify which users should be able to view drafts in this list

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.list.draftversionvisibility.aspx>

3.13 How to get all the list templates available for creating lists

In this example you will see how to get all the list templates available for creating lists using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");
```

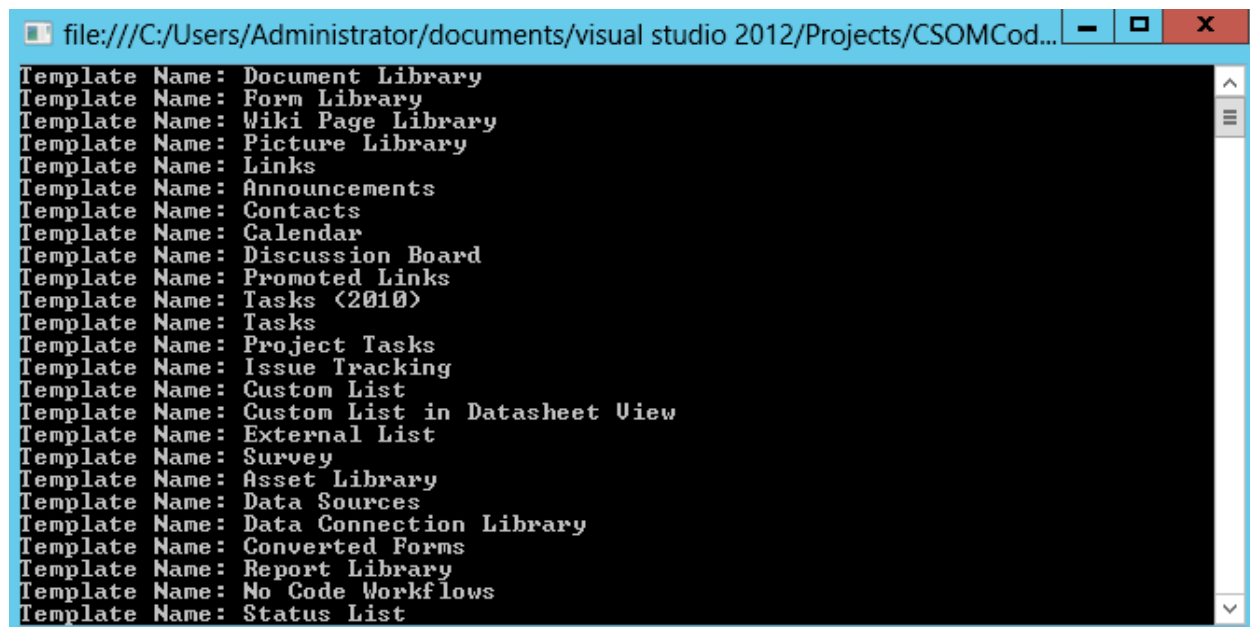
```
// Get the SharePoint web
Web web = clientContext.Web;

// Get all the list templates available for creating lists on the site
ListTemplateCollection templateColl = web.ListTemplates;
clientContext.Load(templateColl);

// Execute the query to the server
clientContext.ExecuteQuery();

// Loop through all the list templates
foreach (ListTemplate template in templateColl)
{
    // Display the list template name
    Console.WriteLine("Template Name: "+template.Name);
}
Console.ReadLine();
}
```

Output



```
file:///C:/Users/Administrator/documents/visual studio 2012/Projects/CSOMCod...
Template Name: Document Library
Template Name: Form Library
Template Name: Wiki Page Library
Template Name: Picture Library
Template Name: Links
Template Name: Announcements
Template Name: Contacts
Template Name: Calendar
Template Name: Discussion Board
Template Name: Promoted Links
Template Name: Tasks (2010)
Template Name: Tasks
Template Name: Project Tasks
Template Name: Issue Tracking
Template Name: Custom List
Template Name: Custom List in Datasheet View
Template Name: External List
Template Name: Survey
Template Name: Asset Library
Template Name: Data Sources
Template Name: Data Connection Library
Template Name: Converted Forms
Template Name: Report Library
Template Name: No Code Workflows
Template Name: Status List
```


Figure3.13.1: *Get all the available list templates*

4 Perform SharePoint website tasks using CSOM

In this section you will see how to perform website related tasks using the SharePoint 2013 .Net Client Side Object Model.

4.1 How to get the properties of a website

In this example you will see how to retrieve the website properties using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // CliContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Load the Web properties
            clientContext.Load(web);

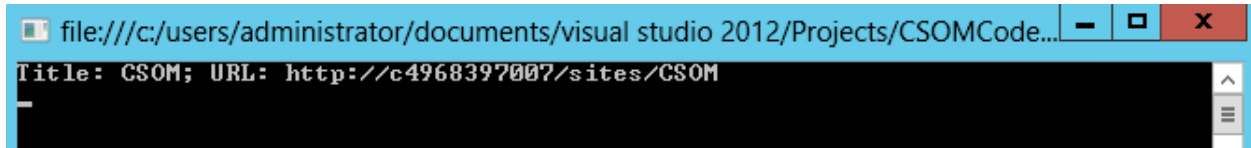
            // Execute the query to the server.
            clientContext.ExecuteQuery();
        }
    }
}
```

```

        // Web properties - Display the Title and URL for the web
        Console.WriteLine("Title: " + web.Title + "; URL: " + web.Url);
        Console.ReadLine();
    }
}

```

Output



Figur4.1.1: Properties of the website

4.2 How to update the properties of a website

In this example you will see how to update the website properties using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // CliContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new

```

```
ClientContext("http://servername/sites/CSOM");

// Get the SharePoint web
Web web = clientContext.Web;

// Update the web Title and Description properties
web.Title = "CSOM Updated";
web.Description = "Updated the web title using CSOM";
web.Update();

// Load the Web properties
clientContext.Load(web);

// Execute the query to the server.
clientContext.ExecuteQuery();

// Web properties - Display the Title and URL for the web
Console.WriteLine("Title: " + web.Title + "; Description: " +
web.Description);
Console.ReadLine();
    }
}
```

Output

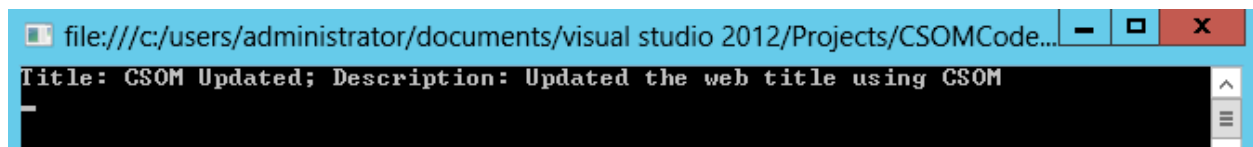


Figure 4.2.1: Update the properties of the website

4.3 How to get only specific properties of a website

In this example you will see how to get specific website properties using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Retrieve only the Title property
            clientContext.Load(web, w=>w.Title);

            // Execute the query to the server.
            clientContext.ExecuteQuery();

            // Web property - Display the Title
            Console.WriteLine("Title: " + web.Title);
            Console.ReadLine();
        }
    }
}

```

Output

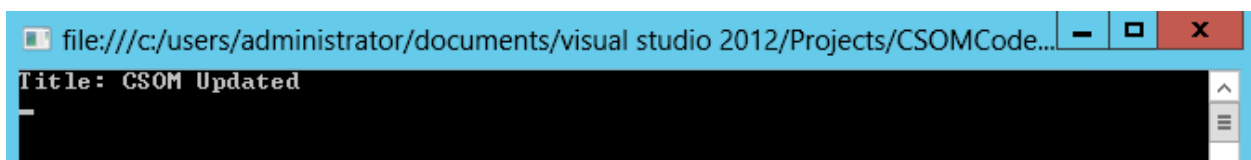


Figure 4.3.1: Get the specific property of the website

Note

If you try to retrieve any other property then you will get the following error:

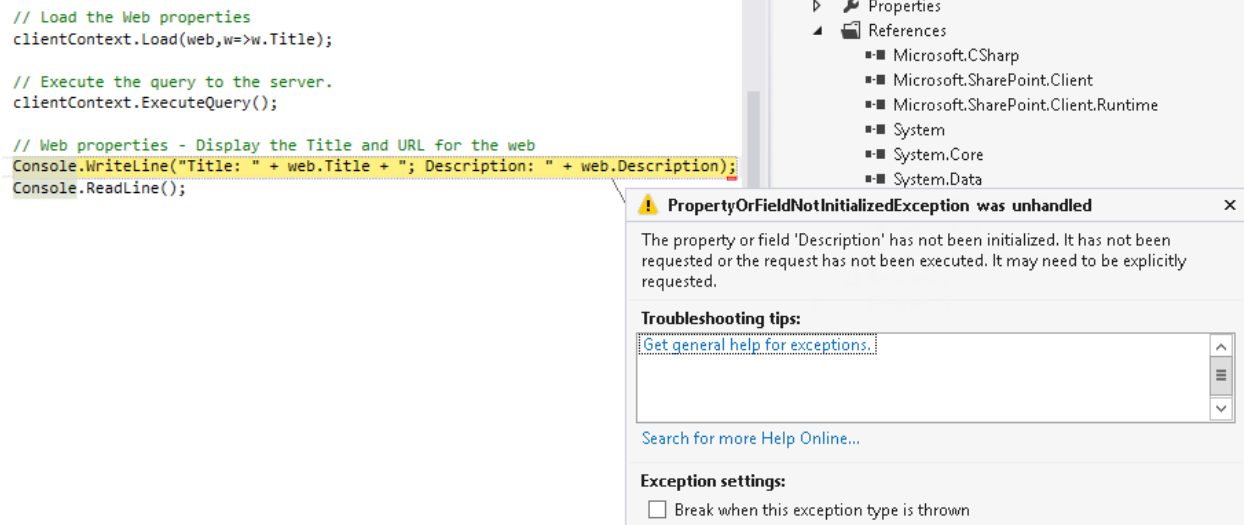


Figure 4.3.2: Error Message

4.4 How to delete a website

In this example you will see how to delete a website using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClieContext - Get the context for the SharePoint Site
        }
    }
}
```

```

        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/New Web/");

        // Delete the web object
        clientContext.Web.DeleteObject();

        // Execute the query to the server.
        clientContext.ExecuteQuery();
    }
}
}

```

Output

The Website is deleted successfully.

4.5 How to get all available web templates from website

In this example you will see how to get all the available web templates using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new

```

```

ClientContext("http://servername/sites/CSOM/");

    // Get the SharePoint web
    Web web = clientContext.Web;

    // Get all the site templates available for the site
    WebTemplateCollection templateColl = web.GetAvailableWebTemplates(1033,
false);

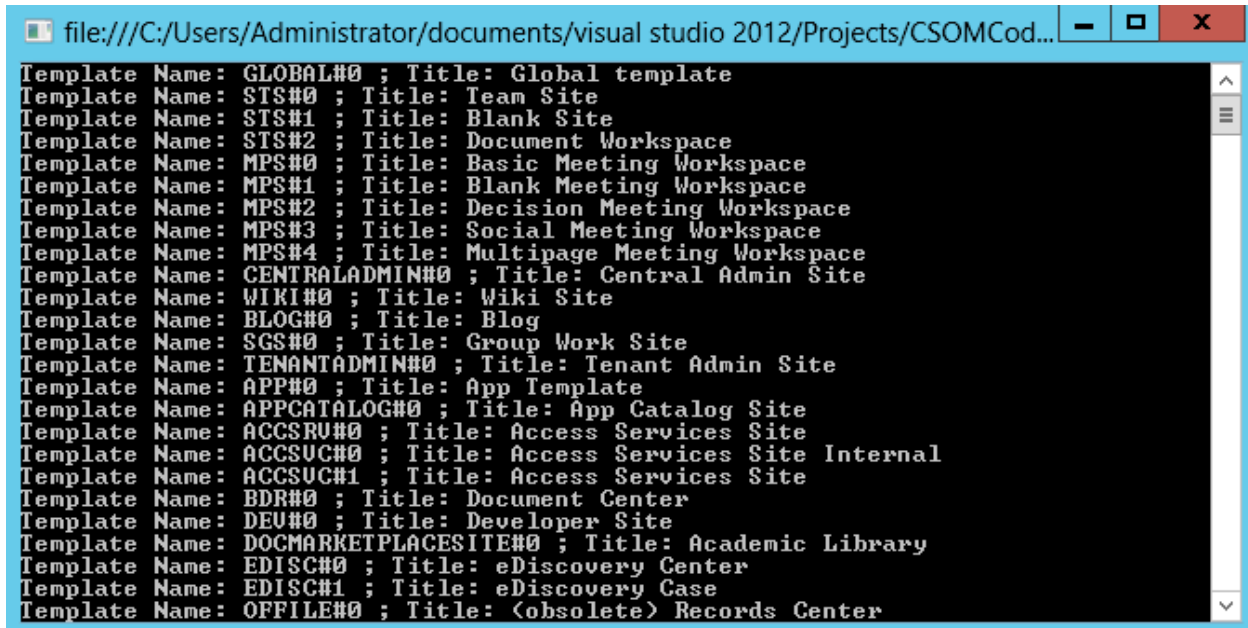
    clientContext.Load(templateColl);

    // Execute the query to the server
    clientContext.ExecuteQuery();

    // Loop through all the site templates available for the site
    foreach (WebTemplate template in templateColl)
    {
        // Display the site template Name and Title property
        Console.WriteLine("Template Name: " + template.Name + " ; Title: " +
template.Title);
    }
    Console.ReadLine();
}
}
}

```

Output



```

file:///C:/Users/Administrator/documents/visual studio 2012/Projects/CSOMCod...
Template Name: GLOBAL#0 ; Title: Global template
Template Name: STS#0 ; Title: Team Site
Template Name: STS#1 ; Title: Blank Site
Template Name: STS#2 ; Title: Document Workspace
Template Name: MPS#0 ; Title: Basic Meeting Workspace
Template Name: MPS#1 ; Title: Blank Meeting Workspace
Template Name: MPS#2 ; Title: Decision Meeting Workspace
Template Name: MPS#3 ; Title: Social Meeting Workspace
Template Name: MPS#4 ; Title: Multipage Meeting Workspace
Template Name: CENTRALADMIN#0 ; Title: Central Admin Site
Template Name: WIKI#0 ; Title: Wiki Site
Template Name: BLOG#0 ; Title: Blog
Template Name: SGS#0 ; Title: Group Work Site
Template Name: TENANTADMIN#0 ; Title: Tenant Admin Site
Template Name: APP#0 ; Title: App Template
Template Name: APPCATALOG#0 ; Title: App Catalog Site
Template Name: ACCSRU#0 ; Title: Access Services Site
Template Name: ACCSUC#0 ; Title: Access Services Site Internal
Template Name: ACCSUC#1 ; Title: Access Services Site
Template Name: BDR#0 ; Title: Document Center
Template Name: DEV#0 ; Title: Developer Site
Template Name: DOCMARKETPLACESITE#0 ; Title: Academic Library
Template Name: EDISC#0 ; Title: eDiscovery Center
Template Name: EDISC#1 ; Title: eDiscovery Case
Template Name: OFFILE#0 ; Title: <obsolete> Records Center

```

Figure 4.5.1: Get all available web templates

4.6 How to get all the active features from website

In this example you will see how to get all the active web features using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get all the active features from the site
            FeatureCollection featureColl = web.Features;
            clientContext.Load(featureColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

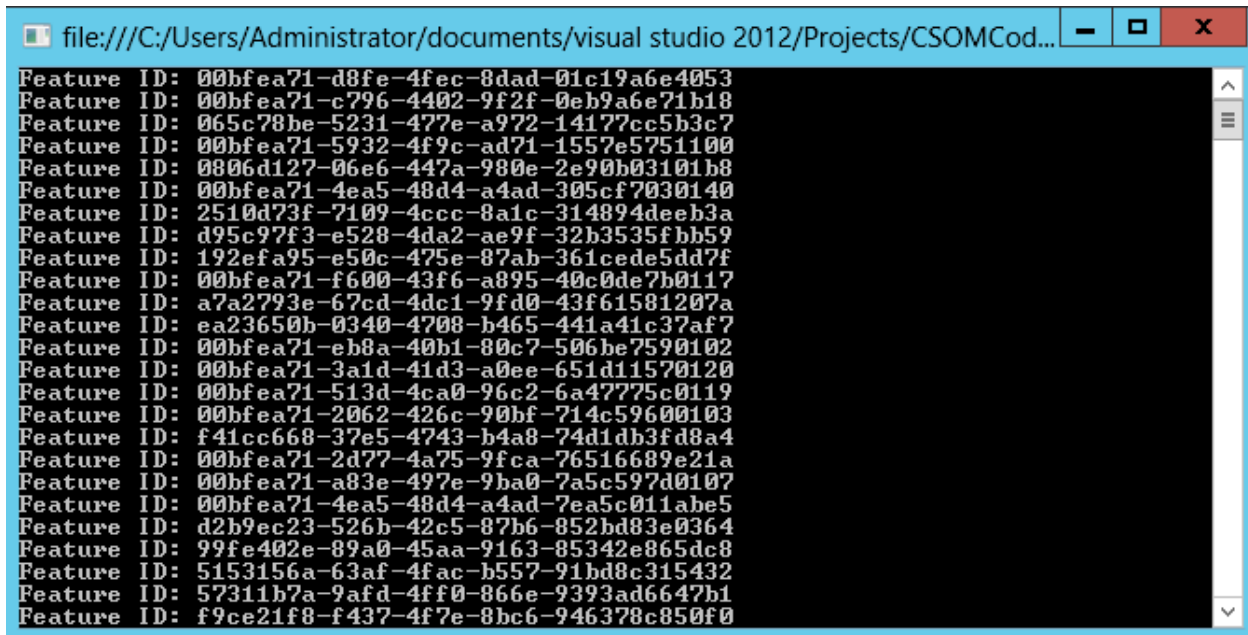
            // Loop through all the active features
            foreach (Feature feature in featureColl)
            {
                // Display the feature ID
                Console.WriteLine("Feature ID: " + feature.DefinitionId);
            }
        }
    }
}
```

```

    }
    Console.ReadLine();
}
}
}

```

Output



The screenshot shows a Windows command prompt window with the title bar "file:///C:/Users/Administrator/documents/visual studio 2012/Projects/CSOMCod...". The window contains a list of 20 Feature IDs, each on a new line, starting with "Feature ID:". The IDs are GUIDs in a standard format (8-4-4-4-12 hex digits). The list is as follows:

```

Feature ID: 00bfea71-d8fe-4fec-8dad-01c19a6e4053
Feature ID: 00bfea71-c796-4402-9f2f-0eb9a6e71b18
Feature ID: 065c78be-5231-477e-a972-14177cc5b3c7
Feature ID: 00bfea71-5932-4f9c-ad71-1557e5751100
Feature ID: 0806d127-06e6-447a-980e-2e90b03101b8
Feature ID: 00bfea71-4ea5-48d4-a4ad-305cf7030140
Feature ID: 2510d73f-7109-4ccc-8a1c-314894deeb3a
Feature ID: d95c97f3-e528-4da2-ae9f-32b3535fbb59
Feature ID: 192efa95-e50c-475e-87ab-361cede5dd7f
Feature ID: 00bfea71-f600-43f6-a895-40c0de7b0117
Feature ID: a7a2793e-67cd-4dc1-9fd0-43f61581207a
Feature ID: ea23650b-0340-4708-b465-441a41c37af7
Feature ID: 00bfea71-eb8a-40b1-80c7-506be7590102
Feature ID: 00bfea71-3a1d-41d3-a0ee-651d11570120
Feature ID: 00bfea71-513d-4ca0-96c2-6a47775c0119
Feature ID: 00bfea71-2062-426c-90bf-714c59600103
Feature ID: f41cc668-37e5-4743-b4a8-74d1db3fd8a4
Feature ID: 00bfea71-2d77-4a75-9fca-76516689e21a
Feature ID: 00bfea71-a83e-497e-9ba0-7a5c597d0107
Feature ID: 00bfea71-4ea5-48d4-a4ad-7ea5c011abe5
Feature ID: d2b9ec23-526b-42c5-87b6-852bd83e0364
Feature ID: 99fe402e-89a0-45aa-9163-85342e865dc8
Feature ID: 5153156a-63af-4fac-b557-91bd8c315432
Feature ID: 57311b7a-9afd-4ff0-866e-9393ad6647b1
Feature ID: f9ce21f8-f437-4f7e-8bc6-946378c850f0

```

Figure4.6.1: *Get all the active features*

5 Perform SharePoint list item tasks using CSOM

In this section you will see how to perform list item related tasks using the SharePoint 2013 .Net Client Side Object Model.

5.1 How to get all the items from the list

In this example you will see how to get all the items from the list using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // CamlQuery to retrieve the items from the custom list
            CamlQuery query = CamlQuery.CreateAllItemsQuery();
```

```
// Get all the items from the list
ListItemCollection itemColl = list.GetItems(query);
clientContext.Load(itemColl);

// Execute the query to the server
clientContext.ExecuteQuery();

// Loop through all the items
foreach (ListItem item in itemColl)
{
    // Display the item title field value
    Console.WriteLine(item["Title"].ToString());
}
Console.ReadLine();
}
}
```

