

PnP Core CSOM Programming

**Basic Operations on SharePoint Using
PnP Core CSOM Library**

Nakkeeran Natarajan

About the Author

Nakkeeran Natarajan have been working in IT industry over 5 years and have more than 4 years of experience in SharePoint technology.

Nakkeeran holds master's degree in Computer Science and Engineering from VIT University. He has started his career as intern from Honeywell Technology Solutions, Bengaluru. He is working as a Senior Software Consultant in Cognizant Technology Solutions, Bengaluru. He is involved in architecting, designing, developing solutions and SharePoint training.

Nakkeeran currently hold **Most Valued Professional** award from C# Corner forum (Mindcracker Network). He has shared over 100 articles of SharePoint on C# Corner forum. He has also authored a book on SharePoint PnP PowerShell Scripts programming. He has produced reusable components and white papers on SharePoint technology.

Nakkeeran holds certifications like Microsoft certified Professional (MCP), Microsoft Specialist (MS) and Microsoft Certified Technology Specialist (MCTS). He has also completed the following Microsoft certifications.

- Developing Microsoft SharePoint Server 2013 Core Solutions
- Programming in HTML5 with JavaScript and CSS3
- TS: Microsoft SharePoint 2010, Application Development



Nakkeeran Nataraja

(Author, Blogger & C# Corner MVP
Senior SharePoint Developer)

Acknowledgment

I would like to thank Mahesh Chand, Praveen Kumar and Dinesh Beniwal from C# Corner core team for their continuous support on C# Corner forum. I would also like to thank the technical team who helps me in reviewing the content on the forum.

I am thankful to my colleagues and mentors for their support. My special thanks for Vijai Anand and Ramakrishnan who always support and guide me in sharing my knowledge through books, blogs and articles.

I am really grateful for my friends and family members who motivate me in sharing the knowledge.

Table of Contents

1	SharePoint PnP Core CSOM Overview	1
2	Prerequisites	2
3	Tool Used	2
4	Connect to SharePoint site	3
5	SharePoint Site Operations	8
5.1	How to create a sub site	8
5.2	How to retrieve sub site URL's	10
5.3	How to retrieve sub site	11
5.4	How to check if Sub Site Exists	12
5.5	How to check if current web is sub site	13
5.6	How to delete a sub site	14
5.7	How to enable a feature (site scoped) on a site collection	15
5.8	How to disable a feature (site scoped) on a site collection	17
5.9	How to check if site collection feature is active (site scoped)	18
5.10	How to enable a feature (web scoped) on a site or a sub site	19
5.11	How to disable a feature (web scoped) on a site or a sub site	21
5.12	How to check if site feature is active (web scoped)	22
6	SharePoint List Operations	24
6.1	How to create a list	24
6.2	How to check if list exists using list name	26
6.3	How to check if list exists using list ID	27
6.4	How to retrieve a list using list name	28
6.5	How to retrieve a list using list URL	30
6.6	How to retrieve a list ID	31
6.7	How to update permissions on the list	32
6.8	How to enable versioning for the list	34
7	SharePoint List View Operations	36
7.1	How to create a list view	36
7.2	How to retrieve a list view using view name	38
7.3	How to retrieve a list view using view Id	39
8	SharePoint Content Type Operations	41
8.1	How to retrieve site content type by content type name	41
8.2	How to retrieve site content type by content type Id	43

8.3	How to create a site content type	45
8.4	How to create a site content type using content type XML	47
8.5	How to check if a site content type already exists by content type name	49
8.6	How to check if a site content type already exists by content type Id	51
8.7	How to add site content type to List.....	52
8.8	How to add site content type to List By content type Id	54
8.9	How to add site content type to List By content type name	56
8.10	How to check if a content type already exists on a List by content type name	57
8.11	How to check if a content type already exists on a List by content type Id	59
8.12	How to delete a content type by Name.....	60
8.13	How to delete a content type by Id	61
9	SharePoint Field Operations	63
9.1	How to create a field on SharePoint site	63
9.2	How to create a field on SharePoint site using Field XML	66
9.3	How to retrieve a field using field Id.....	68
9.4	How to check if a field exists using field Id	70
9.5	How to check if a field exists using field Name.....	71
9.6	How to add a field to site content type	73
9.7	How to add a field to site content type using Content type Id.....	75
9.8	How to add a field to site content type using Content type name.....	77
9.9	How to check if a field exists in the Content type	79
10	SharePoint Folder tasks	81
10.1	How to create a folder on a library	81
10.2	How to check if a folder exists on a library using folder name.....	83
10.3	How to check if a folder exists on a library using folder path	84
10.4	How to retrieve folder from library	86
10.5	How to check and create a folder on a library using folder name.....	87
10.6	How to check and create a folder on a library using folder URL	89
11	SharePoint Basic File tasks	91
11.1	How to upload a file.....	91
11.2	How to retrieve a file	93
11.3	How to download a file.....	95
11.4	How to check-out a file	96
11.5	How to check-in a file.....	98