Output

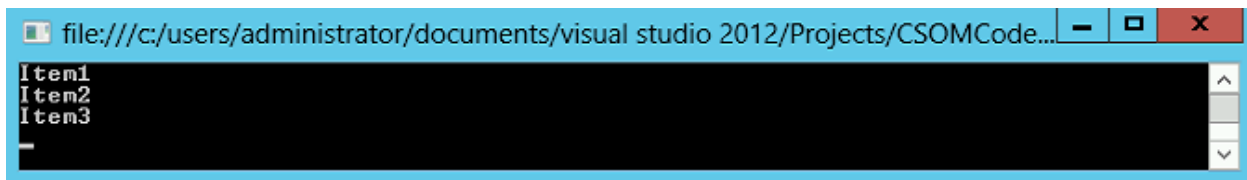


Figure 5.1.1: Get all the list items

5.2 How to create a new item in the list

In this example you will see how to create a new item in the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

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

```

using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Create a new item
            ListItemCreationInformation creationInfo = new ListItemCreationInformation();
            ListItem item = list.AddItem(creationInfo);

            // Set the title value for the new item
            item["Title"] = "New Item";

            // Update the item
            item.Update();
            clientContext.Load(item);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the new item title field value
            Console.WriteLine(item["Title"].ToString());
            Console.ReadLine();
        }
    }
}

```

Output

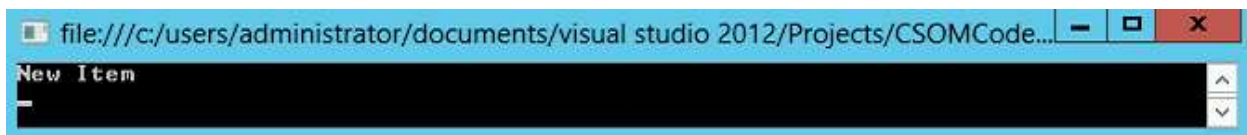


Figure 5.2.1: Create a new list item

5.3 How to update an item in the list

In this example you will see how to update an item in the list using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the list item by ID
            ListItem item = list.GetItemById(1);

            // Update the list item
            item["TextColumn"] = "Item Updated";
            item.Update();

            // Retrieve the item properties
            clientContext.Load(item);
        }
    }
}
```

```

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Display the item title field value
        Console.WriteLine(item["TextColumn"].ToString());
        Console.ReadLine();
    }
}
}

```

Output

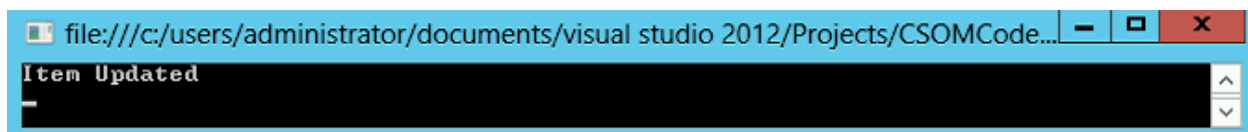


Figure 5.3.1: Update an item

Output

Here I am the item by ID 1 which exists in my list. Make sure you item ID exists in the list.

5.4 How to delete an item in the list

In this example you will see how to delete an item in the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{

```

```

class Program
{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Get the list by Title
        List list = web.Lists.GetByTitle("Custom List");

        // Get the list item by ID
        ListItem item = list.GetItemById(1);

        // Update the list item
        item.DeleteObject();

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

The list item is deleted successfully.

5.5 How to get the items from a list folder

In this example you will see how to get the items from the specified server relative URL of a list folder using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
```



```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Viewfields
            string[] viewFields = { "Title" };

            // CamlQuery to retrieve the items from the custom list
            CamlQuery query = new CamlQuery();

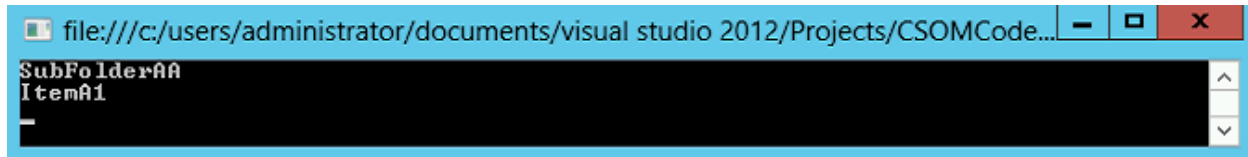
            // Specify the server relative URL of a list folder from which results will
            be returned.
            query.FolderServerRelativeUrl = "/sites/CSOM/Lists/Custom List/FolderA";

            // Get the items from the list
            ListItemCollection itemColl = list.GetItems(query);
            clientContext.Load(itemColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the items
            foreach (ListItem item in itemColl)
            {
                // Display the item title field value
                Console.WriteLine(item["Title"].ToString());
            }
            Console.ReadLine();
        }
    }
}

```

Output*Figure 5.5.1: Get items from the list folder***5.6 How to get the limited number of items from the list**

In this example you will see how to get the limited number of items from the list using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)
- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;
```

```
// Get the list by Title
List list = web.Lists.GetByTitle("Custom List");

// Viewfields
string[] viewFields = { "Title" };

// CamlQuery to retrieve the items from the custom list
// Use row limit and viewfields - To fetch only 2 items and Title field value
CamlQuery query = CamlQuery.CreateAllItemsQuery(2,viewFields);

// Get all the items from the list
ListItemCollection itemColl = list.GetItems(query);
clientContext.Load(itemColl);

// Execute the query to the server
clientContext.ExecuteQuery();

// Loop through all the items
foreach (ListItem item in itemColl)
{
    // Display the item title field value
    Console.WriteLine(item["Title"].ToString());
}
Console.ReadLine();
}
}
```

Output

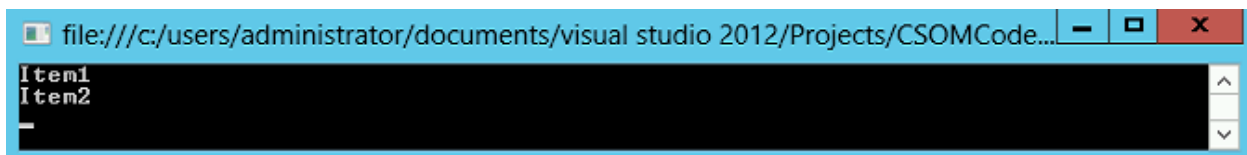


Figure 5.6.1: Get the limited number of items

5.7 How to get all the attachments for the list item

In this example you will see how to get all the attachments for the list item using the .Net Client Side Object Model.

Build the project using the following:

- a. [Create a console application.](#)

- b. Replace Program.cs with the below source code below.
- c. Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the list item by ID
            ListItem item = list.GetItemById(9);

            // Get all the attachments for the list item
            AttachmentCollection attachColl = item.AttachmentFiles;
            clientContext.Load(attachColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the attachments for an list item
            foreach (Attachment attachment in attachColl)
            {
                // Display the file name of the attachment
                Console.WriteLine(attachment.FileName);
            }
            Console.ReadLine();
        }
    }
}
```

```
}
```

Output

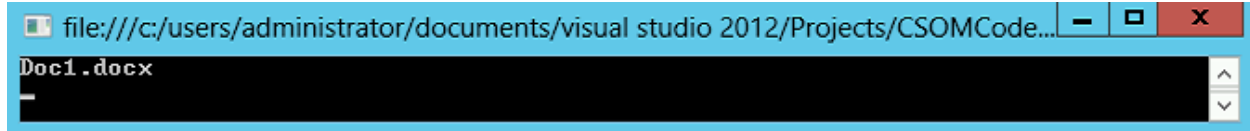


Figure 5.7.1: Get all the attachments

5.8 How to delete an attachment for the list item

In this example you will see how to get all the attachments for the list item using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;
```

```
// Get the list by Title
List list = web.Lists.GetByTitle("Custom List");

// Get the list item by ID
ListItem item = list.GetItemById(9);

// Get the attachment by file name
Attachment attach = item.AttachmentFiles.GetByFileName("Doc1.docx");

// Delete the attachment
attach.DeleteObject();

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}
```

Output

The attachment is deleted successfully for the list item.

6 Perform SharePoint content type tasks using CSOM

In this section you will see how to perform content type related tasks using the SharePoint 2013 .Net Client Side Object Model.

6.1 How to get all the content types from the website

In this example you will see how to get all the content types from the website using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get all the content types from the website
            ContentTypeCollection ctColl = web.ContentTypes;

            // Retrieve all content types from the website
```

```

        clientContext.Load(ctColl,
coll=>coll.Include(contentType=>contentType.Name));

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Loop through all the content types
        foreach (ContentType ct in ctColl)
        {
            // Display the content type name
            Console.WriteLine(ct.Name);
        }
        Console.ReadLine();
    }
}
}

```

Output

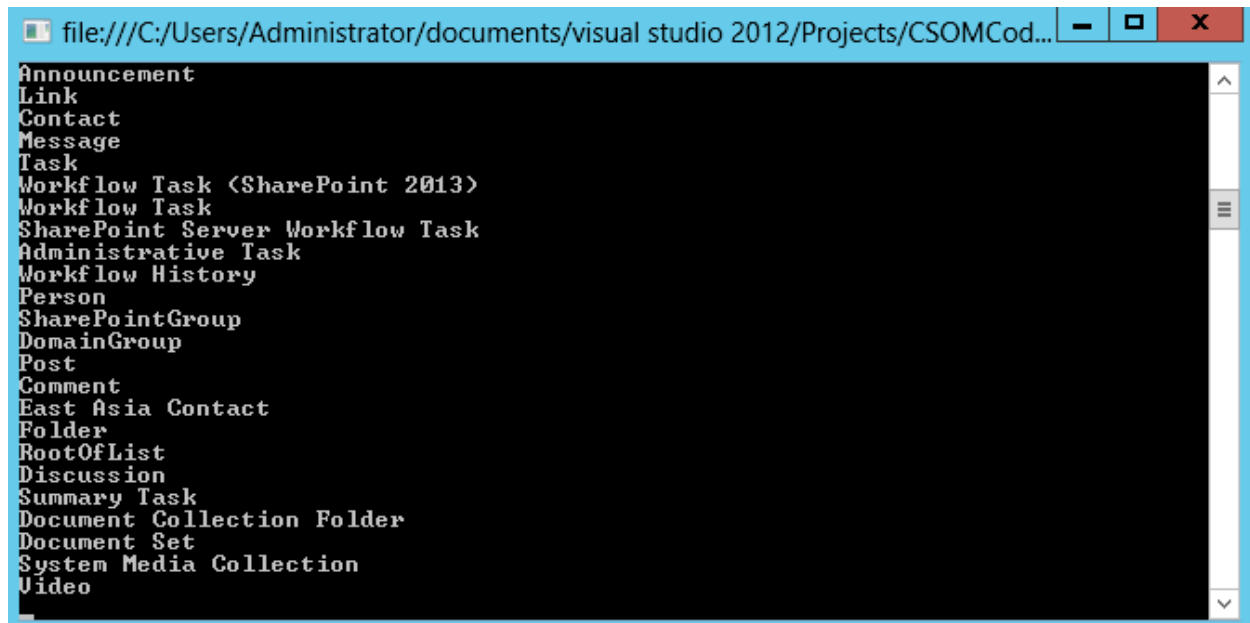


Figure6.1.1: Get all the content types from the website

6.2 How to create a site content type

In this example you will see how to create a new site content type using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get all the content types from the website
            ContentTypeCollection ctColl = web.ContentTypes;

            // Get the parent content type - Item (0x01)
            ContentType parentCT = web.ContentTypes.GetById("0x01");

            // Specifies properties that are used as parameters to initialize a new
content type
            ContentTypeCreationInformation ctCreationInfo = new
ContentTypeCreationInformation();
            ctCreationInfo.Name = "Vijai Content Type";
            ctCreationInfo.Description = "My custom content type created using CSOM";
            ctCreationInfo.Group = "Vijai Content Types";
            ctCreationInfo.ParentContentType = parentCT;

            // Add the new content type to the collection
            ContentType newCT = ctColl.Add(ctCreationInfo);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

```
}
}
```

Output

Navigate to the SharePoint site. Click on **Settings**. Click on **Site Settings**. Click on **Content Types** which is available under the **Galleries** section. You will be able to see a newly created content type under the **Vijai Content Types** group as shown in Figure 6.2.1.

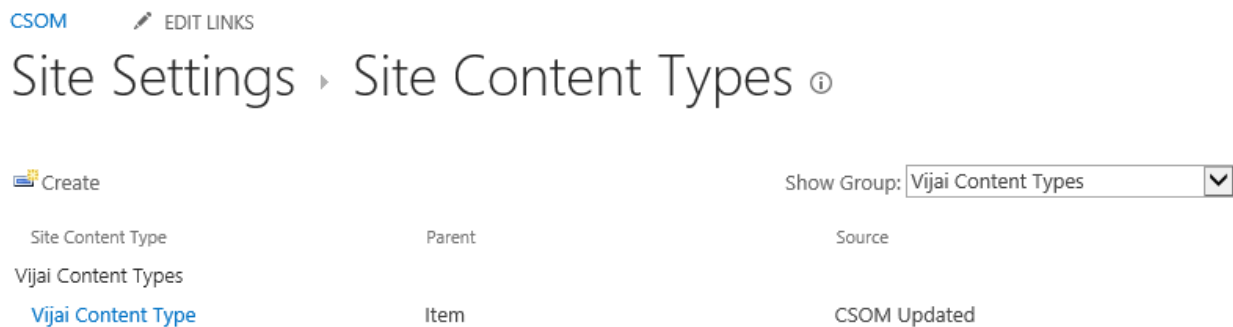


Figure 6.2.1: Newly created site content type

Click on the **Vijai Content Type**. You will be able to see the content type properties as shown in Figure 6.2.2.

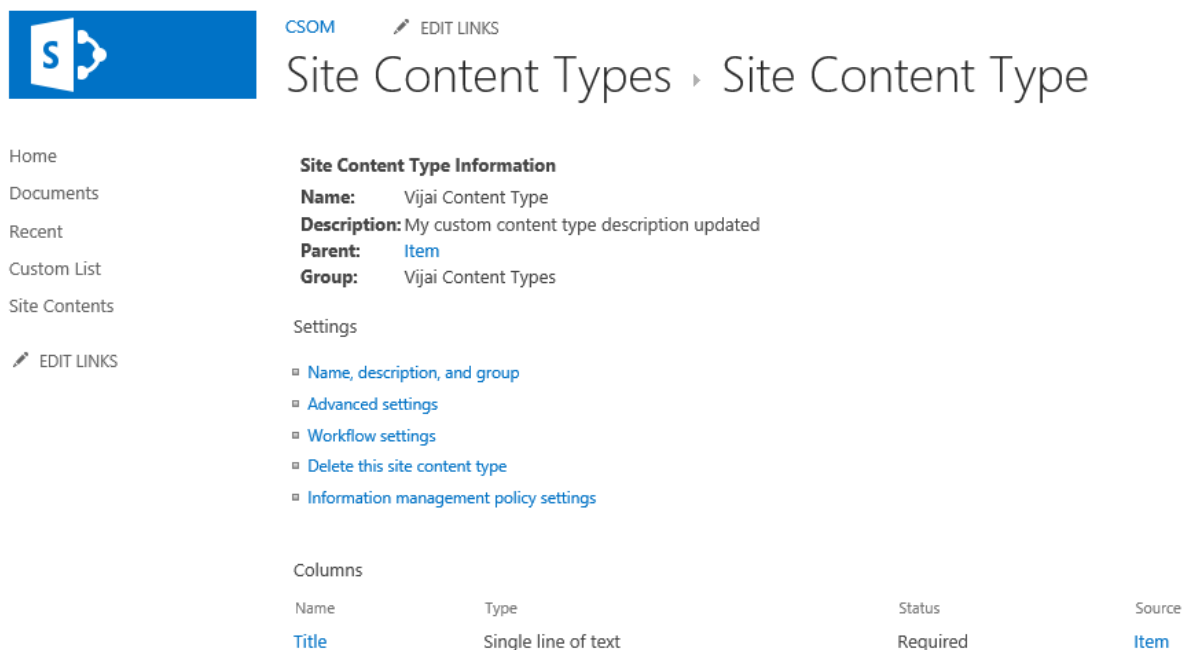


Figure 6.2.2: Newly created site content type properties

6.3 How to delete the site content type

In this example you will see how to delete the content type using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the content type by ID that has to be deleted
            ContentType ct =
web.ContentTypes.GetById("0x01009AFC059CB3CFA34E9F28C7CC3A0722EA");

            // Delete the content type
            ct.DeleteObject();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The site content type is deleted successfully.

6.4 How to set the site content type read only

In this example you will see how to set the site content type read only using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the content type by ID that has to be updated
            ContentType ct =
web.ContentTypes.GetById("0x01006354D981F2C0E04D8230E2E4AC2103D4");

            // Make the content type read only
            ct.ReadOnly = true;
            ct.Update(true);
        }
    }
}

```

```

    // Execute the query to the server
    clientContext.ExecuteQuery();
}
}
}

```

Output

Navigate to the SharePoint site. Click on **Settings**. Click on **Site Settings**. Click on **Content Types** which is available under the **Galleries** section. Click on the **Vijai Content Type** which is available under the **Vijai Content Types** group. You will be able to see that the content type is set as read only as shown in Figure 6.4.1.

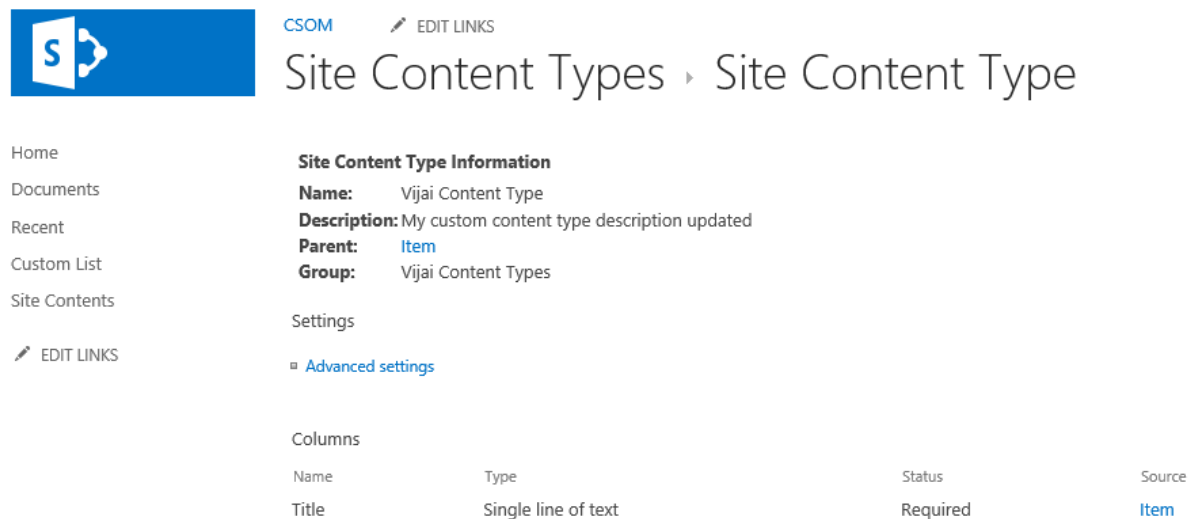


Figure 6.4.1: Read only content type

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.contenttype.readonly.aspx>

6.5 How to get all the content types from the list

In this example you will see how to get all the content types from the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the custom list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Content Type Collection
            ContentTypeCollection ctColl = list.ContentTypes;

            // Retrieve all content types from the list
            clientContext.Load(ctColl, coll => coll.Include(contentType =>
contentType.Name));

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the list content types
            foreach (ContentType ct in ctColl)
            {
                // Display the content type name
                Console.WriteLine(ct.Name);
            }
            Console.ReadLine();
        }
    }
}
```

```
}
}
```

Output

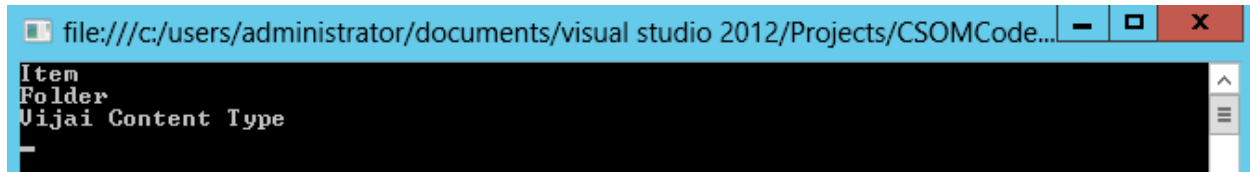


Figure 6.5.1: Get all the content types from the list

6.6 How to delete the content type from the list

In this example you will see how to delete the content type from the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
```

```

Web web = clientContext.Web;

// Get the custom list by title
List list = web.Lists.GetByTitle("Custom List");

// Content Type Collection
ContentTypeCollection ctColl = list.ContentTypes;

// Retrieve all content types from the list
clientContext.Load(ctColl, coll => coll.Include(contentType =>
contentType.Name));

// Execute the query to the server
clientContext.ExecuteQuery();

// Loop through all the list content types
foreach (ContentType ct in ctColl)
{
    // Delete the content type from the list
    if (ct.Name == "Vijai Content Type")
    {
        ct.DeleteObject();
        break;
    }
}

// Execute the query to the server
clientContext.ExecuteQuery();
}
}
}

```

Output

The Content type is deleted successfully from the list.

6.7 How to add existing content type to the list

In this example you will see how to add an existing content type to the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the custom list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the content type by ID
            ContentType ct =
web.ContentTypes.GetById("0x01006354D981F2C0E04D8230E2E4AC2103D4");

            // Add the existing content type to the list
            list.ContentTypes.AddExistingContentType(ct);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The Content type is added successfully to the list.

7 Perform SharePoint field tasks using CSOM

In this section you will see how to perform field related tasks using the SharePoint 2013 .Net Client Side Object Model.

7.1 How to get all the fields from the list

In this example you will see how to get all the fields from the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get all the fields from the list
            FieldCollection fieldColl = list.Fields;
            clientContext.Load(fieldColl);
        }
    }
}
```

```

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Loop through all the fields
        foreach (Field field in fieldColl)
        {
            // Display the field name and ID
            Console.WriteLine("Field Name: "+field.Title+"; ID: "+field.Id);
        }
        Console.ReadLine();
    }
}

```

Output

All the fields available for the list will be displayed.