11.6	How to update file properties.....	99
12	SharePoint Page Tasks	102
12.1	How to create a publishing page	102
12.2	How to retrieve a publishing page.....	104
12.3	How to create a wiki page using page name	105
12.4	How to create a wiki page using URL and file content	106
12.5	How to retrieve wiki page content	108
12.6	How to ensure a wiki page.....	109
12.7	How to update a wiki page content.....	111
12.8	How to add layout to a wiki page	112
12.9	How to retrieve web parts from a page.....	114
12.10	How to add web part to a publishing page.....	115
12.11	How to delete web part from a page.....	117
12.12	How to retrieve web part properties.....	118
12.13	How to update web part properties	120
12.14	How to add web part to a wiki page	121
13	SharePoint User and Group Operations	123
13.1	How to retrieve site collection administrator.....	123
13.2	How to add site collection administrator	125
13.3	How to create a SharePoint group.....	126
13.4	How to retrieve a SharePoint group ID.....	128
13.5	How to check if a SharePoint group exists on Site.....	129
13.6	How to add User to a SharePoint group	130
13.7	Add permissions to SharePoint group / Add Groups with permissions to Site (or list or item) 132	
13.8	Removes permissions from SharePoint group / Removes group with permissions from site (or list or item)	134
13.9	Add permissions to SharePoint User / Add User with permissions to Site (or list or item)	135
13.10	Remove permissions for SharePoint User / Remove User with permissions from Site (or list or item).....	137
13.11	Grant read permissions to everyone except external users	139
14	Taxonomy Tasks	141
14.1	How to create Term Group	141
14.2	How to ensure Term Group	143
14.3	How to retrieve Term Group by Name	144

14.4	How to retrieve Term Group by ID	146
14.5	How to create/ensure Term Set	147
14.6	How to retrieve Term Sets	148
14.7	How to create a Term on Term Set.....	150
14.8	How to retrieve Term from Term Set	151
14.9	How to retrieve Term from Term Store by Path.....	153
14.10	How to import Terms.....	154
14.11	How to export Term Set.....	156
14.12	How to export all Terms.....	157
14.13	How to create Taxonomy Field on Site/List	159
14.14	How to update Taxonomy Field on List Item	161
14.15	How to remove Taxonomy Field from Site	162
15	Office 365 Site Collection tasks.....	164
15.1	How to create a site collection on tenant site	164
15.2	How to retrieve site collections from tenant site	166
15.3	How to check if a site collection exists on tenant site	167
15.4	How to delete a site collection	169
16	Summary	171
17	References	171

1 SharePoint PnP Core CSOM Overview

What is PnP? Patterns and Practices where Microsoft and external members contributes to the implementation practices for SharePoint. PnP libraries are created to serve the needs of developers for easy learning and implementation of logic through client side programming. There are various PnP programming methodologies. In this book, I have explained about implementing PnP programming with .Net managed client side code.

Why we need PnP core CSOM? PnP core library internally has implemented Client Side Object model for its operations. This in turn makes the operations adaptable. The same set of operations can be executed on any SharePoint environment. It simplified the remote development like provider hosted app or windows code. PnP Core CSOM helps users to get the required information in less piece of code, where in with traditional client side or server side object model multiple lines of code is required to access objects. The code complexity reduced through this implementation.

The SharePoint on premise or SharePoint online sites can be managed remotely using PnP core component. The operations are actually supported by PnP packages/libraries. The PnP core component libraries are open source extensions on top of classic SharePoint CSOM and REST APIs. The PnP core CSOM libraries can be imported to user's machine from the official site whenever required.

The book is designed for SharePoint developers who can easily learn and implement the functionalities.

The PnP Core CSOM libraries are available for different versions of SharePoint like SharePoint 2013 & 2016 on-premises and Office 365 environments.

The documentation of PnP Core CSOM is available on the github site <https://github.com/OfficeDev/PnP-Sites-Core/blob/master/Core/Documentation.md>. The package installation guide is available on Nuget site.

SharePoint online is part of Office 365 which has/provides powerful features without managing the infrastructure, while SharePoint on-premises gives us the luxury of accessing the infrastructure and SharePoint features. There are limited ways of accessing SharePoint online data compared to SharePoint on-premises. The SharePoint online approach should have client side methods to access the content. So we can use PnP Core CSOM to access the SharePoint online objects independent of infrastructure, since Client Side packages are internally used on PnP libraries. The same can be adapted/compatible for on-premises environments.

Some of the operations that can be performed on SharePoint are

- Managing site collections and sub sites
- Managing content types, columns and lists
- Managing users and groups
- Working with folders and files
- Managing pages
- Working with term stores and taxonomy.

The operations related to each of the above items are explained in detail in the following sections.

2 Prerequisites

Here, you will see the prerequisites required for executing any of the PnP Core operations.

The following prerequisite needs to be installed on the machine.

- Microsoft Visual Studio 2013 or above
- NuGet Package Manager for Visual Studio

The latest version of the packages can be downloaded from Nuget site and installed on NuGet Package Manager console available on the Visual Studio

(<https://www.nuget.org/packages/SharePointPnPCoreOnline/>).

In this book, you will see the operations executed on Visual Studio.

The examples explained are compatible for any environment like SharePoint 2013, SharePoint 2016 and Office 365. I have executed the samples using SharePoint online site.

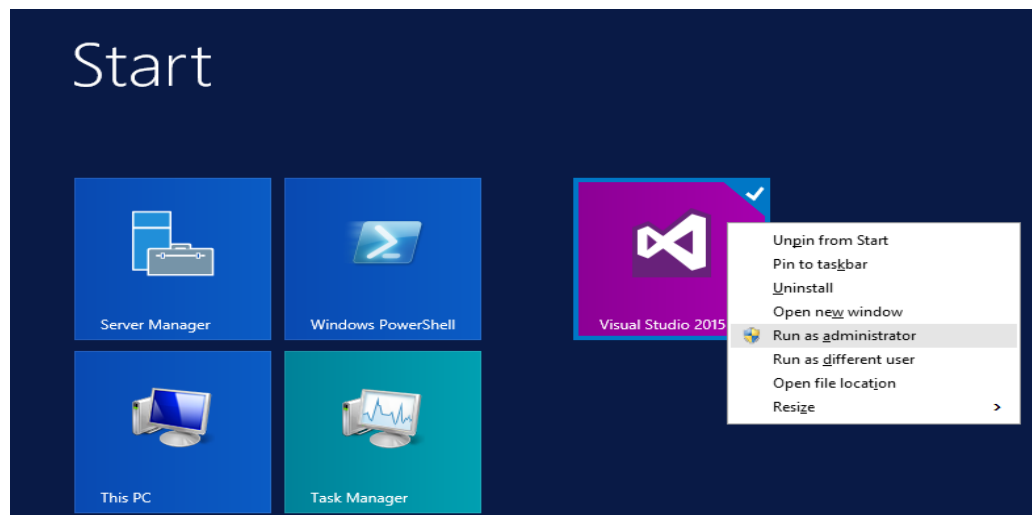
3 Tool Used

In this book, I have explained about creating/running the code using Microsoft Visual Studio. The steps explained in the below sections are tested/taken from development environment. If its production, you have to package the solution developed using below samples and should be deployed on to the environments using scripts.

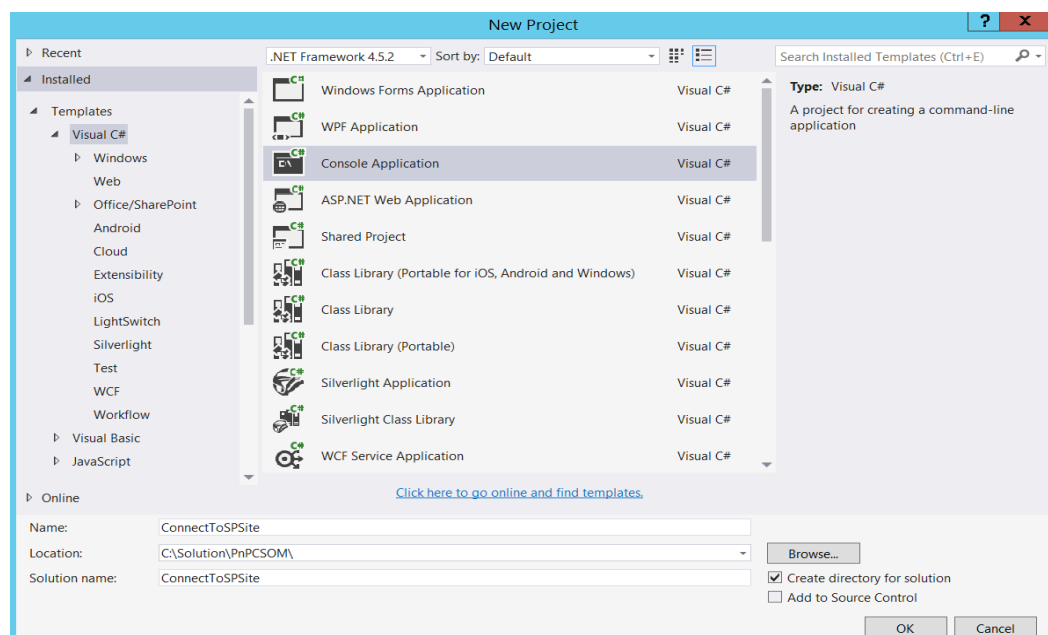
4 Connect to SharePoint site

The below operations are executed using Visual Studio. The following steps are used to create and run the code.