7.2 How to update a specific field available in the list

In this example you will see how to update a specific field available in the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {

```

```

// ClientContext - Get the context for the SharePoint Site
// SharePoint site URL - http://servername/sites/CSOM
ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

// Get the SharePoint web
Web web = clientContext.Web;

// Get the list by Title
List list = web.Lists.GetByTitle("Custom List");

// Get a specific field by Title
Field field = list.Fields.GetByTitle("Column2");

// Update the description
field.Description = "Field updated using CSOM";
field.Update();
clientContext.Load(field);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display the field name and description
Console.WriteLine(field.Title + " description updated to : " +
field.Description);
Console.ReadLine();
    }
}
}

```

Output

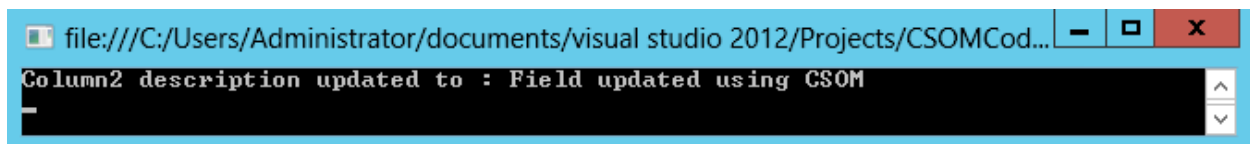


Figure 7.2.1: Update a specific field

7.3 How to add a field in the list

In this example you will see how to add a field to the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)

- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // string variable to store the field schema xml
            string schemaXML = "<Field DisplayName='CustomField' Type='Text' />";

            // Add a field to the list
            list.Fields.AddFieldAsXml(schemaXML, true, AddFieldOptions.DefaultValue);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

Navigate to the SharePoint site. Click on the **Custom List** link available in the quick launch bar. Click on the **List Settings** in the ribbon interface. You will be able to see a new field added to the list which is available under the **Columns** section.

7.4 How to add an existing field to the list

In this example you will see how to add an existing field to the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get a specific field by Title from site columns
            Field field = web.Fields.GetByTitle("MultiChoiceColumn");

            // Add the existing field to the list
            list.Fields.Add(field);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

Navigate to the SharePoint site. Click on the **Custom List** link available in the quick launch bar. Click on the **List Settings** in the ribbon interface. You will be able to see that a new field was added to the list which is available under the **Columns** section.

7.5 How to delete a field from the list

In this example you will see how to delete a field from the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get a specific field by Title
            Field field = list.Fields.GetByTitle("CustomField");
```

```

        // Delete a field from the list
        field.DeleteObject();

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Custom List** link available in the quick launch bar. Click on the **List Settings** in the ribbon interface. You will be able to see that a field is deleted from the list which is available under the **Columns** section.

7.6 How to set the default value for the list field

In this example you will see how to set the default value for the list field using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new

```



```

ClientContext("http://servername/sites/CSOM/");

    // Get the SharePoint web
    Web web = clientContext.Web;

    // Get the list by Title
    List list = web.Lists.GetByTitle("Custom List");

    // Get a specific field by Title
    Field field = list.Fields.GetByTitle("TextColumn");

    // Set the default value for the field
    field.DefaultValue = "SampleText";

    // Update the field
    field.Update();

    // Execute the query to the server
    clientContext.ExecuteQuery();
}
}
}

```

Output

Navigate to the SharePoint site. Click on the **Custom List** link available in the quick launch bar. Click on the **List Settings** in the ribbon interface. Click on the the **TextColumn** which is available under the **Columns** section. You will be able see that a default value was set for the field.

7.7 How to get the calculated field formula

In this example you will see how to get the calculated field formula using the .Net Client Side Object Model.

Build the project using the following:

- [Create a console application.](#)
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the calculated field by title
            Field field = list.Fields.GetByTitle("Calculated");

            // Cast the field
            FieldCalculated calculatedField =
clientContext.CastTo<FieldCalculated>(field);
            clientContext.Load(calculatedField);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Get the calculated field formula
            Console.WriteLine(calculatedField.Formula);
            Console.ReadLine();
        }
    }
}

```

Output

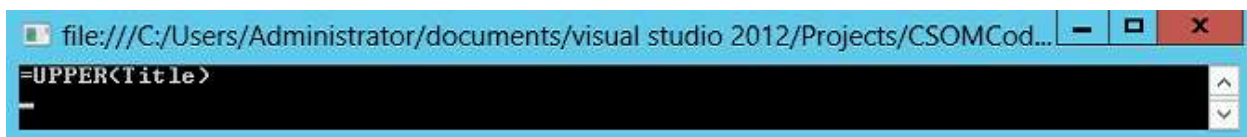


Figure 7.7.1: Read only content type

7.8 How to set the formula for the calculated field

In this example you will see how to set the formula for the calculated field using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the calculated field by title
            Field field = list.Fields.GetByTitle("Calculated");

            // Cast the field
            FieldCalculated calculatedField =
clientContext.CastTo<FieldCalculated>(field);

            // Set the formula for the calculated field
            calculatedField.Formula = "=UPPER([Title])";
```

```

        // Update the field
        calculatedField.Update();

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

The screenshot displays the 'Custom List' interface in SharePoint 2013. The left sidebar shows the navigation pane with 'Custom List' selected. The main area shows a table of items. The 'Calculated' column values are highlighted with a red box.

Item	Title	ID	TextColumn	Calculated
FolderA	FolderA	4		FOLDERA
FolderB	FolderB	7		FOLDERB
Item2	Item2	2		ITEM2
Item3	Item3	3		ITEM3
New Item	New Item	9		NEW ITEM

Figure 7.8.1: Values are updated based on the formula

8 Perform SharePoint list view tasks using CSOM

In this section you will see how to perform list view related tasks using the SharePoint 2013 .Net Client Side Object Model.

8.1 How to get all the views for the list

In this example you will see how to get all the views for the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get all the view for the custom list
            ViewCollection viewColl = list.Views;
            clientContext.Load(viewColl);
        }
    }
}
```

```

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Loop through all the views
        foreach (View view in viewColl)
        {
            // Display the view name
            Console.WriteLine(view.Title);
        }
        Console.ReadLine();
    }
}
}

```

Output

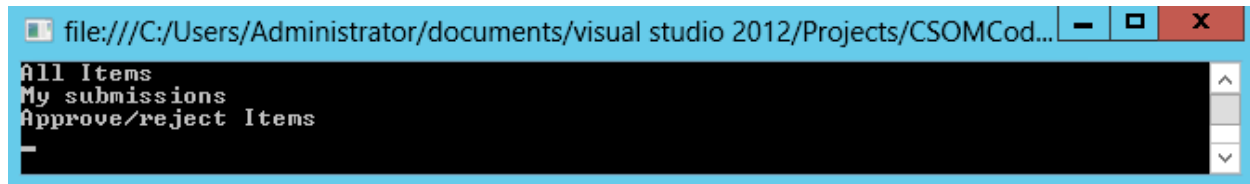


Figure 8.1.1: Get all the list views

8.2 How to create a new list view

In this example you will see how to create a new list view using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

```

```

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get all the view for the custom list
            ViewCollection viewColl = list.Views;

            // Specify the columns that should be displayed
            string[] viewFields = { "Column1" };

            // Specifies the properties used to create a new list view
            ViewCreationInformation creationInfo = new ViewCreationInformation();
            creationInfo.Title = "Vijai View";
            creationInfo.RowLimit = 50;
            creationInfo.ViewFields = viewFields;
            creationInfo.ViewTypeKind = ViewType.None;
            creationInfo.SetAsDefaultView = true;
            viewColl.Add(creationInfo);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

A new list view is created successfully.

8.3 How to get all the fields available in the list view

In this example you will see how to get all the fields available in the list view using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the specific view by Title
            View view = list.Views.GetByTitle("Vijai View");

            // Get all the fields available in the list view
            ViewFieldCollection viewFieldColl = view.ViewFields;
            clientContext.Load(viewFieldColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the view fields
            foreach (string viewField in viewFieldColl)
            {
                //Display the view field available in the list view
                Console.WriteLine(viewField);
            }
        }
    }
}
```



```

        Console.ReadLine();
    }
}

```

Output

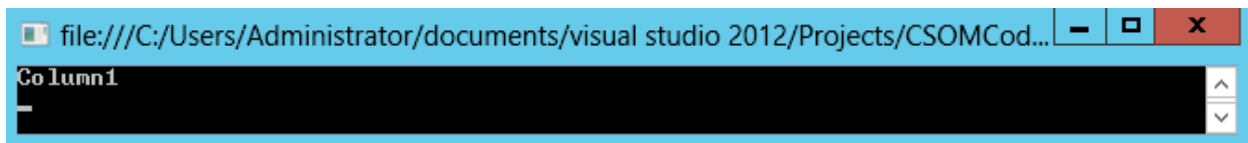


Figure 8.3.1: Get all the fields

8.4 How to set the default view in the list

In this example you will see how to set the default value in the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web

```

```

Web web = clientContext.Web;

// Get the list by Title
List list = web.Lists.GetByTitle("Custom List");

// Get the specific view by Title
View view = list.Views.GetByTitle("All Items");

// Set the view as default view
view.DefaultView = true;

// Update the list view
view.Update();

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}

```

Output

The All Items view is set as the default view for the custom list.

8.5 How to add a field to the list view

In this example you will see how to add a field to the list view using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{

```

```

class Program
{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Get the list by Title
        List list = web.Lists.GetByTitle("Custom List");

        // Get the specific view by Title
        View view = list.Views.GetByTitle("Vijai View");

        // Add the field to the list view
        view.ViewFields.Add("Column2");

        // Update the list view
        view.Update();

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

The **Column2** field is added successfully to the “**Vijai View**” list view.

8.6 How to delete a field from the list view

In this example you will see how to delete a field from the list view using the Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the specific view by Title
            View view = list.Views.GetByTitle("Vijai View");

            // Remove a field from the list view
            view.ViewFields.Remove("Column2");

            // Update the list view
            view.Update();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

The **Column2** field is removed successfully from the “**Vijai View**” list view.

8.7 How to delete a list view

In this example you will see how to delete a list view using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Custom List");

            // Get the specific view by Title
            View view = list.Views.GetByTitle("Vijai View");

            // Delete the list view
            view.DeleteObject();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The **“Vijai View”** list view is deleted successfully from the custom list.

9 Perform SharePoint folder tasks using CSOM

In this section you will see how to perform folder related tasks using the SharePoint 2013 .Net Client Side Object Model.

9.1 How to get all the top level folders from the website

In this example you will see how to get all the top level folders from the website using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get all the top level folder collection from the website
            FolderCollection folderColl = web.Folders;
            clientContext.Load(folderColl);

            // Execute the query to the server
```

```

        clientContext.ExecuteQuery();

        // Loop through all the folders in the website
        foreach (Folder folder in folderColl)
        {
            // Display the folder name
            Console.WriteLine(folder.Name);
        }
        Console.ReadLine();
    }
}
}

```

Output

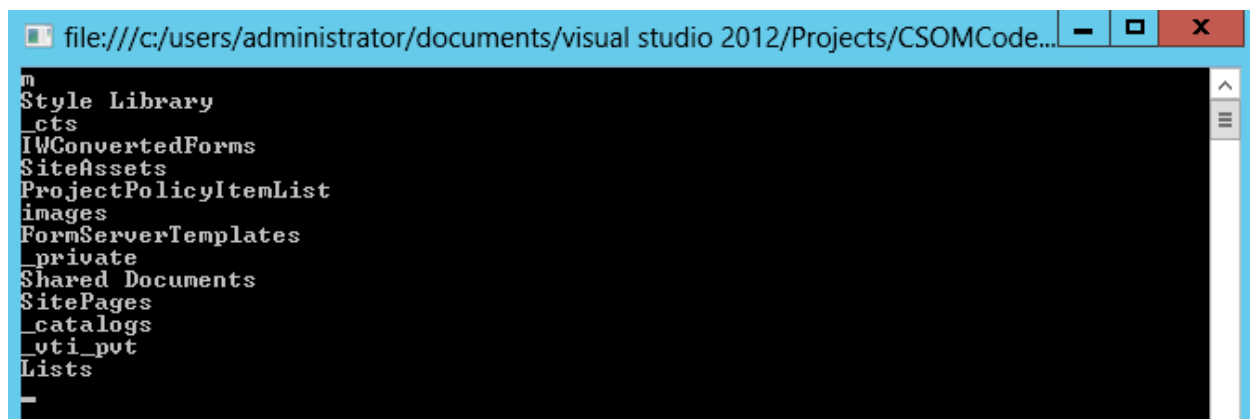


Figure 9.1.1: Get all the top level folders

9.2 How to get all the top level folders from the list

In this example you will see how to get all the top level folders from the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;

```



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by title
            List list = web.Lists.GetByTitle("Documents");

            // Get all the top level folder collection from the list
            FolderCollection folderColl = list.RootFolder.Folders;
            clientContext.Load(folderColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the folders in the list
            foreach (Folder folder in folderColl)
            {
                // Display the folder name
                Console.WriteLine(folder.Name);
            }
            Console.ReadLine();
        }
    }
}

```

Output

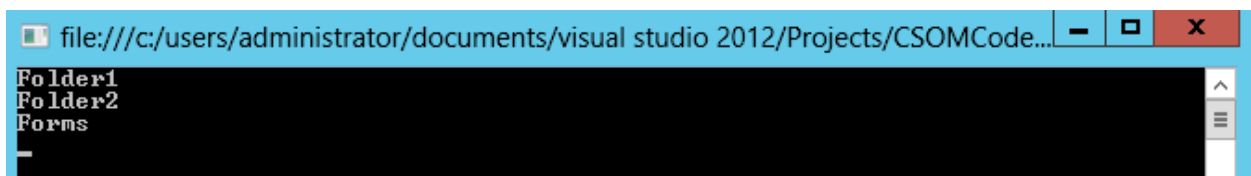


Figure9.2.1: *Get all the top level folders*

9.3 How to get the subfolders from the list

In this example you will see how to get the subfolders from the list using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Returns the folder object located at the specified server-relative URL
            FolderCollection folderColl = web.GetFolderByServerRelativeUrl("Shared
Documents/Folder1").Folders;
            clientContext.Load(folderColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the folders
            foreach (Folder folder in folderColl)
            {
                // Display the folder name
            }
        }
    }
}
```

```

        Console.WriteLine(folder.Name);
    }
    Console.ReadLine();
}
}
}

```

Output

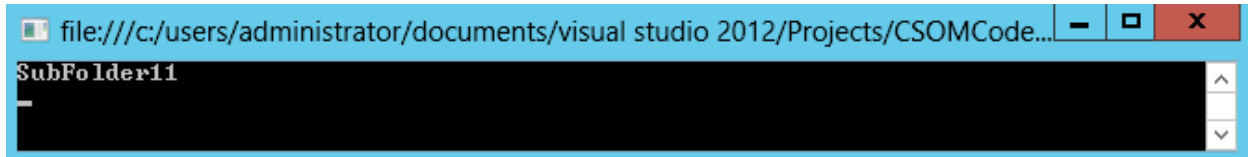


Figure 9.3.1: Get all the subfolders

9.4 How to delete a folder from the list

In this example you will see how to delete a folder from the list using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new

```

```

ClientContext("http://servername/sites/CSOM/");

    // Get the SharePoint web
    Web web = clientContext.Web;

    // Returns the folder object located at the specified server-relative URL
    Folder folder = web.GetFolderByServerRelativeUrl("Shared
Documents/Folder1/SubFolder11");

    // Delete the folder
    folder.DeleteObject();

    // Execute the query to the server
    clientContext.ExecuteQuery();
}
}
}

```

Output

The Folder at the specified server relative URL is deleted successfully.

9.5 How to create a new folder in the document library

In this example you will see how to create a new folder in the document library using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Adds the folder that is located at the specified URL to the collection
        // Create a new folder in Shared Documents
        Folder folder = web.Folders.Add("Shared Documents/Folder3");

        // Retrieves the newly created folder properties
        clientContext.Load(folder);

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Display the folder name and url
        Console.WriteLine("Folder Name: " + folder.Name + " ; URL: " +
folder.ServerRelativeUrl);
        Console.ReadLine();
    }
}

```

Output

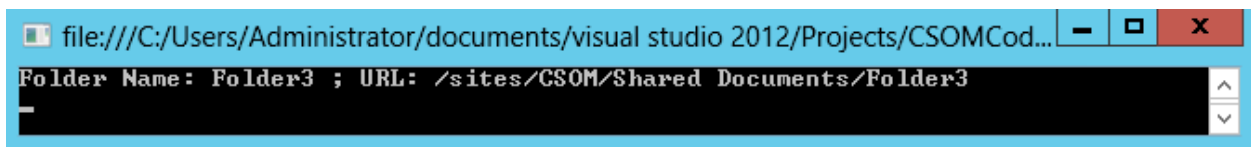


Figure 9.5.1: Create a new folder

9.6 How to get the number of items inside the folder

In this example you will see how to get the number of items inside the folder using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)

- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Returns the folder object located at the specified server-relative URL
            Folder folder = web.GetFolderByServerRelativeUrl("Shared Documents/Folder3");

            // Retrieve the folder properties
            clientContext.Load(folder);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the number of items inside the list folder
            Console.WriteLine("Number of items inside the folder: " + folder.ItemCount);
            Console.ReadLine();
        }
    }
}
```

Output

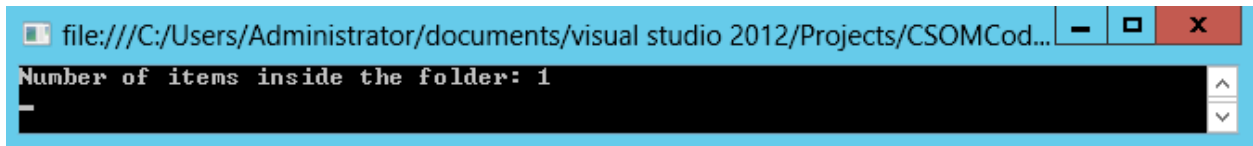


Figure9.6.1: *Get the count of items*

10 Perform SharePoint file tasks using CSOM

In this section you will see how to perform file related tasks using the SharePoint 2013 .Net Client Side Object Model.

10.1 How to get the major version of the file

In this example you will see how to get the major version of the file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);
        }
    }
}
```



```

        // Gets the file that is represented by the item from a document library
        File file = item.File;
        clientContext.Load(file);

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Display the major version of the file
        Console.WriteLine("Major version of the file: "+file.MajorVersion);
        Console.ReadLine();
    }
}
}

```

Output

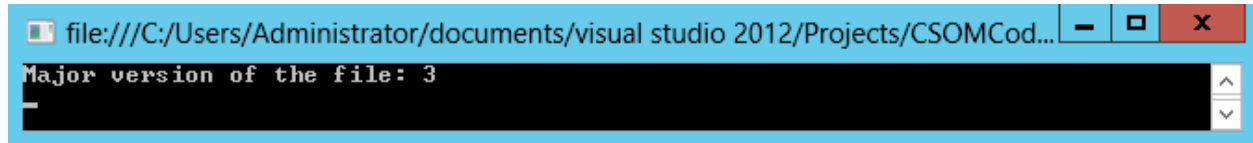


Figure10.1.1: Get the major version of the file

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.majorversion.aspx>

10.2 How to get the minor version of the file

In this example you will see how to get the minor version of the file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;
            clientContext.Load(file);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the minor version of the file
            Console.WriteLine("Minor version of the file: "+file.MinorVersion);
            Console.ReadLine();
        }
    }
}

```

Output

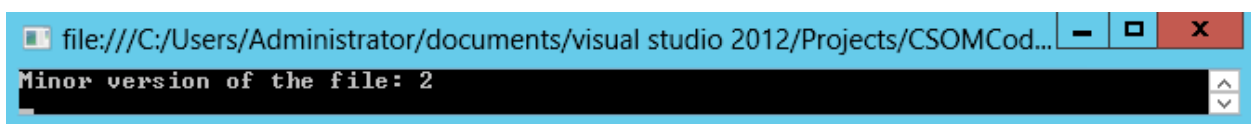


Figure10.2.1: Get the minor version of the file

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.minorversion.aspx>

10.3 How to check out the file in the document library

In this example you will see how to check out the file in the document library using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;

            // Check out the file
            file.CheckOut();
        }
    }
}
```

```

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

The specified file is checked out successfully.

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.checkout.aspx>

10.4 How to get the user login name who has checked out the file

In this example you will see how to get the user login name that has checked out the file using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

```

```

// Get the SharePoint web
Web web = clientContext.Web;

// Get the list by Title
List list = web.Lists.GetByTitle("Documents");

// Get an item by ID
ListItem item = list.GetItemById(6);

// Gets the file that is represented by the item from a document library
File file = item.File;

// Get the checked out by user object
User user = file.CheckedOutByUser;
clientContext.Load(user);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display the user login name who has checked out the file
Console.WriteLine("Checked out by: " + user.LoginName);
Console.ReadLine();
    }
}
}

```

Output

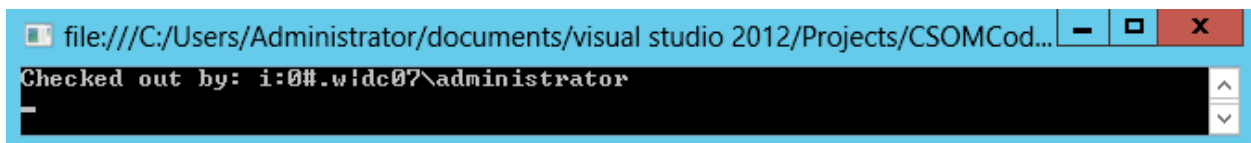


Figure10.4.1: Get the login name of the user

Reference

<http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.file.checkedoutbyuser.aspx>

10.5 How to get the user login name who added the file

In this example you will see how to get the user login name that added the file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;

            // Get the user who added the file
            User user = file.Author;
            clientContext.Load(user);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the user login name who added the file
            Console.WriteLine("Author: " + user.LoginName);
            Console.ReadLine();
        }
    }
}
```

```
}
```

Output

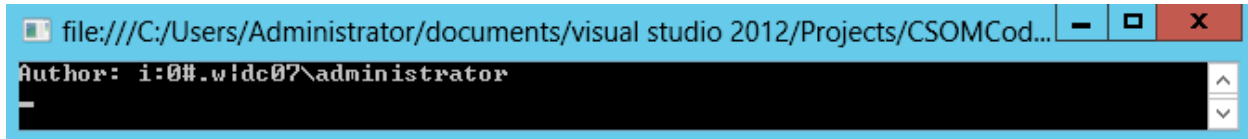


Figure 10.5.1: Get the login name of the user

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.author.aspx>

10.6 How to get the check out type associated with the file

In this example you will see how to get the check out type associated with the file using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
```

```

ClientContext("http://servername/sites/CSOM/");

// Get the SharePoint web
Web web = clientContext.Web;

// Get the list by Title
List list = web.Lists.GetByTitle("Documents");

// Get an item by ID
ListItem item = list.GetItemById(6);

// Gets the file that is represented by the item from a document library
File file = item.File;
clientContext.Load(file);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display the check out type associated with the file
Console.WriteLine("CheckOutType: " + file.CheckOutType.ToString());
Console.ReadLine();
    }
}
}

```

Output

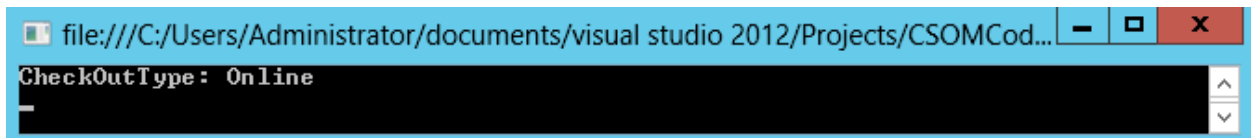


Figure10.6.1: Get the check out type

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.checkouttype.aspx>

10.7 How to check in the file

In this example you will see how to check in the file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.

c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;

            // Check in the file
            file.CheckIn("Checked in using CSOM", CheckinType.MajorCheckIn);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The file is checked in as major version successfully.

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.checkin.aspx>

10.8 How to get the check-in comment of the file

In this example you will see how to get the latest check-in comment of the file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;
            clientContext.Load(file);
        }
    }
}
```

```

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Display the check in comment
        Console.WriteLine("File check in comment: " + file.CheckInComment);
        Console.ReadLine();
    }
}
}

```

Output

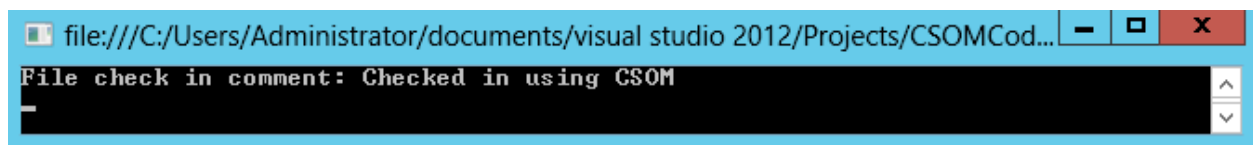


Figure10.8.1: Check in comment

Reference

msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.checkincomment.aspx

10.9 How to unpublish the major version of the file

In this example you will see how to unpublish the major version of the file using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

```

```

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;

            // Unpublish the major version
            file.UnPublish("Unpublish the major version using CSOM");

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

The major version of the file is unpublished successfully.

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.unpublish.aspx>

10.10 How to discard checked-out of the file

In this example you will see how to discard the checked-out file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)

- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;

            // Undo check out
            file.UndoCheckOut();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.undocheckout.aspx>

10.11 How to delete the file from the document library

In this example you will see how to delete the file from the document library using the Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Gets the file that is represented by the item from a document library
            File file = item.File;

            // Delete the file object
            file.DeleteObject();

            // Execute the query to the server
```

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

Reference

The file is deleted successfully from the document library.

Reference

<http://msdn.microsoft.com/en-IN/library/microsoft.sharepoint.client.file.deleteobject.aspx>

11 Perform SharePoint file version tasks using CSOM

In this section you will see how to perform file version related tasks using the SharePoint 2013 .Net Client Side Object Model.

11.1 How to get all the versions for the file

In this example you will see how to get all the versions for the file using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list=web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item =list.GetItemById(6);
        }
    }
}
```



```

// Get all the file versions for an item
FileVersionCollection versionColl=item.File.Versions;
clientContext.Load(versionColl);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display the version details
Console.WriteLine("Version Details");
Console.WriteLine("-----");
// Loop through all the versions
foreach (FileVersion version in versionColl)
{
    // Display the versionlabel and check in comment
    Console.WriteLine("VersionLabel   : "+version.VersionLabel);
    Console.WriteLine("CheckInComment : " + version.CheckInComment);
    Console.WriteLine();
}
Console.ReadLine();
}
}
}

```

Output

```

file:///C:/Users/Administrator/documents/visual studio 2012/Projects/CSOMCod...
Version Details
-----
VersionLabel   : 1.0
CheckInComment : 
VersionLabel   : 1.1
CheckInComment : 
VersionLabel   : 1.2
CheckInComment : 
VersionLabel   : 1.3
CheckInComment : 
VersionLabel   : 1.4
CheckInComment : 
VersionLabel   : 1.5
CheckInComment : 
VersionLabel   : 2.0
CheckInComment : Published
VersionLabel   : 2.1
CheckInComment : 

```

Figure11.1.1: Get all the versions for the file

11.2 How to get the file version for the document by version id

In this example you will see how to get the file version for the document by version id using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Get a file version for the document using version ID
            FileVersion version = item.File.Versions.GetById(512);
            clientContext.Load(version);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

```

        // Display the version details
        Console.WriteLine("Version Details");
        Console.WriteLine("-----");

        // Display the versionlabel and check in comment
        Console.WriteLine("VersionLabel   : " + version.VersionLabel);
        Console.WriteLine("CheckInComments: " + version.CheckInComment);
        Console.ReadLine();
    }
}
}

```

Output

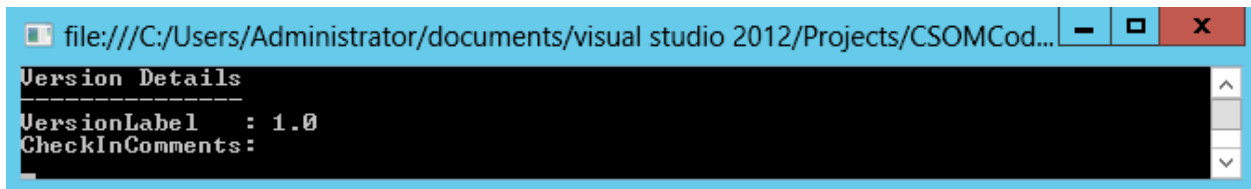


Figure 11.2.1: Get the file version by version ID

11.3 How to delete a file version by version ID for the document

In this example you will see how to delete a file version for the document using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{

```

```
class Program
{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Get the list by Title
        List list = web.Lists.GetByTitle("Documents");

        // Get an item by ID
        ListItem item = list.GetItemById(6);

        // Delete the specified version of the file
        item.File.Versions.DeleteById(513);

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}
```

Output

The specified version of the file is deleted successfully.

Version History

[Delete All Versions](#) | [Delete Minor Versions](#)

No. ↓	Modified	Modified By	Size	Comments
This is the current published major version				
3.0	7/30/2013 5:30 AM	<input type="checkbox"/> Administrator	2.1 KB	Updated the Title
	Title Program5			
2.1	7/30/2013 5:30 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program			
2.0	7/30/2013 5:29 AM	<input type="checkbox"/> Administrator	2.1 KB	Published
	Title Program4			
1.5	7/30/2013 5:29 AM	<input type="checkbox"/> Administrator	2.1 KB	
1.4	7/30/2013 5:29 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program			
1.3	7/30/2013 5:28 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program2			
1.2	7/30/2013 5:28 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program1			

Figure11.3.1: Delete the specific version

11.4 How to delete a file version by version label for the document

In this example you will see how to delete a file version by version label for the document using the .Net Client Side Object Model.

Build the project using the following:

- [Create a console application.](#)
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Deletes the file version object with the specified version label
            item.File.Versions.DeleteByLabel("1.2");

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The specified version of the file is deleted successfully.

Version History

[Delete All Versions](#) | [Delete Minor Versions](#)

No. ↓	Modified	Modified By	Size	Comments
This is the current published major version				
3.0	7/30/2013 5:30 AM	<input type="checkbox"/> Administrator	2.1 KB	Updated the Title
	Title Program5			
2.1	7/30/2013 5:30 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program			
2.0	7/30/2013 5:29 AM	<input type="checkbox"/> Administrator	2.1 KB	Published
	Title Program4			
1.5	7/30/2013 5:29 AM	<input type="checkbox"/> Administrator	2.1 KB	
1.4	7/30/2013 5:29 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program			
1.3	7/30/2013 5:28 AM	<input type="checkbox"/> Administrator	2.1 KB	
	Title Program2			

Figure 11.4.1: Delete the specific version

11.5 How to restore a specific file version for the document

In this example you will see how to restore a specific file version for the document using the .Net Client Side Object Model.

Build the project using the following:

- [Create a console application.](#)
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

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

```

using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Creates a new file version from the file specified by the version label
            item.File.Versions.RestoreByLabel("2.0");

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

The specified version of the file is restored successfully.

11.6 How to check if the file version is a current version for the document

In this example you will see how to check if the file version is a current version for the document using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.

c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the list by Title
            List list = web.Lists.GetByTitle("Documents");

            // Get an item by ID
            ListItem item = list.GetItemById(6);

            // Get all the versions of the file
            FileVersionCollection versionColl = item.File.Versions;
            clientContext.Load(versionColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the versions
            foreach (FileVersion version in versionColl)
            {
                // Check if the version is the current version
                if (version.IsCurrentVersion)
                {
                    // Display the current version label
                    Console.WriteLine("Current version of the file: " +
version.VersionLabel);
                    break;
                }
            }
        }
    }
}
```

```

    }
    Console.ReadLine();
}
}
}

```

Output

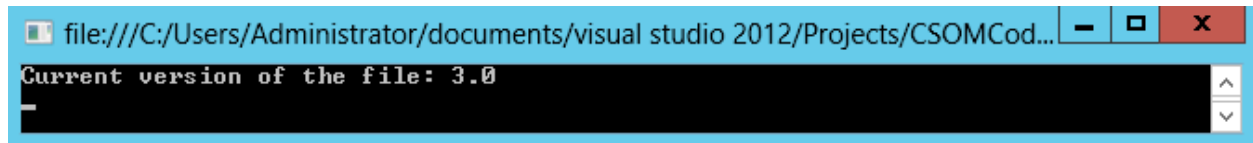


Figure 11.6.1: Check if the version is current version

11.7 How to delete all the file versions for the document

In this example you will see how to delete all the file versions for the document using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

```

```
// Get the SharePoint web
Web web = clientContext.Web;

// Get the list by Title
List list = web.Lists.GetByTitle("Documents");

// Get an item by ID
ListItem item = list.GetItemById(6);

// Gets the file that is represented by the item from a document library
File file = item.File;

// Delete all the versions for the file
file.Versions.DeleteAll();

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
```

Output

All the file versions for the document are deleted successfully.

12 Perform SharePoint group tasks using CSOM

In this section you will see how to perform group related tasks using the SharePoint 2013 .Net Client Side Object Model.

12.1 How to get all the site groups

In this example you will see how to get all the site groups using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get all the site groups
            GroupCollection groupColl = web.SiteGroups;
            clientContext.Load(groupColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

```

        // Loop through all the groups
        foreach (Group group in groupColl)
        {
            // Display the group name
            Console.WriteLine(group.Title);
        }
        Console.ReadLine();
    }
}

```

Output

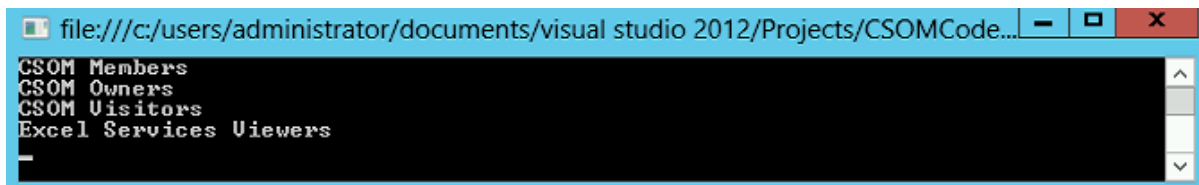


Figure 12.1.1: Get all the site groups

12.2 How to create a new site group

In this example you will see how to create a new site group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Get all the site groups
        GroupCollection groupColl = web.SiteGroups;

        // Create a new site group
        GroupCreationInformation creationInfo = new GroupCreationInformation();
        creationInfo.Title = "Vijai Custom Group";
        creationInfo.Description = "Custom group created using CSOM";
        Group newGroup = groupColl.Add(creationInfo);
        clientContext.Load(newGroup);

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Display the newly created group name
        Console.WriteLine(newGroup.Title + " is created successfully");
        Console.ReadLine();
    }
}

```

Output

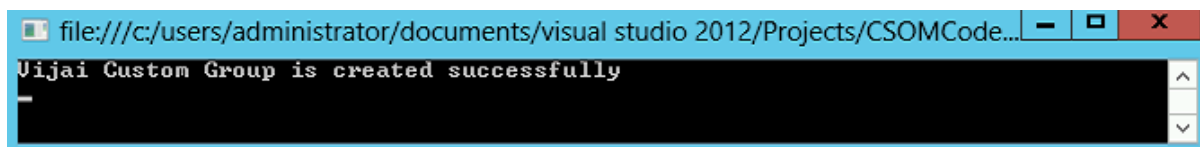


Figure12.2.1: Create a new site group

12.3 How to set the user as owner for the site group

In this example you will see how to set the user as owner for the site group using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;
            User ownerUser = web.EnsureUser("Vijai");

            // Set the user as owner for the site group
            Group group = web.SiteGroups.GetByName("Vijai Custom Group");
            group.Owner = ownerUser;

            // Update the group
            group.Update();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

CSOM EDIT LINKS

People and Groups > Change Group Settings ⓘ

Name and About Me Description
Type a name and description for the group.

Name:
Vijai Custom Group x

About Me:

Custom group created using CSOM

Owner
The owner can change anything about the group such as adding and removing members or deleting the group. Only one user or group can be the owner.