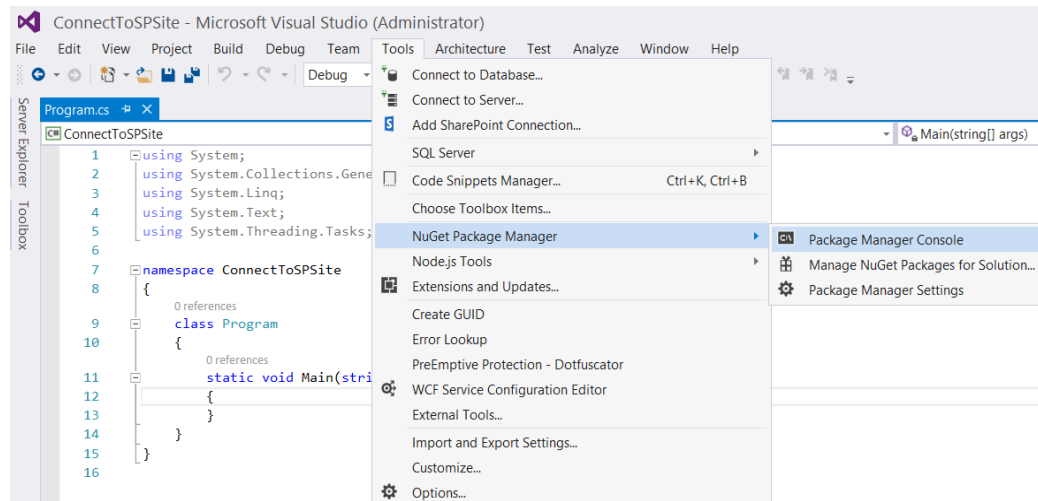
1. Go to Start Menu and search for Visual Studio. Right click on Visual Studio and run as administrator.



2. Click on File -> New -> Project. Under the templates, navigate to Visual C# and then select Console Application on the templates pane. Provide the project name and click on Ok.