Group owner:
Vijai x

Figure12.3.1: Set the user as owner for the site group

12.4 How to set the group as owner for the site group

In this example you will see how to set the group as owner for the site group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
```



```
class Program
{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Get a specific site group by name
        Group ownerGroup = web.SiteGroups.GetByName("CSOM Owners");

        // Set the group as owner for the site group
        Group group = web.SiteGroups.GetByName("Vijai Custom Group");
        group.Owner = ownerGroup;

        // Update the group
        group.Update();

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}
```

Output

CSOM EDIT LINKS

People and Groups ▸ Change Group Settings ⓘ

Name and About Me Description
Type a name and description for the group.

Name:
Vijai Custom Group x

About Me:

Custom group created using CSOM

Owner
The owner can change anything about the group such as adding and removing members or deleting the group. Only one user or group can be the owner.

Group owner:
CSOM Owners x

Figure12.4.1: Set the group as owner for the site group

12.5 How to get all the users from the site group

In this example you will see how to get all the users from the site group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
```

```

{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the SharePoint web
        Web web = clientContext.Web;

        // Get the specific site group by name
        Group group = web.SiteGroups.GetByName("Vijai Custom Group");

        // Get all the users who belongs to this specific group
        UserCollection userColl = group.Users;
        clientContext.Load(userColl);

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Loop through all the users
        foreach (User user in userColl)
        {
            // Display the login name of the user
            Console.WriteLine(user.LoginName);
        }
        Console.ReadLine();
    }
}

```

Output

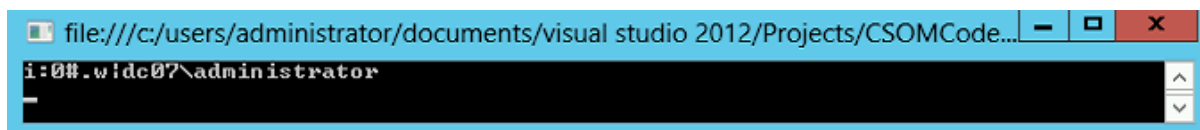


Figure12.5.1: Get all the users from the site group

12.6 How to add a user to the site group

In this example you will see how to add a user to the site group using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;
            User user = web.EnsureUser("Vijai");

            // Get the specific site group by name
            Group group = web.SiteGroups.GetByName("CSOM Owners");

            // Add a user to the specific group
            group.Users.AddUser(user);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

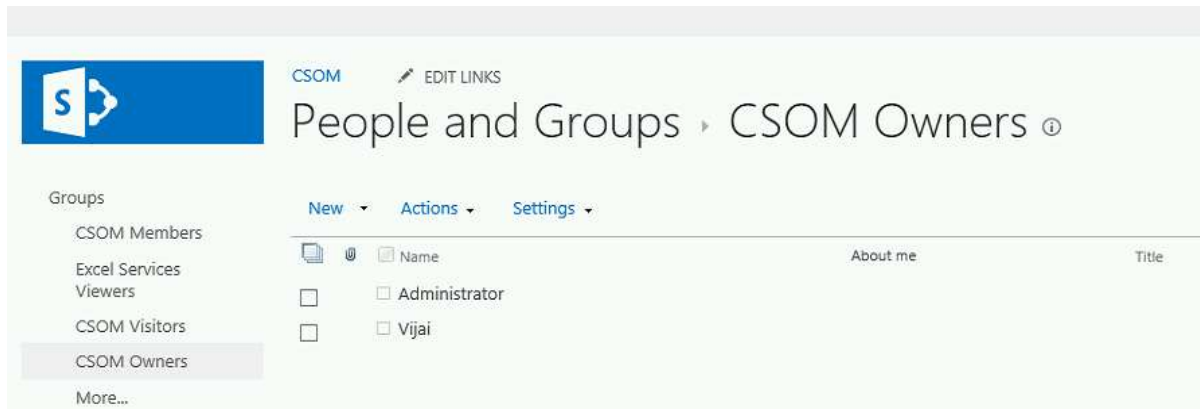


Figure12.6.1: Add user to the group

12.7 How to remove a user from the site group

In this example you will see how to remove a user from the site group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
```

```

Web web = clientContext.Web;
User user = web.EnsureUser("Vijai");
clientContext.Load(user);

// Execute the query to the server
clientContext.ExecuteQuery();

// Get the specific site group by name
Group group = web.SiteGroups.GetByName("CSOM Owners");

// Remove a user from the specific group
group.Users.RemoveByLoginName(user.LoginName);

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}

```

Output

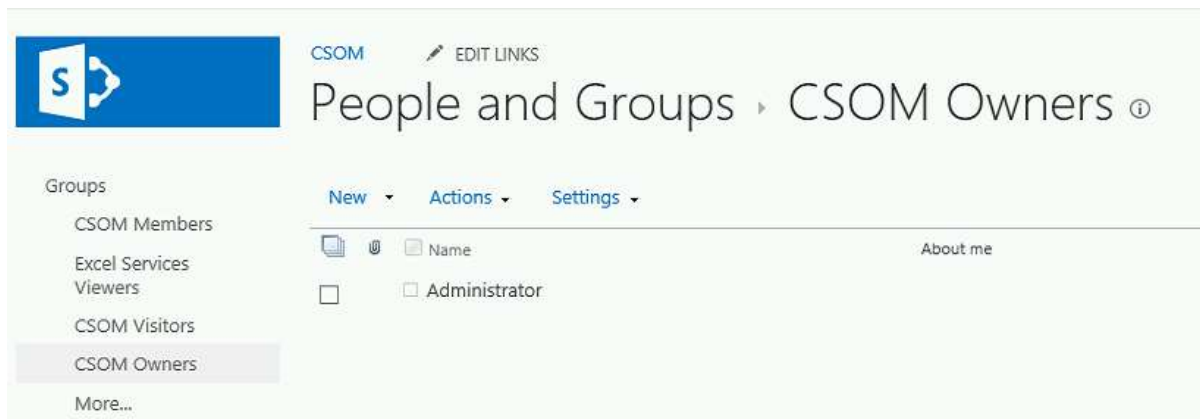


Figure12.7.1: Remove user from the group

12.8 How to delete a site group

In this example you will see how to delete a site group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using System.IO;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the specific site group by name
            Group group = web.SiteGroups.GetByName("Vijai Custom Group");

            // Remove a site group
            web.SiteGroups.Remove(group);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The specified site group is deleted successfully.

13 Perform SharePoint role tasks using CSOM

In this section you will see how to perform role related tasks using the SharePoint 2013 .Net Client Side Object Model.

13.1 How to get all the permission levels from the website

In this example you will see how to get all the roles or permission levels from the website using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get all the permission levels
            RoleDefinitionCollection roleDefColl = web.RoleDefinitions;
            clientContext.Load(roleDefColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```



```

    // Loop through all the permission levels
    foreach (RoleDefinition roleDef in roleDefColl)
    {
        // Display the names of the permission levels
        Console.WriteLine(roleDef.Name);
    }

    Console.ReadLine();
}
}
}

```

Output

Navigate to the SharePoint site. Click on the "Settings" and then Click on the "Site Settings". Click on the "Site Permissions" available under the "Users and Permissions" section. Click on the "Permission Levels" available in the ribbon interface. You will see all the permission levels available in the website.



Figure13.1.1: Get all the roles

13.2 How to create a permission level in the website

In this example you will see how to create a new role or permission level in the website using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Permissions that has to be added to the role definition
            BasePermissions permissions = new BasePermissions();
            permissions.Set(PermissionKind.ViewListItems);
            permissions.Set(PermissionKind.ViewVersions);

            // Contains properties that are used as parameters to initialize a role
definition
            RoleDefinitionCreationInformation creationInfo=new
RoleDefinitionCreationInformation ();
            creationInfo.Name = "My Role";
            creationInfo.Description = "Role created by me using CSOM";
            creationInfo.BasePermissions = permissions;

            // Add the role definition to the website
            web.RoleDefinitions.Add(creationInfo);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to the SharePoint site. Click on the "Settings" and then Click on the "Site Settings". Click on the "Site Permissions" available under the "Users and Permissions" section. Click on the "Permission Levels" available in the ribbon interface. You will see that a new role or permission level has been created successfully as shown in Figure 13.2.1.

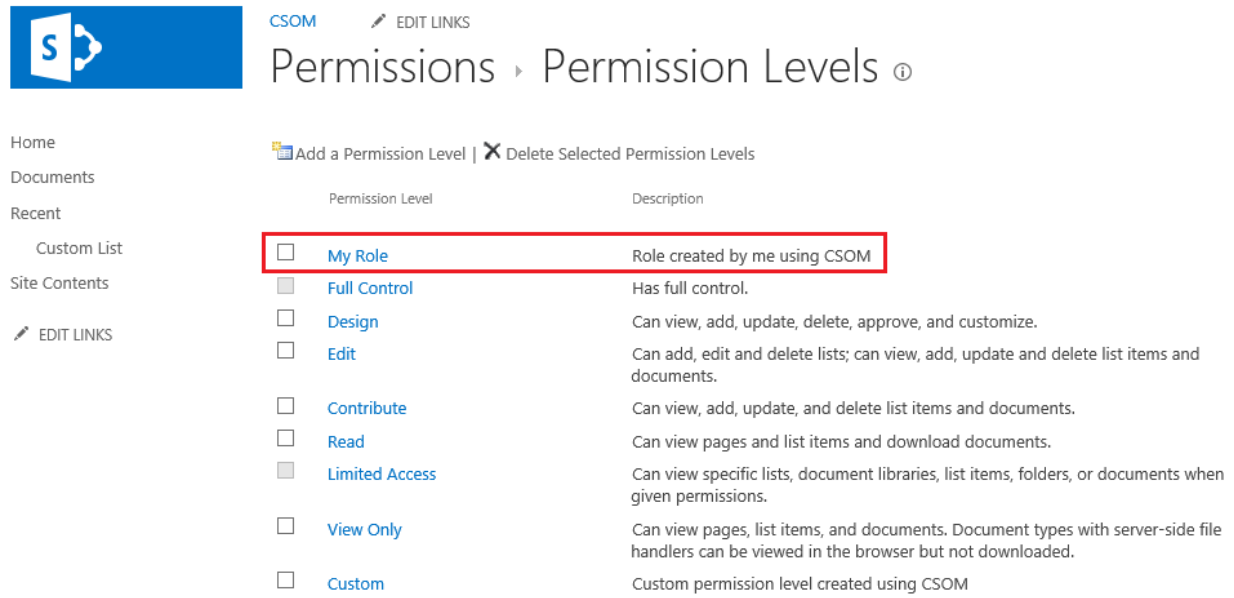


Figure13.2.1: Create a new role

13.3 How to update the permission level in the website

In this example you will see how to update the permission level in the website using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
```

```

{
    // ClientContext - Get the context for the SharePoint Site
    // SharePoint site URL - http://servername/sites/CSOM
    ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

    // Get the SharePoint web
    Web web = clientContext.Web;

    // Get the specific permission level by name
    RoleDefinition roleDef = web.RoleDefinitions.GetByName("Custom");

    // Set the permission level description
    roleDef.Description = "Custom permission level created using CSOM";

    // Add the permissions
    BasePermissions permissions = new BasePermissions();
    permissions.Set(PermissionKind.ApproveItems);
    permissions.Set(PermissionKind.CreateAlerts);
    roleDef.BasePermissions = permissions;

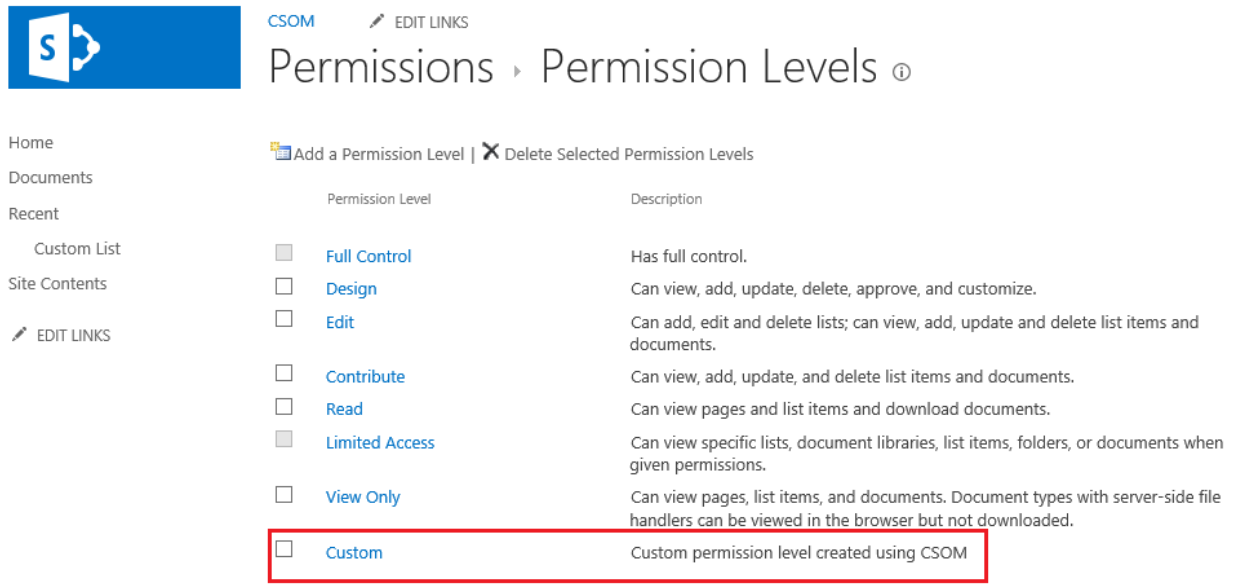
    // Update the permission level
    roleDef.Update();
    clientContext.Load(roleDef);

    // Execute the query to the server
    clientContext.ExecuteQuery();
}
}
}

```

Output

Navigate to the SharePoint site. Click on the "Settings" and then Click on the "Site Settings". Click on the "Site Permissions" available under the "Users and Permissions" section. Click on the "Permission Levels" available in the ribbon interface. You will see that the specified role or permission level has been updated successfully as shown in Figure 13.3.2.



The screenshot shows the SharePoint 2013 interface for managing permissions. The top navigation bar includes the SharePoint logo, the text 'CSOM', and an 'EDIT LINKS' button. The main heading is 'Permissions > Permission Levels'. On the left, a sidebar contains links for 'Home', 'Documents', 'Recent', 'Custom List', 'Site Contents', and another 'EDIT LINKS' button. The main content area has two tabs: 'Add a Permission Level' (active) and 'Delete Selected Permission Levels'. Below the tabs is a table with two columns: 'Permission Level' and 'Description'.

Permission Level	Description
<input type="checkbox"/> Full Control	Has full control.
<input type="checkbox"/> Design	Can view, add, update, delete, approve, and customize.
<input type="checkbox"/> Edit	Can add, edit and delete lists; can view, add, update and delete list items and documents.
<input type="checkbox"/> Contribute	Can view, add, update, and delete list items and documents.
<input type="checkbox"/> Read	Can view pages and list items and download documents.
<input type="checkbox"/> Limited Access	Can view specific lists, document libraries, list items, folders, or documents when given permissions.
<input type="checkbox"/> View Only	Can view pages, list items, and documents. Document types with server-side file handlers can be viewed in the browser but not downloaded.
<input type="checkbox"/> Custom	Custom permission level created using CSOM

Figure13.3.1: Update the role

List Permissions

- ☐ Manage Lists - Create and delete lists, add or remove columns in a list, and add or remove public views of a list.
- ☐ Override List Behaviors - Discard or check in a document which is checked out to another user, and change or override settings which allow users to read/edit only their own items
- ☐ Add Items - Add items to lists and add documents to document libraries.
- ☐ Edit Items - Edit items in lists, edit documents in document libraries, and customize Web Part Pages in document libraries.
- ☐ Delete Items - Delete items from a list and documents from a document library.
- ☐ View Items - View items in lists and documents in document libraries.
- ☒ Approve Items - Approve a minor version of a list item or document.
- ☐ Open Items - View the source of documents with server-side file handlers.
- ☐ View Versions - View past versions of a list item or document.
- ☐ Delete Versions - Delete past versions of a list item or document.
- ☒ Create Alerts - Create alerts.

Figure13.3.2: Update the permissions

13.4 How to remove the permissions from the permission level

In this example you will see how to remove the permissions from the permission level using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the specific permission level by name
            RoleDefinition roleDef = web.RoleDefinitions.GetByName("Custom");

            // Remove the permissions
            BasePermissions permissions = new BasePermissions();
            permissions.Clear(PermissionKind.ApproveItems);
            roleDef.BasePermissions = permissions;

            // Update the permission level
            roleDef.Update();
            clientContext.Load(roleDef);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to the SharePoint site. Click on the **Settings** and then Click on the **Site Settings**. Click on the **Site Permissions** which is available under **Users and Permissions** section. Click on the **Permission Levels** which is available in the ribbon interface. Click on the **Custom** permission level. You will be able to see the permissions are removed successfully from the role.

13.5 How to delete the permission level from the website

In this example you will see how to delete the role or permission level from the website using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the specific permission level by name
            RoleDefinition roleDef = web.RoleDefinitions.GetByName("Custom");

            // Delete the permission level
            roleDef.DeleteObject();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

Navigate to the SharePoint site. Click on the "Settings" and then Click on the "Site Settings". Click on the "Site Permissions" available under the "Users and Permissions" section. Click on the "Permission Levels" available in the ribbon interface. You will see that the specified role or permission level has been successfully deleted from the website.

13.6 How to assign the role to a specific group

In this example you will see how to assign the role to a specific group using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the site group by name for which a role should be assigned
            Group group = web.SiteGroups.GetByName("CSOM Owners");

            // Get the specific permission level by name
            RoleDefinition roleDef = web.RoleDefinitions.GetByName("My Role");

            // Defines the role definitions that are bound to a role assignment object
            RoleDefinitionBindingCollection roleDefBindColl = new
```

```

RoleDefinitionBindingCollection(clientContext);
    roleDefBindColl.Add(roleDef);

    // Assign the role to the group
    RoleAssignment roleAssign = web.RoleAssignments.Add(group, roleDefBindColl);

    // Execute the query to the server
    clientContext.ExecuteQuery();
}
}
}

```

Output

Navigate to the SharePoint site. Click on the "Settings" and then Click on the "Site Settings". Click on the "Site Permissions" available under the "Users and Permissions" section. Click on the "Permission Levels" available in the ribbon interface. You will see that the role or permission level has been assigned to the specified group as shown in Figure 13.6.1.

<input type="checkbox"/>	Name	Type	Permission Levels
<input type="checkbox"/>	CSOM Members	SharePoint Group	Edit
<input type="checkbox"/>	CSOM Owners	SharePoint Group	My Role, Full Control
<input type="checkbox"/>	CSOM Visitors	SharePoint Group	Read
<input type="checkbox"/>	Excel Services Viewers	SharePoint Group	View Only

Figure 13.6.1: Assign the role to a specific group

13.7 How to assign the role to a specific user

In this example you will see how to assign the role to a specific user using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SharePoint web
            Web web = clientContext.Web;

            // Get the user for whom a role should be assigned
            User user = web.EnsureUser("dc07\\pai");

            // Get the specific permission level by name
            RoleDefinition roleDef = web.RoleDefinitions.GetByName("My Role");

            // Defines the role definitions that are bound to a role assignment object
            RoleDefinitionBindingCollection roleDefBindColl = new
RoleDefinitionBindingCollection(clientContext);
            roleDefBindColl.Add(roleDef);

            // Assign the role to a user
            RoleAssignment roleAssign = web.RoleAssignments.Add(user, roleDefBindColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to the SharePoint site. Click on the "Settings" and then Click on the "Site Settings". Click on the "Site Permissions" available under the "Users and Permissions" section. Click on the "Permission Levels" available in the ribbon interface. You will see that the role or permission level has been assigned to the specified user as shown in Figure 13.7.1.

<input type="checkbox"/>	<input type="checkbox"/> Name	Type	Permission Levels
<input type="checkbox"/>	<input type="checkbox"/> CSOM Members	SharePoint Group	Edit
<input type="checkbox"/>	<input type="checkbox"/> CSOM Owners	SharePoint Group	My Role, Full Control
<input type="checkbox"/>	<input type="checkbox"/> CSOM Visitors	SharePoint Group	Read
<input type="checkbox"/>	<input type="checkbox"/> Excel Services Viewers	SharePoint Group	View Only
<input type="checkbox"/>	<input type="checkbox"/> Pai	User	My Role

Figure13.7.1: Assign the role to a specific user

14 Perform SharePoint Taxonomy related tasks using CSOM

In this section you will see how to perform taxonomy related tasks using the SharePoint 2013 .Net Client Side Object Model.

14.1 How to get all the Term Stores for the provided site

In this example you will see how to get all the termstores using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
            TaxonomySession.GetTaxonomySession(clientContext);

            // Retrieve all the term stores (Name property alone)
            clientContext.Load(
                taxonomySession.TermStores,
                termStores => termStores.Include
                    (termStore => termStore.Name)
            );
        }
    }
}
```

```

//Execute the query to the server
clientContext.ExecuteQuery();

// Check if the TaxonomySession is not null
if (taxonomySession != null)
{
    Console.WriteLine("Termstores available for the taxonomy session:");
    // Loop through all the term stores for the taxonomy session
    foreach (TermStore termStore in taxonomySession.TermStores)
    {
        // Print the termstore name
        Console.WriteLine(termStore.Name);
    }
    Console.ReadLine();
}
}
}
}

```

Output

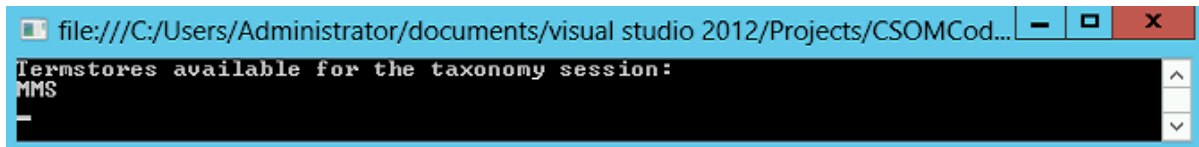


Figure14.1.1: Get all the termstores

14.2 How to get the Default site collection Termstore

In this example you will see how to get the default site collection termstore using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the default site collection term store
            TermStore termStore = taxonomySession.GetDefaultSiteCollectionTermStore();

            // Retrieve default site collection term store
            clientContext.Load(termStore);

            //Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the default site collection termstore
            Console.WriteLine("The default site collection termstore:
"+termStore.Name.ToString());
            Console.ReadLine();
        }
    }
}

```

Output

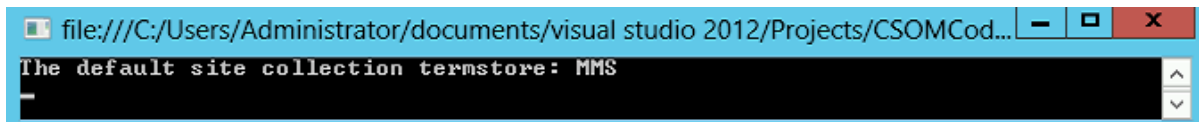


Figure14.2.1: Get the default site collection termstore

14.3 How to get the default keyword termstore

In this example you will see how to get the default keyword termstore using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the default keyword term store
            TermStore termStore = taxonomySession.GetDefaultKeywordsTermStore();

            // Retrieve default site collection term store
            clientContext.Load(termStore);

            //Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the default keyword termstore
            Console.WriteLine("The default keyword termstore:
"+termStore.Name.ToString());
        }
    }
}
```



```

        Console.ReadLine();
    }
}

```

Output

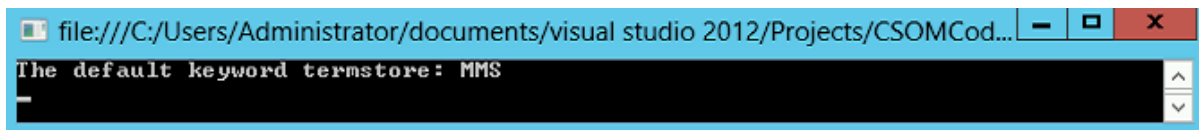


Figure14.3.1: Get the default keyword termstore

14.4 How to get all the groups for the termstore

In this example you will see how to get all the groups for the specific termstore using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

```

```

// Get the TaxonomySession
TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

// Get the term store by name
TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

// Get all the groups for the term store
clientContext.Load(termStore.Groups,
    termGroups => termGroups.Include
        (termGroup => termGroup.Name));

// Execute the query to the server
clientContext.ExecuteQuery();

// Loop through all the termgroup for the termstore
foreach (TermGroup group in termStore.Groups)
{
    // Display the group name
    Console.WriteLine(group.Name);
}
Console.ReadLine();
}
}
}

```

Output

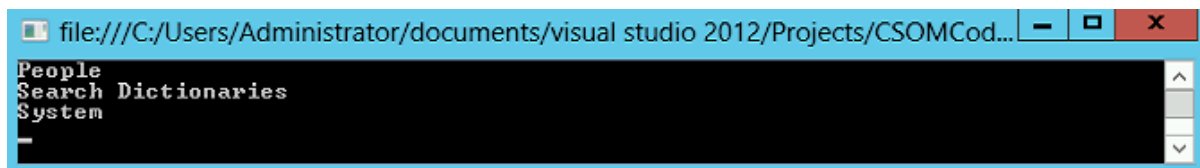


Figure14.4.1: Get all the taxonomy groups

14.5 How to create a new group for the term store

In this example you will see how to create a new taxonomy group for the termstore using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.

c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
            TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Create a new guid for group
            Guid guid = Guid.NewGuid();

            // Create a new group
            TermGroup termGroup = termStore.CreateGroup("NewGroup", guid);

            // Commit all the changes
            termStore.CommitAll();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

A new taxonomy group is created successfully as shown in Figure 14.5.1.

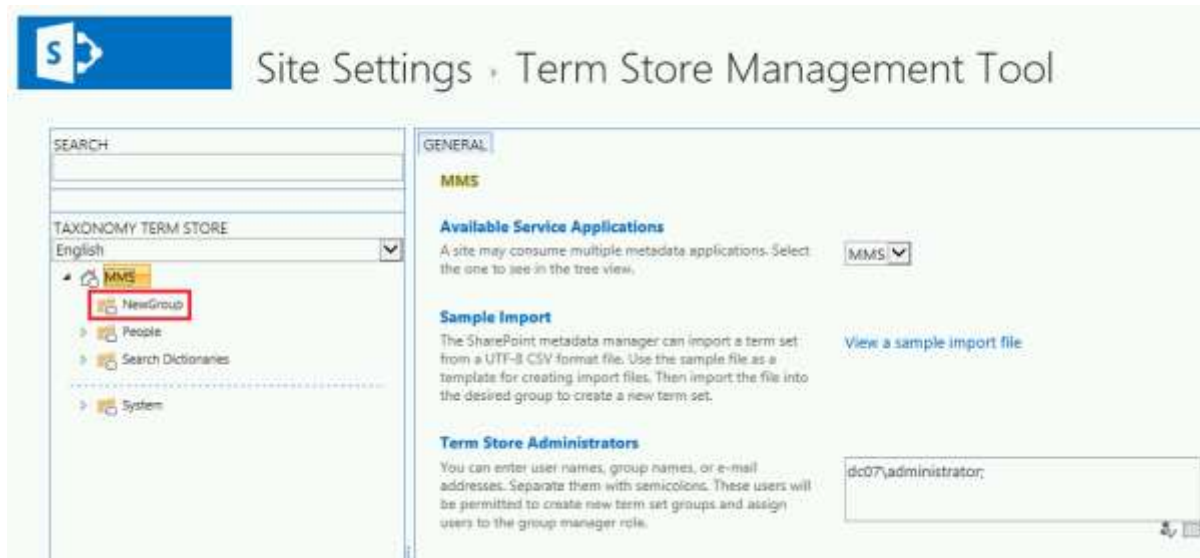


Figure14.5.1: Create a new taxonomy group

14.6 How to delete the group from the term store

In this example you will see how to delete a specific taxonomy group from the termstore using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
```

```

static void Main(string[] args)
{
    // ClientContext - Get the context for the SharePoint Site
    // SharePoint site URL - http://servername/sites/CSOM
    ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

    // Get the TaxonomySession
    TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

    // Get the term store by name
    TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

    // Group GUID
    Guid guid = new Guid("4bdbf81f-d3c4-4ae9-87f8-f8057e1b1a0f");

    // Get the term group using GUID
    TermGroup termGroup = termStore.GetGroup(guid);

    // Delete the term group
    termGroup.DeleteObject();

    // Execute the query to the server
    clientContext.ExecuteQuery();
}
}
}

```

Output

The specified taxonomy group is deleted successfully from the termstore.

14.7 How to get all the termsets for the taxonomy group

In this example you will see how to get all the termsets for the taxonomy group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Group GUID
            Guid guid = new Guid("49e1e372-88d4-46db-9454-f50913be03cc");

            // Get the term group using GUID
            TermGroup termGroup = termStore.GetGroup(guid);

            // Get all the term sets
            TermSetCollection termSetColl = termGroup.TermSets;

            clientContext.Load(termSetColl);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Loop through all the termsets
            foreach (TermSet termSet in termSetColl)
            {
                // Display the term set name
                Console.WriteLine(termSet.Name);
            }

            Console.ReadLine();
        }
    }
}

```

```
}
}
```

Output

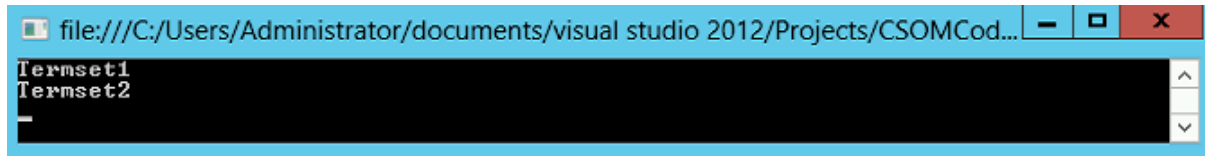


Figure14.7.1: Get all the termsets

14.8 How to create a term set for the specified group

In this example you will see how to create a new termset for the specified group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");
```

```

        // Get the TaxonomySession
        TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

        // Get the term store by name
        TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

        // Group GUID
        Guid guid = new Guid("49e1e372-88d4-46db-9454-f50913be03cc");

        // Get the term group using GUID
        TermGroup termGroup = termStore.GetGroup(guid);

        // New Term Set Name
        string termSetName = "NewTermSet";

        // New Term Set Guid
        Guid newTermSetGuid = Guid.NewGuid();

        // New Term Set LCID
        int lcid = 1033;

        // Create a new term set
        TermSet termSetColl = termGroup.CreateTermSet(termSetName, newTermSetGuid,
lcid);

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

A new termset is created successfully for the specified taxonomy group.

14.9 How to delete the term set from the specified group

In this example you will see how to delete the termset from the specified group using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
            TermSet termSet = termGroup.TermSets.GetByName("NewTermSet");

            // Delete the term set
            termSet.DeleteObject();

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The termset is deleted successfully from the specified taxonomy group.

14.10 How to get all the terms for the termset

In this example you will see how to get all the terms for the termset using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
            TermSet termSet = termGroup.TermSets.GetByName("Termset1");

            // Get all the terms
            TermCollection termColl = termSet.Terms;
```

```

        clientContext.Load(termColl);

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Loop through all the terms
        foreach (Term term in termColl)
        {
            // Display the term name
            Console.WriteLine(term.Name);
        }
        Console.ReadLine();
    }
}

```

Output

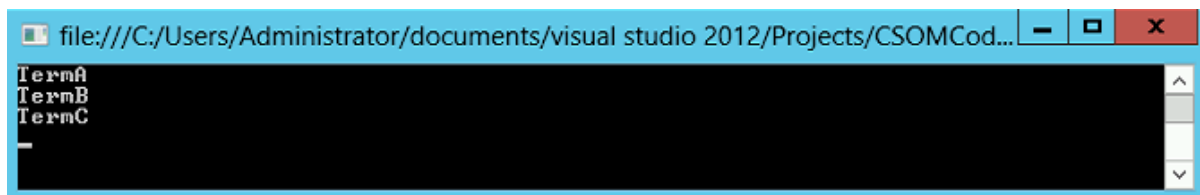


Figure14.10.1: Get all the terms

14.11 How to create a new term for the termset

In this example you will see how to create a new term for the termset using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;

```

```

using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
            TermSet termSet = termGroup.TermSets.GetByName("Termset1");

            // String variable - new term name
            string termName = "New Term";

            // Int variable - new term lcid
            int lcid = 1033;

            // Guid - new term guid
            Guid newTermId = Guid.NewGuid();

            // Create a new term
            Term newTerm = termSet.CreateTerm(termName, lcid, newTermId);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

A new term is created successfully for the specified termset.

14.12 How to delete the term from the term set

In this example you will see how to delete the term from the termset using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
            TermSet termSet = termGroup.TermSets.GetByName("Termset1");

            // Get the term by Name
            Term term = termSet.Terms.GetByName("New Term");
```

```

        // Delete the term
        term.DeleteObject();

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

The specified term is deleted successfully from the termset.

14.13 How to create a copy of the term within the termset

In this example you will see how to delete the term from the termset using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession

```

```

TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

// Get the term store by name
TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

// Get the term group by Name
TermGroup termGroup = termStore.Groups.GetByName("Group");

// Get the term set by Name
TermSet termSet = termGroup.TermSets.GetByName("Termset1");

// Get the term by Name
Term term = termSet.Terms.GetByName("TermA");

// Make a copy of the term within the termset
// Need to pass a bool parameter - whether to copy the child terms or not
term.Copy(false);

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}

```

Output

The specified term is copied to the same termset as shown in Figure 14.11.1.

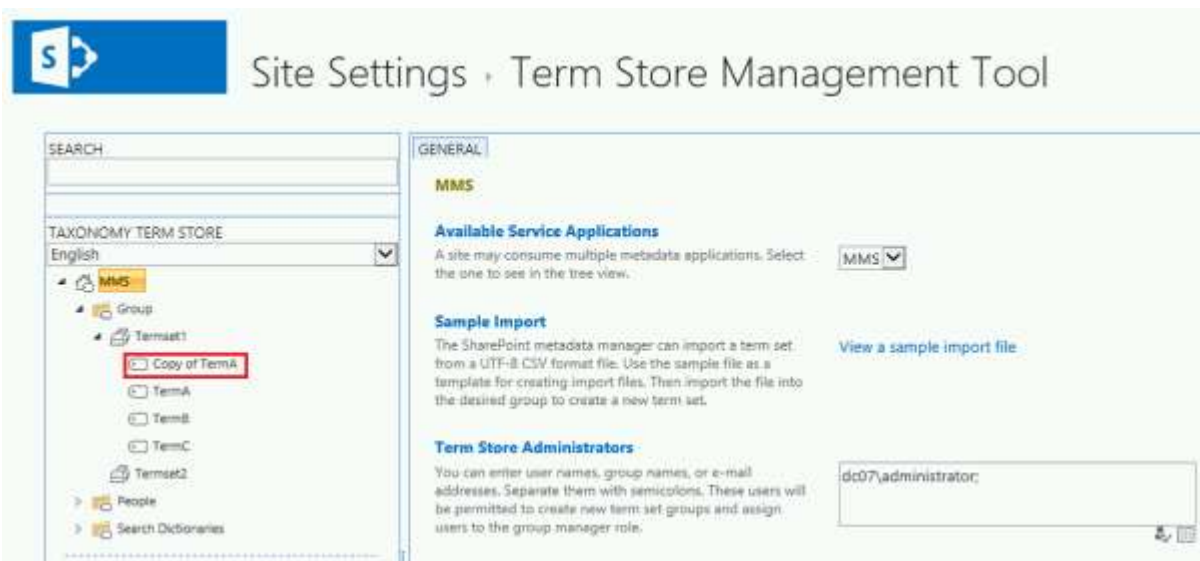


Figure 14.11.1: Create a copy of the term

14.14 How to move the specified term to different termset

In this example you will see how to move the specified term to a different termset using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
            TermSet termSet = termGroup.TermSets.GetByName("Termset1");
```



```

// Get the term by Name that has to be moved
Term moveTerm = termSet.Terms.GetByName("TermA");

// Get the destination where the term has to be moved
TermSet destTermSet = termGroup.TermSets.GetByName("Termset2");

// Move the term to the destination location
moveTerm.Move(destTermSet);

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}

```

Output

The specified term is moved to the other termset as shown in Figure 14.14.1.

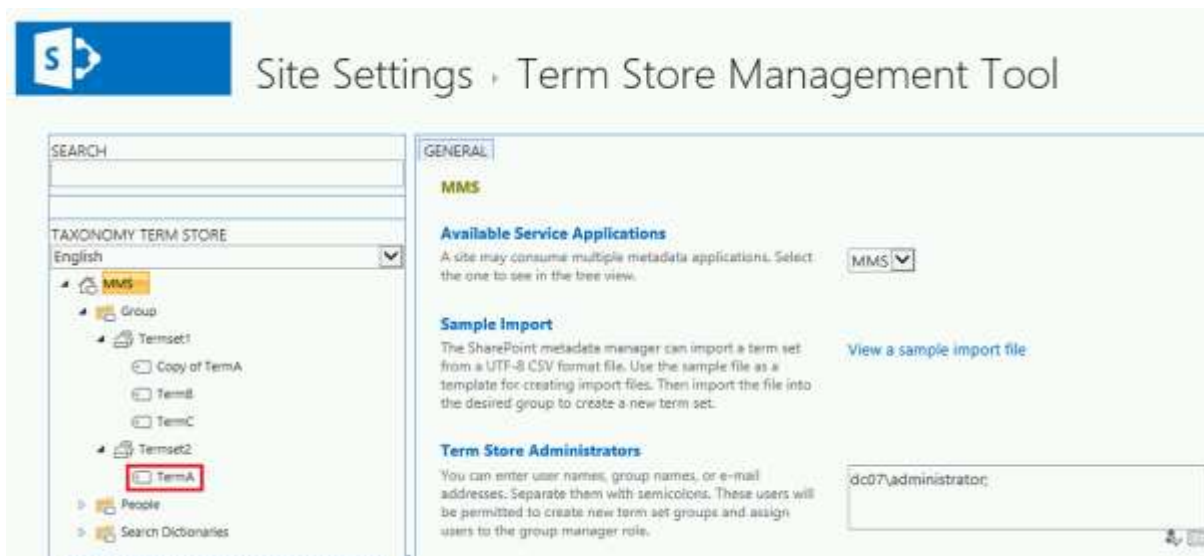


Figure 14.14.1: Move the term to different termset.

14.15 How to deprecate the specified term

In this example you will see how to deprecate the specified term using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.

c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
            TermSet termSet = termGroup.TermSets.GetByName("Termset1");

            // Get the term by Name that has to be deprecated
            Term term = termSet.Terms.GetByName("TermB");

            // Deprecate the term
            term.Deprecate(true);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}
```

Output

The specified term is deprecated as shown in Figure 14.15.1.

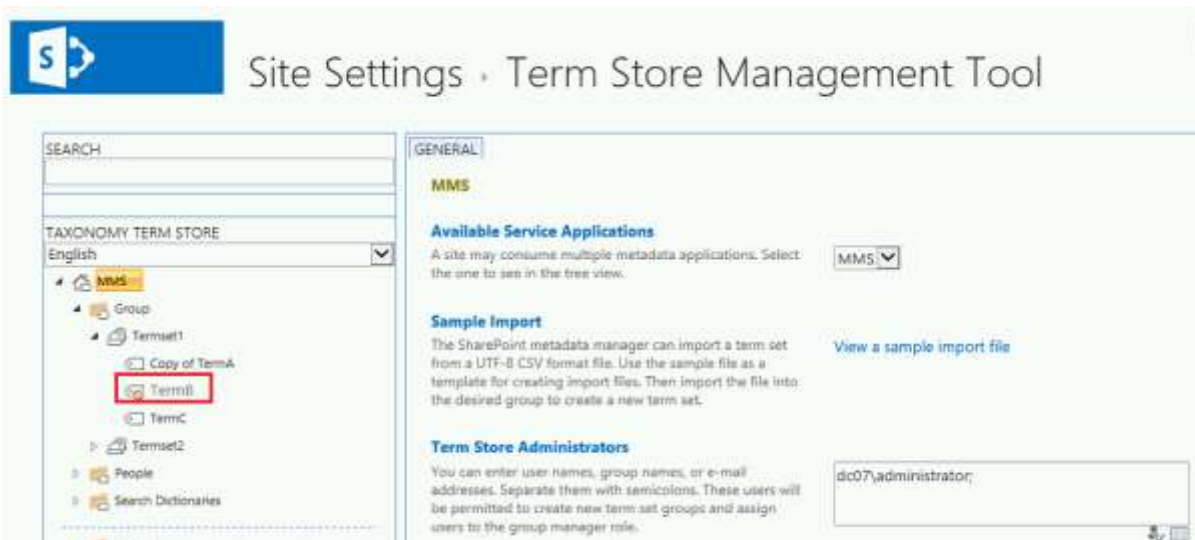


Figure 14.15.1: Deprecate the specified term

14.16 How to get all the labels for specified term

In this example you will see how to get all the labels for the specified term using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
```

```

class Program
{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Get the TaxonomySession
        TaxonomySession taxonomySession =
TaxonomySession.GetTaxonomySession(clientContext);

        // Get the term store by name
        TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

        // Get the term group by Name
        TermGroup termGroup = termStore.Groups.GetByName("Group");

        // Get the term set by Name
        TermSet termSet = termGroup.TermSets.GetByName("Termset1");

        // Get the term by Name
        Term term = termSet.Terms.GetByName("TermC");

        // Get all the labels for the term
        LabelCollection labelColl = term.Labels;

        clientContext.Load(labelColl);

        // Execute the query to the server
        clientContext.ExecuteQuery();

        // Loop through all the labels
        foreach (Label label in labelColl)
        {
            // Display the label value
            Console.WriteLine(label.Value);
        }

        Console.ReadLine();
    }
}

```

Output

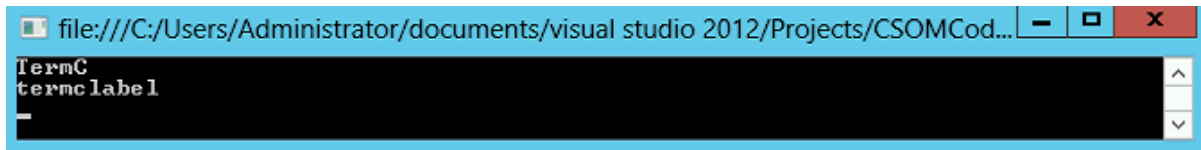


Figure14.16.1: Get all the labels

14.17 How to create a new label for specified term

In this example you will see how to create a new label for the specified term using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
            TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");
```

```

// Get the term group by Name
TermGroup termGroup = termStore.Groups.GetByName("Group");

// Get the term set by Name
TermSet termSet = termGroup.TermSets.GetByName("Termset1");

// Get the term by Name
Term term = termSet.Terms.GetByName("TermC");

// string variable - Label Name
string labelName = "newlabel";

// int variable- LCID
int lcid = 1033;

// bool variable - isDefault
bool isDefault = false;

// Create a new term for the term
Label label = term.CreateLabel(labelName, lcid, isDefault);

// Execute the query to the server
clientContext.ExecuteQuery();
}
}
}

```

Output

The screenshot displays the SharePoint 2013 Taxonomy Term Store interface. On the left, the 'TAXONOMY TERM STORE' pane shows a hierarchy: English > MMS > Group > Termset1 > TermC. The 'TermC' term is selected and highlighted. The main pane on the right shows the configuration for 'TermC' under the 'GENERAL' tab. The configuration includes:

- Available for Tagging:** A checkbox that is checked, indicating the term is available for tagging.
- Language:** A dropdown menu set to 'English'.
- Description:** A text area for describing the term.
- Default Label:** A text box containing 'TermC'.
- Other Labels:** A list of labels including 'newlabel', 'termclabel', and 'add new label'.

Figure14.17.1: *Create a new label*

14.18 How to delete the label for specified term

In this example you will see how to delete the label for a specified term using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.Taxonomy;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the TaxonomySession
            TaxonomySession taxonomySession =
            TaxonomySession.GetTaxonomySession(clientContext);

            // Get the term store by name
            TermStore termStore = taxonomySession.TermStores.GetByName("MMS");

            // Get the term group by Name
            TermGroup termGroup = termStore.Groups.GetByName("Group");

            // Get the term set by Name
```

```
TermSet termSet = termGroup.TermSets.GetByName("Termset1");

// Get the term by Name
Term term = termSet.Terms.GetByName("TermC");

// Get the label for the term
Label label = term.Labels.GetByValue("newlabel");

// Delete the label for the term
label.DeleteObject();

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}
```

Output

The specified label is deleted successfully for the term.