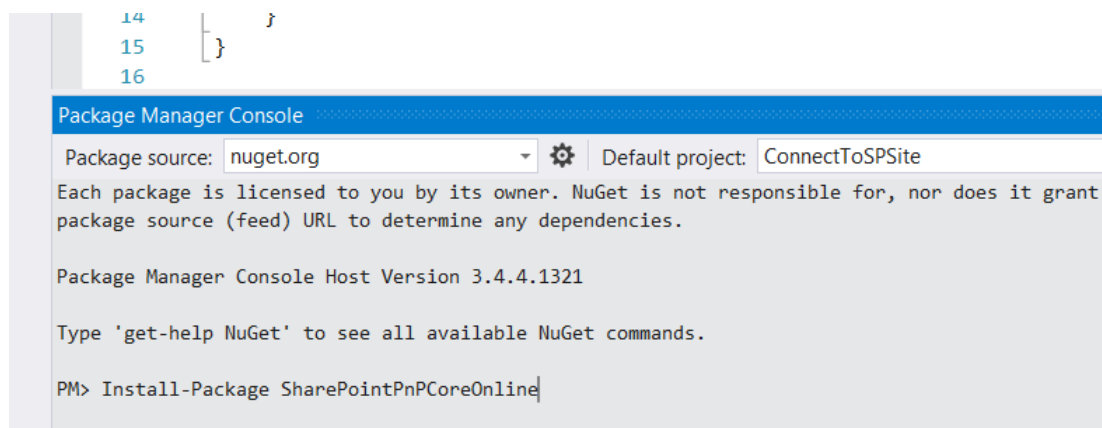


3. From menu, navigate to Tools -> NuGet Package Manager -> Package Manager Console.

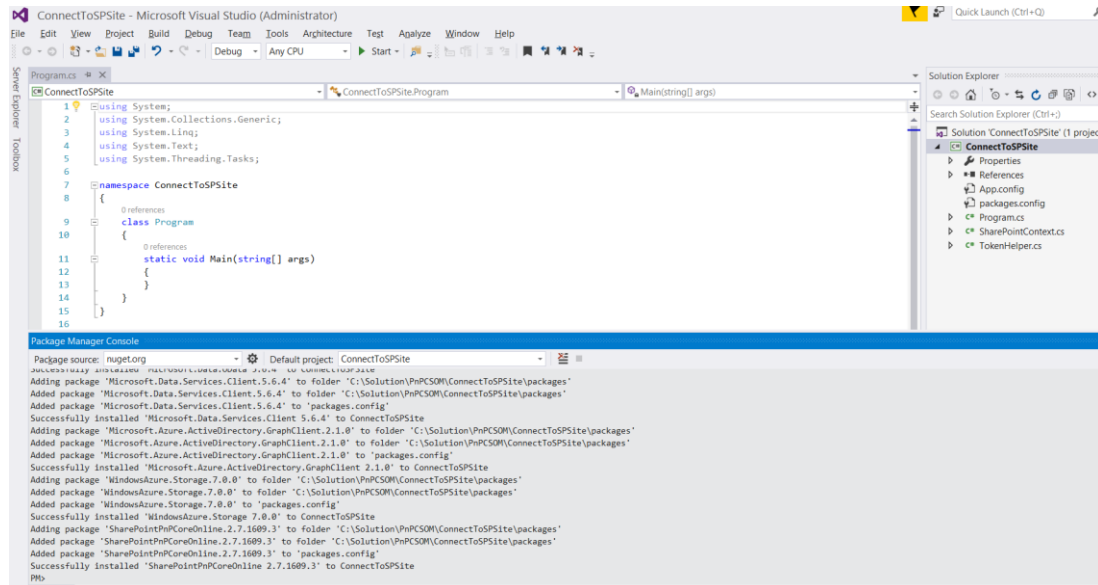


4. The NuGet Package Manager console will open in Visual Studio. Depending on the SharePoint target version, import the package.
 - a. For SharePoint 2013 sites, import the SharePoint 2013 PnP Core CSOM package on the console using the command “Install-Package SharePointPnPCore2013”.
 - b. For SharePoint 2016 sites, import the SharePoint 2016 PnP Core CSOM package on the console using the command “Install-Package SharePointPnPCore2016”.
 - c. For SharePoint online sites, import the SharePoint online PnP Core CSOM package on the console, using the command “Install-Package SharePointPnPCoreOnline”.

For example, I have imported the SharePoint online package as shown below.



Once the command is executed, Visual Studio will take maximum a minute to complete the installation. The command will run and import the libraries from the NuGetsite



5. Paste the required code inside the Main method of program file and save the project. The following steps helps in authentication.

- a. For authentication, the required inputs are site URL, User Id and password. In case if your target version is SharePoint on premise, domain is required as additional input.
 - b. Create the authentication manager object.
 - c. Get the client context of the site by using authentication manager object and invoke appropriate method with the above inputs.
- If your target version is SharePoint online, then method will be GetSharePointOnlineAuthenticatedContextTenant.

```
var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName,
password)
```

If your target version is SharePoint 2013/2016 on premise, then method will be GetNetworkCredentialAuthenticatedContext.

```
var clientContext =
authManager.GetNetworkCredentialAuthenticatedContext(siteUrl, userName,
password, domain)
```

- d. Subsequently, using the client context obtained, access the necessary objects/methods.
- e. The code snippet given below shows the authentication, also shows the code logic to check whether the list is exist or not. This code shows for SharePoint online authentication. For SharePoint On-Premises, put the required authentication, as explained in step c.

Code for Authentication

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

namespace ConnectToSPSite
{
    class Program
    {
        static void Main(string[] args)
        {
            // Input Parameters
            string siteUrl = "https://nakkeerann.sharepoint.com/";
            string userName = "abc@nakkeerann.onmicrosoft.com";
            string password = "***";

            // PnP component to set context
            AuthenticationManager authManager = new AuthenticationManager();

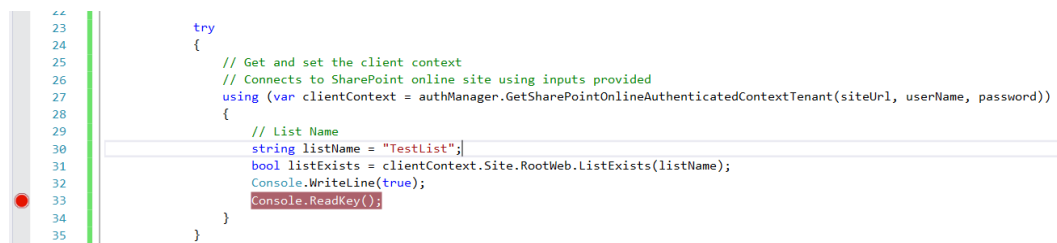
            try
            {
                // Get and set the client context
                // Connects to SharePoint online site using inputs provided
                using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
                {
                    // List Name
                    string listName = "TestList";
                    bool listExists = clientContext.Site.RootWeb.ListExists(listName);
                    Console.WriteLine(listExist);
                    Console.ReadKey();
                }
            }
        }
    }
}
```

```

    }
    catch (Exception ex)
    {
        Console.WriteLine("Error Message: " + ex.Message);
        Console.ReadKey();
    }
}
}
}

```

6. If required, keep the breakpoints for debugging the code. For example, go to line number 32, then go to Debug option → Toggle Break Point. Or just press F9 from the required line.