15 Perform SharePoint User Profile tasks using CSOM

In this section you will see how to perform user profile related tasks using the SharePoint 2013 .Net Client Side Object Model.

15.1 How to get a particular user's properties

In this example you will see how to get specific user properties using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            /// String Variable to store the account name
            string accountName = "dc07\\pai";

            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            /// PeopleManager class provides the methods for operations related to
people
            PeopleManager peopleManager = new PeopleManager(clientContext);

            /// PersonProperties class is used to represent the user properties
            /// GetPropertiesFor method is used to get the user properties for a
```

```

particular user
    PersonProperties personProperties =
peopleManager.GetPropertiesFor(accountName);
    clientContext.Load(personProperties);
    clientContext.ExecuteQuery();

    /// Display the user profile properties - AccountName, Email, DisplayName
    Console.WriteLine("Account Name: " + personProperties.AccountName);
    Console.WriteLine("Email: " + personProperties.Email);
    Console.WriteLine("Display Name: " + personProperties.DisplayName);
    Console.ReadLine();
}
}
}

```

Output

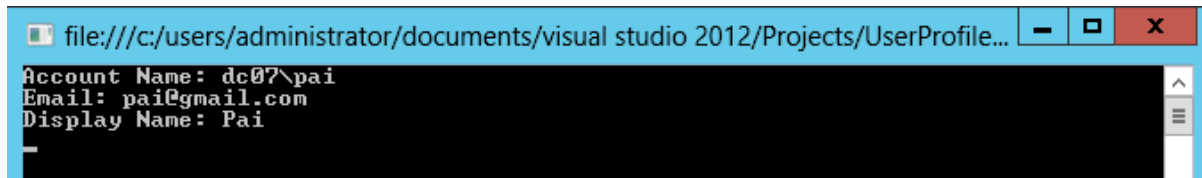


Figure15.1.1: Get the particular user properties

15.2 How to get the current user's properties

In this example you will see how to get the current user's properties using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;

```

```

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

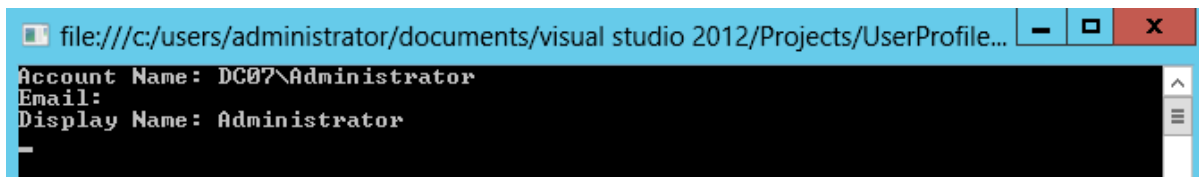
            //// PeopleManager class provides the methods for operations related to
people
            PeopleManager peopleManager = new PeopleManager(clientContext);

            //// PersonProperties class is used to represent the user properties
            //// GetMyProperties method is used to get the current user's properties
            PersonProperties personProperties = peopleManager.GetMyProperties();
            clientContext.Load(personProperties, p => p.AccountName, p => p.Email, p =>
p.DisplayName);
            clientContext.ExecuteQuery();

            //// Display the user properties - AccountName, Email, DisplayName
            Console.WriteLine("Account Name: " + personProperties.AccountName);
            Console.WriteLine("Email: " + personProperties.Email);
            Console.WriteLine("Display Name: " + personProperties.DisplayName);
            Console.ReadLine();
        }
    }
}

```

Output



```

file:///c:/users/administrator/documents/visual studio 2012/Projects/UserProfile...
Account Name: DC07\Administrator
Email:
Display Name: Administrator

```

Figure15.2.1: Get the current user properties

15.3 How to get the specific user profile property for a user

In this example you will see how to get the specific user's profile property for a user using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // String Variable to store the account name of the user for whom to get the
            // user profile property
            string accountName = "DC07\\pai";

            // String Variable to store the user profile property
            string userProfileProperty = "AccountName";

            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

            // PeopleManager class provides the methods for operations related to people
            PeopleManager peopleManager = new PeopleManager(clientContext);

            // GetUserProfilePropertyFor method is used to get a specific user profile
            // property for a user
            ClientResult<string> profileProperty =
            peopleManager.GetUserProfilePropertyFor(accountName, userProfileProperty);
            clientContext.ExecuteQuery();

            // Display the user profile property value for the user
            Console.WriteLine("Account Name: " + profileProperty.Value);
            Console.ReadLine();
        }
    }
}
```

Output

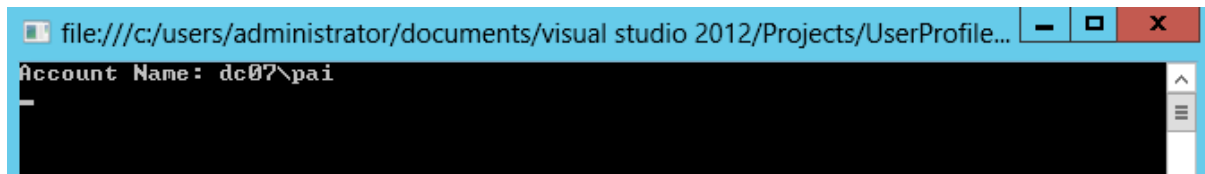
A screenshot of a Visual Studio console window. The title bar shows the file path: file:///c:/users/administrator/documents/visual studio 2012/Projects/UserProfile... The console output displays the text "Account Name: dc07\pai" on a black background with white text. The window has standard minimize, maximize, and close buttons.

Figure15.3.1: Get the specific user profile property

16 Perform SharePoint social tasks using CSOM

In this section you will see how to perform social related tasks using the SharePoint 2013 .Net Client Side Object Model.

16.1 How to determine whether the specified user is following the target user

In this example you will see how to determine whether the specified user is following the target user using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // String Variable to store the account name of the user
            string userAccountName = "DC07\\pai";

            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials for whom we need to check whether user is following
            the target user
```

```

//By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

// Get the SocialFollowingManager instance
// Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

// Create a SocialActorInfo object to represent the target user (Identifies a
user, document, site, or tag in social feed and following activities).
SocialActorInfo socialActorInfo = new SocialActorInfo();
socialActorInfo.AccountName = userAccountName;

// Check whether the specified user is following the target user
ClientResult<bool> isFollowed = followingManager.IsFollowed(socialActorInfo);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display whether the user is following the target user
Console.WriteLine("Is Ranjith following the target user? " +
isFollowed.Value);
Console.ReadLine();
    }
}
}

```

Output

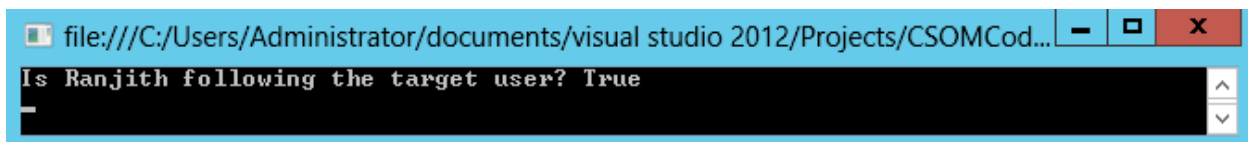


Figure16.1.1: Check the specified user is following the target user

16.2 How to get the count of people the user is following

In this example you will see how to get the count of people the specified user is following using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials for whom we need to get the count of followed users
            // By default It will take current user who runs this application - Here it
            // will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
            // documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Get the count of people who the user is following
            ClientResult<int> followingCount =
followingManager.GetFollowedCount(SocialActorTypes.Users);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the count of people who the user is following
            Console.WriteLine("Number of people the user is following: " +
```



```

followingCount.Value);
    Console.ReadLine();
}
}
}

```

Output

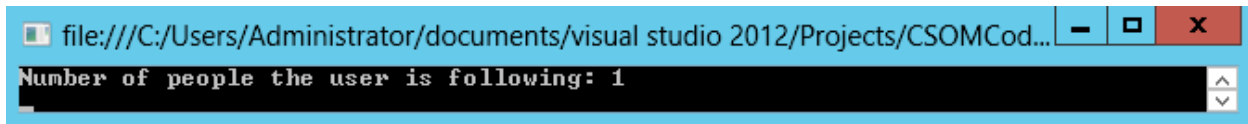


Figure 16.2.1: Count the number of people the user is following

16.3 How to get the count of documents followed by the user

In this example you will see how to get the count of documents followed by the specified user using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site

```

```

// SharePoint site URL - http://servername/sites/CSOM
ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

// Pass the credentials for whom we need to get the count of followed
documents
//By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

// Get the SocialFollowingManager instance
// Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

// Get the count of followed documents
ClientResult<int> followingCount =
followingManager.GetFollowedCount(SocialActorTypes.Documents);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display the count of followed documents
Console.WriteLine("Count of followed documents: " + followingCount.Value);
Console.ReadLine();
    }
}
}

```

Output

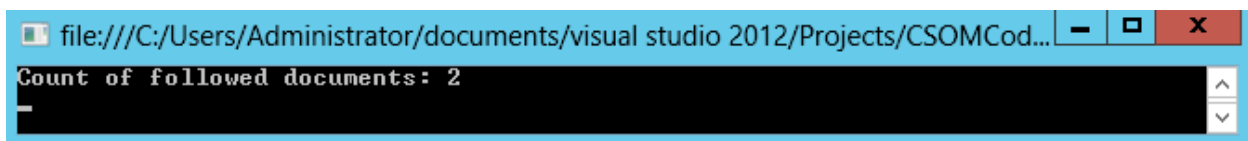


Figure16.3.1: Count of followed documents

16.4 How to get the count of sites followed by the user

In this example you will see how to get the count of sites followed by the specified user using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials for whom we need to get the count of followed sites
            //By default It will take current user who runs this application - Here it
            //will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
            // documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Get the count of followed sites
            ClientResult<int> followingCount =
followingManager.GetFollowedCount(SocialActorTypes.Sites);

            // Execute the query to the server
            clientContext.ExecuteQuery();

            // Display the count of followed sites
            Console.WriteLine("Count of followed sites: " + followingCount.Value);
        }
    }
}
```

```

        Console.ReadLine();
    }
}

```

Output

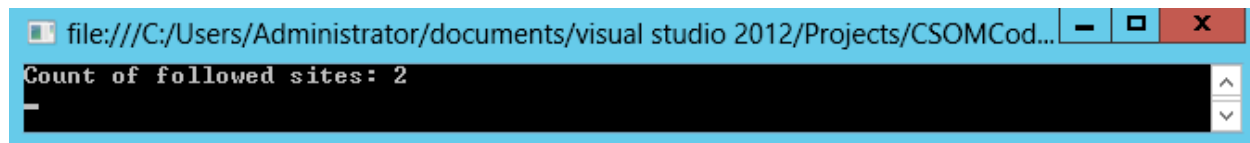


Figure16.4.1: Count of followed sites

16.5 How to get the count of actors followed by the user

In this example you will see how to get the count of actors (documents, sites, users and tags) followed by the specified user using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {

```

```

// ClientContext - Get the context for the SharePoint Site
// SharePoint site URL - http://servername/sites/CSOM
ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

// Pass the credentials for whom we need to get the count of all followed
actors (users, documents, sites and tags)
//By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

// Get the SocialFollowingManager instance
// Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

// Get the count of followed actors
ClientResult<int> followingCount =
followingManager.GetFollowedCount(SocialActorTypes.All);

// Execute the query to the server
clientContext.ExecuteQuery();

// Display the count of followed actors
Console.WriteLine("Count of followed actors (users, documents, tags and
sites): " + followingCount.Value);
Console.ReadLine();
    }
}
}

```

Output

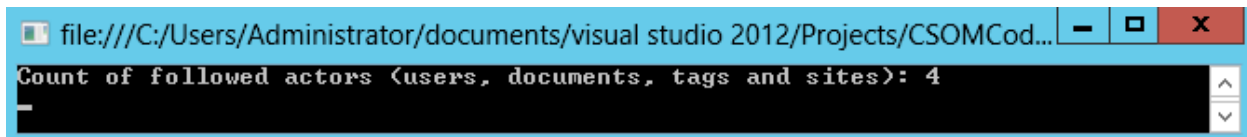


Figure16.5.1: Count of followed actors

16.6 How to get all the followers for the user

In this example you will see how to get all the followers for the specified user using the Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials for whom we need to get all the followers
            // By default It will take current user who runs this application - Here it
            // will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
            // documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Get all the followers
            ClientResult<SocialActor[]> followers = followingManager.GetFollowers();

            // Execute the query to the server
            clientContext.ExecuteQuery();
            Console.WriteLine("My Followers: ");
        }
    }
}
```

```

        // Loop through all the users
        foreach (SocialActor actor in followers.Value)
        {
            // Display the user account name
            Console.WriteLine(actor.AccountName);
        }
        Console.ReadLine();
    }
}

```

Output

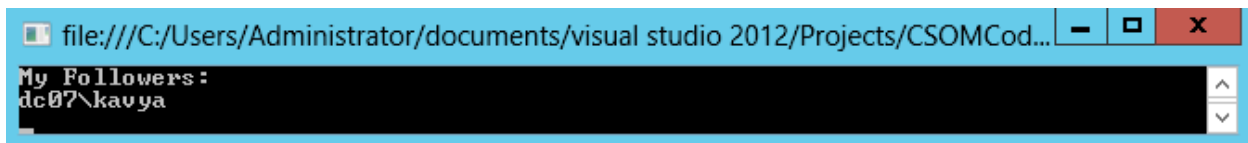


Figure16.6.1: Get all the followers

16.7 How to get all the people followed by the user

In this example you will see how to get all the people followed by the specified user using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{

```

```

class Program
{
    static void Main(string[] args)
    {
        // ClientContext - Get the context for the SharePoint Site
        // SharePoint site URL - http://servername/sites/CSOM
        ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

        // Pass the credentials for whom we need to get all the followed users
        // By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
        clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

        // Get the SocialFollowingManager instance
        // Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
        SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

        // Get all the followed users
        ClientResult<SocialActor[]> followedUsers =
followingManager.GetFollowed(SocialActorTypes.Users);

        // Execute the query to the server
        clientContext.ExecuteQuery();
        Console.WriteLine("I am following the below users: ");

        // Loop through all the users
        foreach (SocialActor actor in followedUsers.Value)
        {
            // Display the user account name
            Console.WriteLine(actor.AccountName);
        }
        Console.ReadLine();
    }
}

```

Output

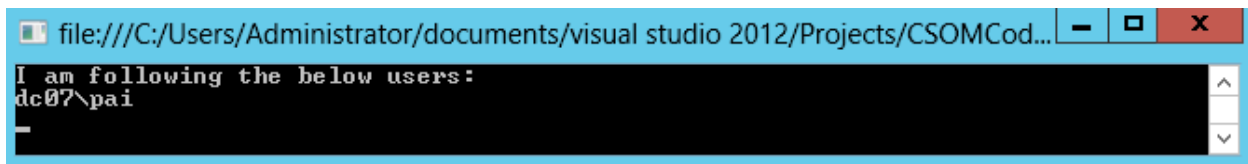


Figure16.7.1: Get all the people followed by the user

16.8 How to get all the documents followed by the user

In this example you will see how to get all the documents followed by the specified user using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials for whom we need to get all the followed documents
            // By default It will take current user who runs this application - Here it
            // will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
            // documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);
```

```

        // Get all the followed documents
        ClientResult<SocialActor[]> followedDocuments =
followingManager.GetFollowed(SocialActorTypes.Documents);

        // Execute the query to the server
        clientContext.ExecuteQuery();
        Console.WriteLine("I am following the below documents: ");

        // Loop through all the documents
        foreach (SocialActor actor in followedDocuments.Value)
        {
            // Display the document uri
            Console.WriteLine(actor.ContentUri);
        }
        Console.ReadLine();
    }
}

```

Output

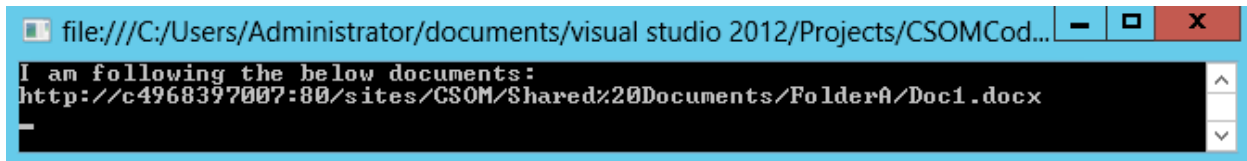


Figure16.8.1: Get all the followed documents

16.9 How to get all the sites followed by the user

In this example you will see how to get all the sites followed by the specified user using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials for whom we need to get all the followed sites
            // By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Get all the followed sites
            ClientResult<SocialActor[]> followedSites =
followingManager.GetFollowed(SocialActorTypes.Sites);

            // Execute the query to the server
            clientContext.ExecuteQuery();
            Console.WriteLine("I am following the below sites: ");

            // Loop through all the sites
            foreach (SocialActor actor in followedSites.Value)
            {
                // Display the site uri
                Console.WriteLine(actor.ContentUri);
            }
            Console.ReadLine();
        }
    }
}

```

Output



Figure16.9.1: Get all the followed sites

16.10 How to follow the target user

In this example you will see how to start following the target user using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // String Variable to store the account name of the target user
            string targetUser = "DC07\\Kavya";

            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
            ClientContext("http://servername/sites/CSOM/");

            // Get the SocialFollowingManager instance
```

```

        // Provides methods for managing a user's list of followed actors (users,
        documents, sites, and tags)
        SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

        // Identifies a user, document, site, or tag in social feed and following
activities
        SocialActorInfo actorInfo = new SocialActorInfo();
        actorInfo.ActorType = SocialActorType.User;
        actorInfo.AccountName = targetUser;

        // Follow the target user
        followingManager.Follow(actorInfo);

        // Execute the query to the server
        clientContext.ExecuteQuery();
    }
}

```

Output

Navigate to My Site and then Click on “Newsfeed”. You will see that the user has started following the target user.

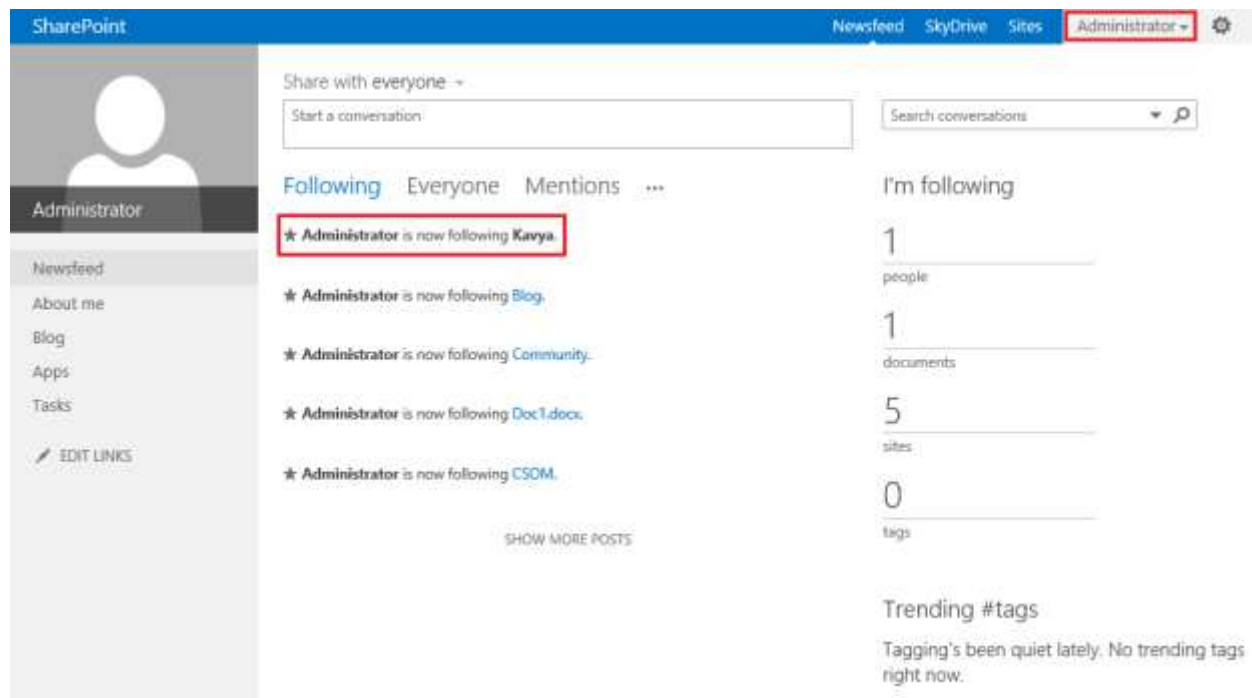


Figure16.10.1: Follow the user

16.11 How to stop following the target user

In this example you will see how to stop following the target user using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // String Variable to store the account name of the target user
            string targetUser = "DC07\\Kavya";

            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Identifies a user, document, site, or tag in social feed and following
activities
            SocialActorInfo actorInfo = new SocialActorInfo();
            actorInfo.ActorType = SocialActorType.User;
            actorInfo.AccountName = targetUser;
```

```

// Stop following the target user
followingManager.StopFollowing(actorInfo);

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}

```

Output

Navigate to My Site and then Click on “Newsfeed”. You will see that the user has stopped following the target user. The number of people following has been reduced.