```

23
24
25
26
27
28
29
30
31
32
33
34
35

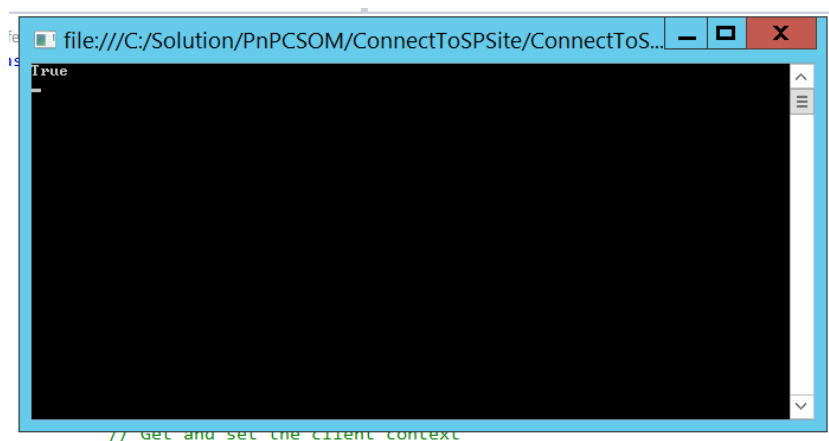
```

```

try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // List Name
        string listName = "TestList";
        bool listExists = clientContext.Site.RootWeb.ListExists(listName);
        Console.WriteLine(true);
        Console.ReadKey();
    }
}

```

7. For testing/debugging the code hit F5 or from the top menu, navigate and click on Debug -> Start Debugging option.
8. You can see necessary information/response on the Debugger console.



```

file:///C:/Solution/PnPCSOM/ConnectToSPSite/ConnectToS...
True

```

Note

In all my examples below, I will be using SharePoint online sites. So SharePoint online authentication method is used. If your target platform is SharePoint 2013/2016 On-Premise, then please use the appropriate authentication mechanism as explained in the above steps.

5 SharePoint Site Operations

In this section, you will learn how to perform create, retrieve, check and delete sites using PnP Core CSOM library. You will also learn the feature operations for the SharePoint sites.

The methods given below are used for the operations/examples explained in this section.

- CreateWeb
- GetAllWebUrls
- GetWeb
- WebExists
- IsSubSite
- DeleteWeb
- ActivateFeature
- DeactivateFeature
- IsFeatureActive

5.1 How to create a sub site

In this example, you will learn how to create a sub site from SharePoint site, using PnP Core CSOM. The subsite can be created using CreateWeb method from the root Web object.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- In the program file, paste the code given below and save the file.
- The input parameters for the operation are given below.
 - Sub Site Title
 - Sub Site URL
 - Description
 - Template
 - Language
 - Inherit Permissions (Boolean)
 - Inherit Navigation (Boolean)
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // Gets Sub Site URLs
        string subSiteTitle = "SubSite3";
        string subSiteUrl = "SubSite3";
        string subSiteDesc = "SubSite Created using PnP CSOM";
        string subSiteTemplate = "STS#0";
        int language = 1033;
        bool inheritPermissions = true;
        bool inheritNavigation = false;

        // Creates Subsite
        Web newWebSite = clientContext.Site.RootWeb.CreateWeb(subSiteTitle, subSiteUrl, subSiteDesc,
subSiteTemplate, language, inheritPermissions, inheritNavigation);
        Console.WriteLine(newWebSite.Title + " subsite created");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The sub site will be created. You can check the site by pasting the URL (<https://nakkeerann.sharepoint.com/sites/learning/Subsite3>) on the Browser.

5.2 How to retrieve sub site URL's

In this example, you will learn how to retrieve site collection information using PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- In the program file, paste the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

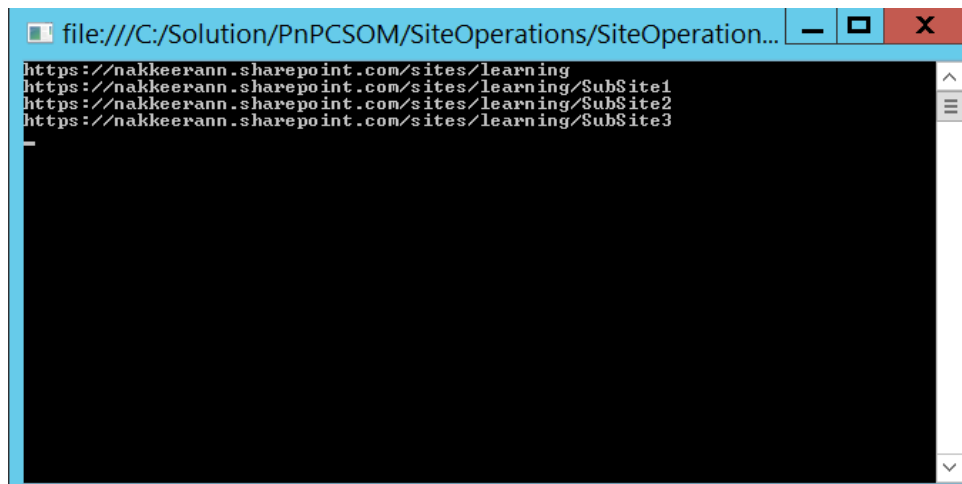
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Gets all Subsites
        var webs = clientContext.Site.GetAllWebUrls();

        foreach (var web in webs) {
            Console.WriteLine(web);
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on console, as shown below.



5.3 How to retrieve sub site

In this example, you will learn how to retrieve sub site from SharePoint site, using PnP Core CSOM. The sub site can be retrieved, using GetWeb method with the help of the root web object.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameter will be Sub Site leaf URL.
- In the program file, paste the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

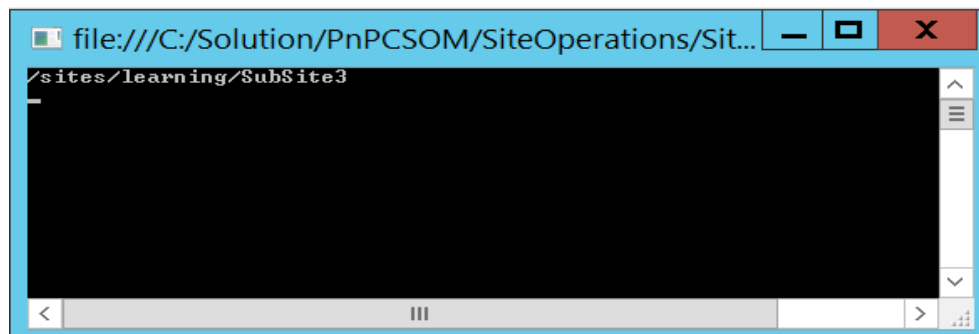
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Gets Subsite by subsite url
        Web web = clientContext.Site.RootWeb.GetWeb("SubSite3");
        // Displays server relative Url.
        Console.WriteLine(web.ServerRelativeUrl);
        Console.ReadKey();
    }
}
catch (Exception ex)
```

```
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on console, as shown below.



All the Website properties cannot be accessed directly, using PnP Core CSOM. To access all the properties, use traditional CSOM managed code.

5.4 How to check if Sub Site Exists

In this example, you will learn, if the sub site exists on the SharePoint site, using PnP Core CSOM. The sub site existence can be checked, using WebExists object.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The necessary input parameter is Sub Site leaf URL.
- In the program file, paste the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Checks if subsite exists Subsite by subsite url
        bool siteExists = clientContext.Site.RootWeb.WebExists("SubSite3");
        if (siteExists)
        {
            Console.WriteLine("Subsite exists");
        }
        else
        {
            Console.WriteLine("Subsite doesnt exists");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.

5.5 How to check if current Web is sub site

In this example, you will learn how to check a Website is a sub site or not, using PnP Core CSOM library. `IsSubSite` method is used to check, if the current site is sub site.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- In the program file, paste the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        string subsiteName = "SubSite3";
        // Checks if the web accessed is subsite or not
        bool isSubSite = clientContext.Site.OpenWeb(subsiteName).IsSubSite();
        if (isSubSite)
        {
            Console.WriteLine("SubSite3 is Subsite");
        }
        else
        {
            Console.WriteLine("SubSite3 is not a sub site");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.

5.6 How to delete a sub site

In this example, you will learn how to delete a site (Web scoped), using PnP Core CSOM library. The sub site can be deleted, using DeleteWeb method with the help of the root Web object.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter is a sub site name.

- In the program file, insert the code mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        string subsiteName = "SubSite3";
        // Delete subsite
        bool isSubSite = clientContext.Site.RootWeb.DeleteWeb(subsiteName);
        if (isSubSite)
        {
            Console.WriteLine("Site deleted");
        }
        else
        {
            Console.WriteLine("Site not deleted");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console.

5.7 How to enable a feature (site scoped) on a site collection

In this example, you will learn how to activate or enable the site collection feature, which is not active, using PnP Core CSOM library. In this example, we will see how to activate the OOB feature, which is not active on the site. You will learn how to activate Document ID Service feature. The ID of the document ID Service feature is b50e3104-6812-424f-a011-cc90e6327318.

Steps involved

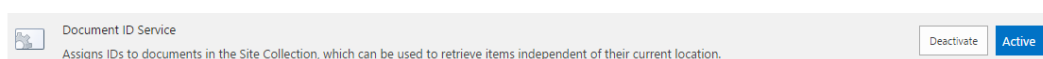
- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book for importing the packages and creating authentication manager object.
- The required input parameter for the operation is feature ID.
- In the program file, insert the code mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Document ID Service Feature Id
        Guid featureId = new Guid("b50e3104-6812-424f-a011-cc90e6327318");
        // Activates site collection feature