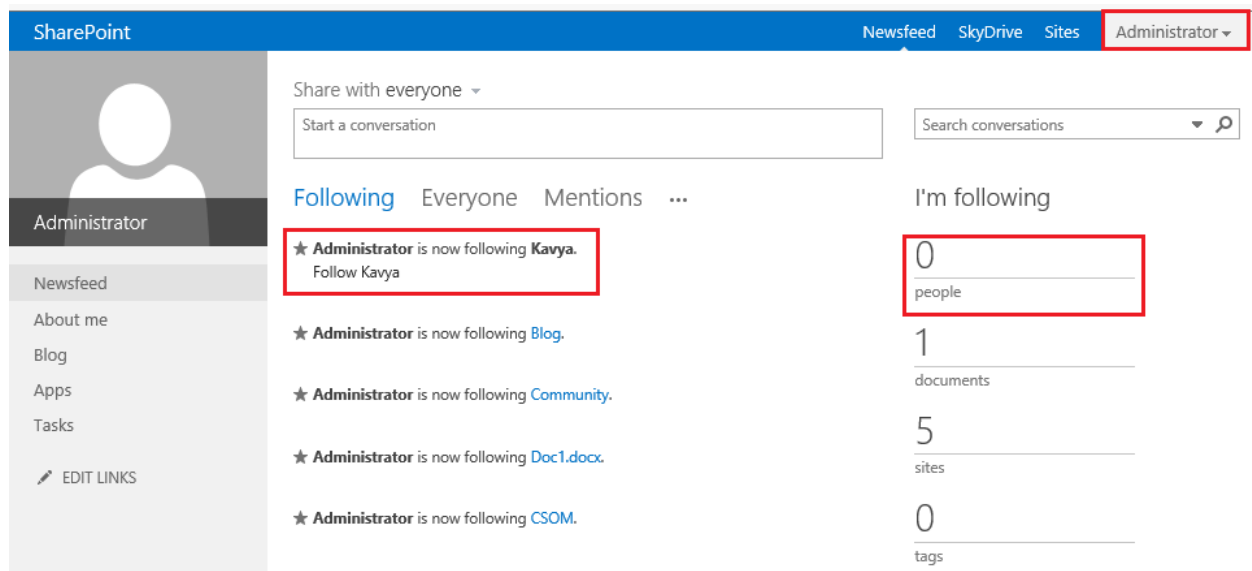


Figure16.11.1: Stop following the user

16.12 How to follow the site

In this example you will see how to follow the site using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials
            // By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Identifies a user, document, site, or tag in social feed and following
activities
            SocialActorInfo actorInfo = new SocialActorInfo();
            actorInfo.ActorType = SocialActorType.Site;
            actorInfo.ContentUri = "http://servername";

            // Follow the target site
            followingManager.Follow(actorInfo);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```


Output

Navigate to My Site and then Click on “Newsfeed”. You will see that the user has started following a site.

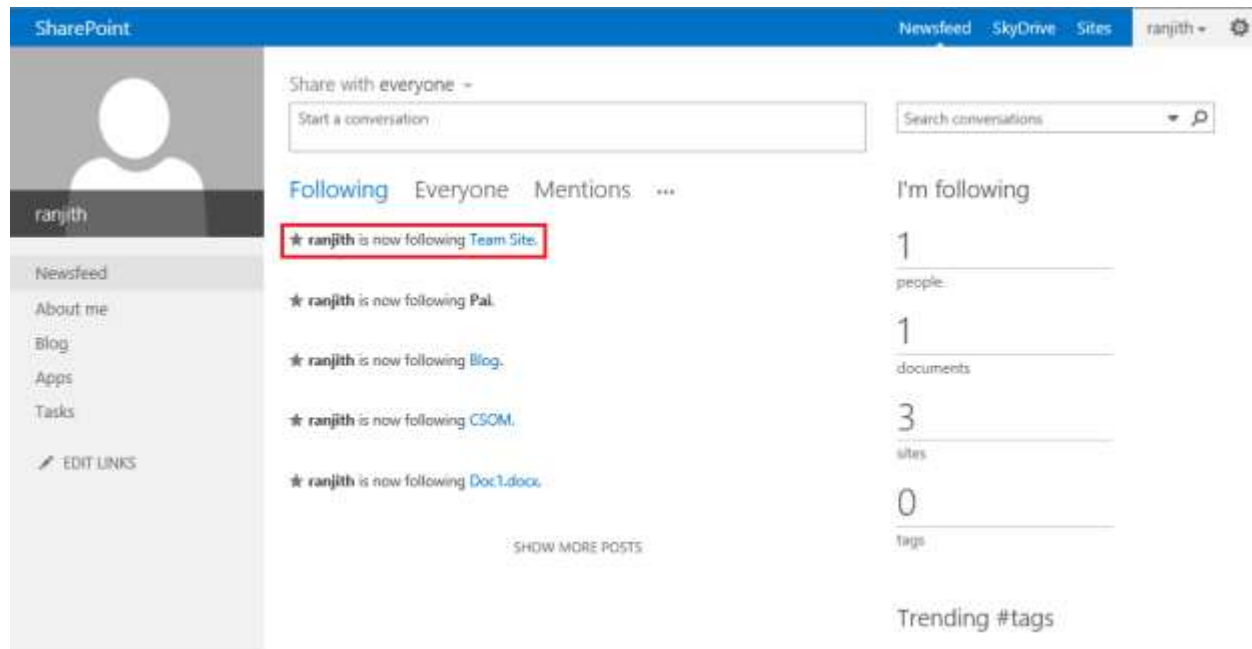


Figure16.12.1: Follow the site

Note

If the user doesn't have permission to the site then you will be getting the following error:

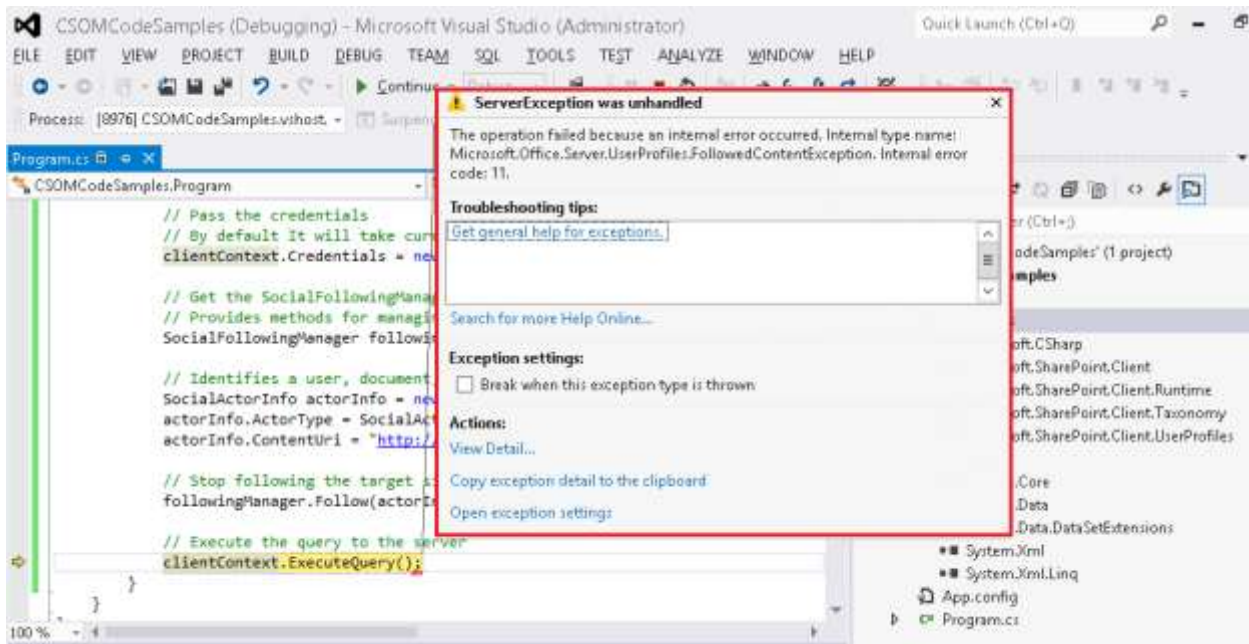


Figure16.12.2: Error Message

16.13 How to stop following the site

In this example you will see how to stop following the site using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials
            // By default It will take current user who runs this application - Here it
            // will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
            // documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);
```

```

// Identifies a user, document, site, or tag in social feed and following
activities
SocialActorInfo actorInfo = new SocialActorInfo();
actorInfo.ActorType = SocialActorType.Site;
actorInfo.ContentUri = "http://servername";

// Stop following the target site
followingManager.StopFollowing(actorInfo);

// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}

```

Output

Navigate to My Site and then Click on “Newsfeed”. You will see that the user has stopped following the site. The number of sites following has been reduced.

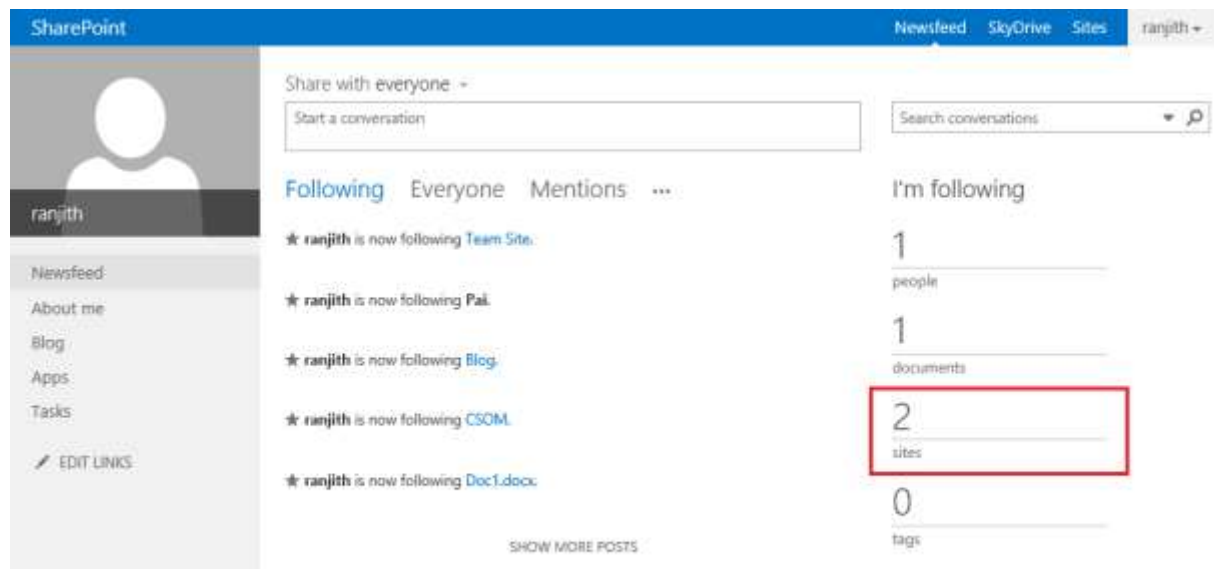


Figure16.13.1: Stop following the site

16.14 How to follow the document

In this example you will see how to follow the document using the .Net Client Side Object Model.

Build the project using the following:

- a) [Create a console application.](#)
- b) Replace Program.cs with the below source code below.
- c) Hit F5.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials
            // By default It will take current user who runs this application - Here it
            // will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
            // documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Identifies a user, document, site, or tag in social feed and following
            // activities
            SocialActorInfo actorInfo = new SocialActorInfo();
            actorInfo.ActorType = SocialActorType.Document;
            actorInfo.ContentUri = "http://servername/Shared%20Documents/Book.xlsx";

            // Follow the document
            followingManager.Follow(actorInfo);
        }
    }
}
```

```
// Execute the query to the server
clientContext.ExecuteQuery();
    }
}
}
```

Output

Navigate to My Site and then Click on “Newsfeed”. You will see that the user has started following the document.

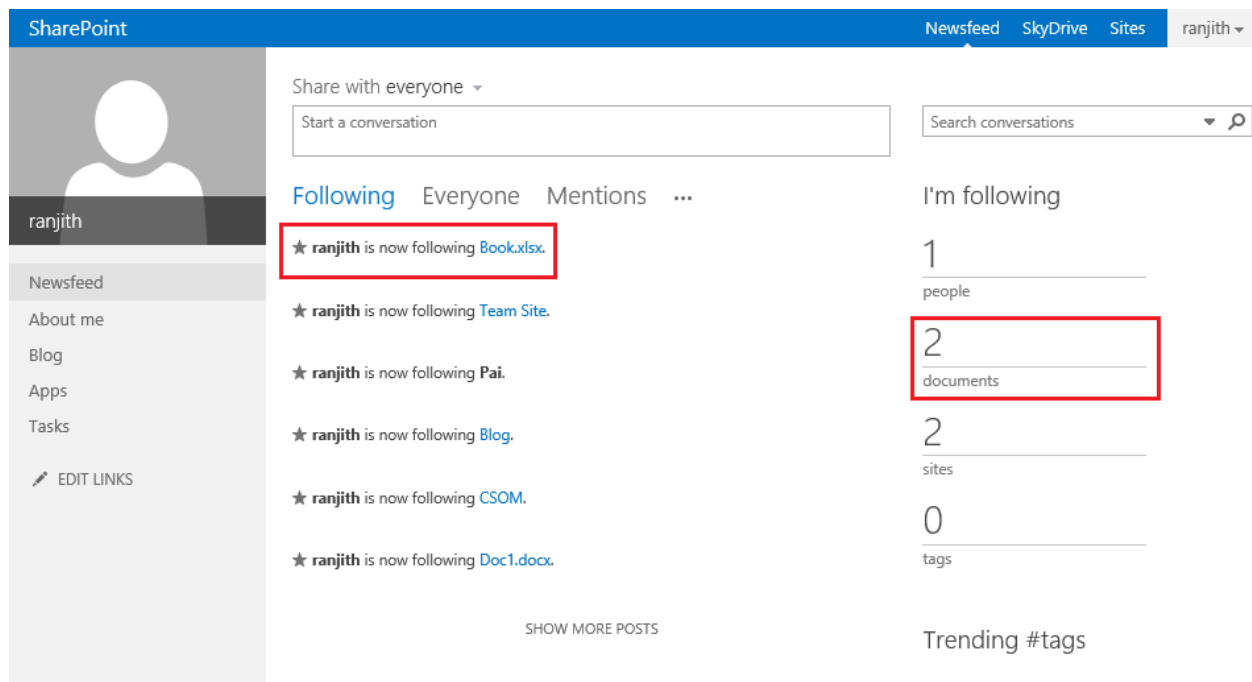


Figure16.14.1: Follow the document

16.15 How to stop following the document

In this example you will see how to stop following the document using the .Net Client Side Object Model.

Build the project using the following:

- Create a console application.
- Replace Program.cs with the below source code below.
- Hit F5.

Source Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Client.UserProfiles;
using Microsoft.SharePoint.Client.Social;

namespace CSOMCodeSamples
{
    class Program
    {
        static void Main(string[] args)
        {
            // ClientContext - Get the context for the SharePoint Site
            // SharePoint site URL - http://servername/sites/CSOM
            ClientContext clientContext = new
ClientContext("http://servername/sites/CSOM/");

            // Pass the credentials
            // By default It will take current user who runs this application - Here it
will take Administrator because the application is running with that account
            clientContext.Credentials = new NetworkCredential("ranjith", "password@1",
"dc07");

            // Get the SocialFollowingManager instance
            // Provides methods for managing a user's list of followed actors (users,
documents, sites, and tags)
            SocialFollowingManager followingManager = new
SocialFollowingManager(clientContext);

            // Identifies a user, document, site, or tag in social feed and following
activities
            SocialActorInfo actorInfo = new SocialActorInfo();
            actorInfo.ActorType = SocialActorType.Document;
            actorInfo.ContentUri = "http://servername/Shared%20Documents/Book.xlsx";

            // Stop following the document
            followingManager.StopFollowing(actorInfo);

            // Execute the query to the server
            clientContext.ExecuteQuery();
        }
    }
}

```

Output

Navigate to My Site and then Click on “Newsfeed”. You will see that the user has stopped following the document. The number of following documents has been reduced.

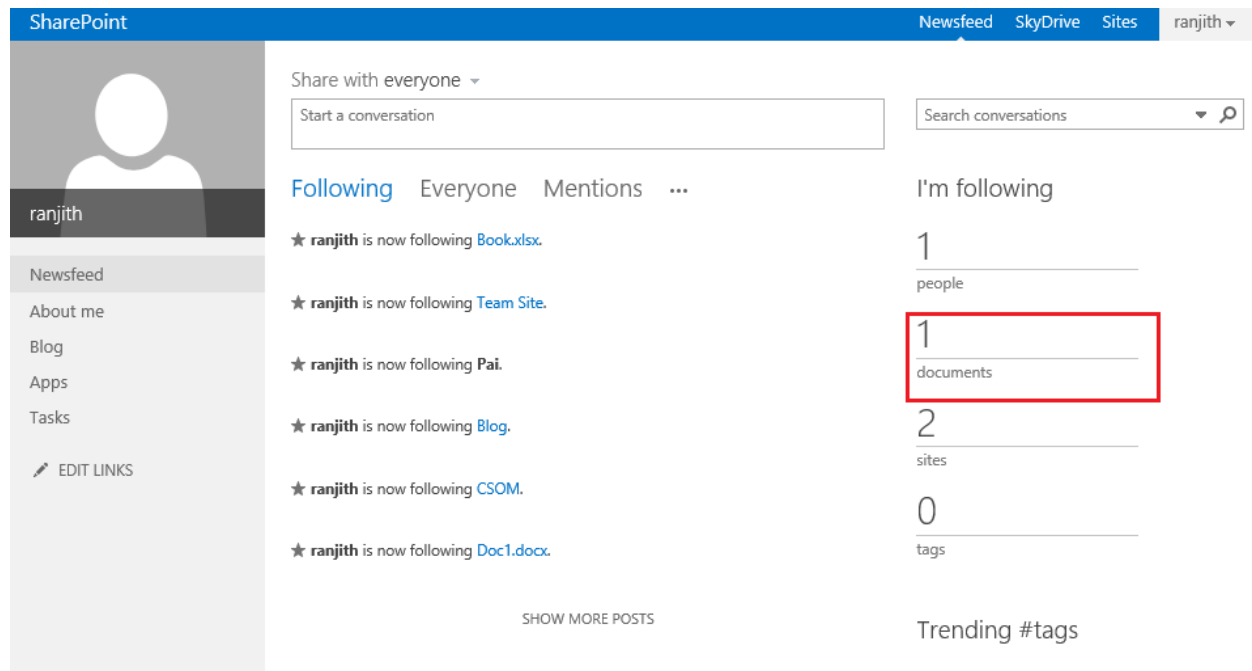


Figure16.15.1: Stop following the document

Summary:

In this book we have covered nearly all the basic operations that can be performed using the SharePoint 2013 .Net Client Side Object Model. This book is only the beginning; there are many things that can be done and you are now ready to develop advanced solutions using the .Net Client Side Object Model. This book is mainly useful for the app models introduced in SharePoint 2013 to be developed using the Client Side Object Model

Thanks for Reading!!!!!!