        clientContext.Site.ActivateFeature(featureId);
        Console.WriteLine("Feature is activated");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The Document ID Service feature is activated. The feature can be identified on the site collection features page. The site collection features page URL of the example site is https://nakkeerann.sharepoint.com/sites/learning_layouts/15/ManageFeatures.aspx?Scope=Site. The image given below shows the document ID Service feature, which is activated.



The OOB features list along with feature ID of SharePoint 2013 is available on [MSDN blog site](#). The required site scoped feature Id can be copied from the list.

5.8 How to disable a feature (site scoped) on a site collection

In this example, you will learn how to deactivate or disable the feature (site scoped), which is active on the site collection, using PnP Core CSOM library.

In this example, we will see how to deactivate the OOB feature, which is active on the site. The OOB features list along with the feature ID of SharePoint 2013 is available on [MSDN blog site](#). The required site scoped feature Id can be copied from the list. You will learn how to deactivate the site policy feature. The ID of the site policy feature is 2fcd5f8a-26b7-4a6a-9755-918566dba90a.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- In the program file, insert the code given below and save the file.
- The required input parameter for the operation is the feature ID.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Site Policy Feature Id
        Guid featureId = new Guid("2fcd5f8a-26b7-4a6a-9755-918566dba90a");
        // Deactivates site collection feature
        clientContext.Site.DeactivateFeature(featureId);
        Console.WriteLine("Feature is deactivated");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```


Result

The result will be displayed on the console. The site policy feature is deactivated. The feature can be identified on the site collection features page. The site collection features page URL of the example site is

https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/ManageFeatures.aspx?Scope=Site. The below image shows the document ID service feature which is activated.



5.9 How to check if the site collection feature is active (site scoped)

In this example, you will learn how to check, if the site collection feature (site scoped) is activated or not, using PnP Core CSOM library.

In this example, we will see how to check the OOB feature status on the site. The OOB features list along with the feature ID of SharePoint 2013 is available on [MSDN blog site](#). The required site scoped feature Id can be copied from the list. You will learn how to check the site policy feature status. The ID of the site policy feature is 2fcd5f8a-26b7-4a6a-9755-918566dba90a.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the operation is the feature ID.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Site Policy Feature Id
        Guid featureId = new Guid("2fcd5f8a-26b7-4a6a-9755-918566dba90a");
        // Checks if site collection feature is active
        bool isActive = clientContext.Site.IsFeatureActive(featureId);
    }
}
```

```
        if (isActive)
        {
            Console.WriteLine("Site Collection Feature is Active");
        }
        else
        {
            Console.WriteLine("Site Collection Feature is Not Active");
        }

        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The site policy feature status result will be displayed on the console. The feature can be identified on the site collection features page manually. The site collection features page URL of the example site is

https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/ManageFeatures.aspx?Scope=Site.

5.10 How to enable a feature (web scoped) on a site or a sub site

In this example, you will learn how to activate or enable the Web scoped feature which is not active, using PnP Core CSOM library.

In this example, we will see how to activate OOB Web feature, which is not active on the site. The OOB features list along with the feature ID of SharePoint 2013 is available on **MSDN blog site**. The required Web scoped feature Id can be copied from the list. You will learn how to activate SharePoint Server Publishing feature on the sub site. The ID of the SharePoint Server Publishing feature is 94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in section 4 (Connect to SharePoint site) of the book to import the packages and create authentication manager object.
- The required input parameter for the operation is the feature ID. The operation is triggered, using necessary web object.

- In the program file, insert the code mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

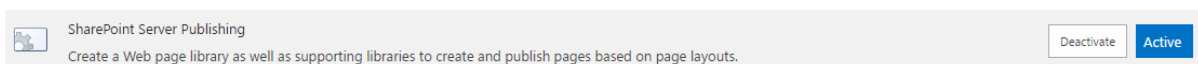
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // SharePoint Server Publishing Feature Id
        Guid featureId = new Guid("94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb");
        // Activates web collection feature
        clientContext.Site.OpenWeb("SubSite2").ActivateFeature(featureId);

        // uncomment to active feature at root level site
        // clientContext.Site.RootWeb.ActivateFeature(featureId);
        Console.WriteLine("Web Feature is activated");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

SharePoint Server publishing feature is activated. The feature can be identified on the site features page. The site features page URL of the example site is https://nakkeerann.sharepoint.com/sites/learning/SubSite2/_layouts/15/ManageFeatures.aspx. The snapshot given below depicts the feature.



5.11 How to disable a feature (web scoped) on a site or a sub site

In this example, you will learn how to deactivate or disable the Web scoped site feature , which is active, using PnP Core CSOM library.

In this example, we will see how to deactivate OOB Web feature, which is active on the site. OOB features list along with the feature ID of SharePoint 2013 is available on [MSDN blog site](#). The required Web scoped feature Id can be copied from the list. You will learn how to deactivate SharePoint Server Publishing feature on the sub site. The ID of SharePoint Server Publishing feature is 94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the operation is the feature ID. The operation is triggered, using necessary Web object.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

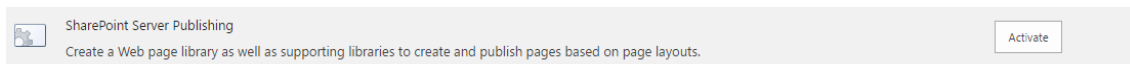
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // SharePoint Server Publishing Feature Id
        Guid featureId = new Guid("94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb");
        // Deactivates site collection feature
        clientContext.Site.OpenWeb("SubSite2").DeactivateFeature(featureId);
        Console.WriteLine("Web Feature is deactivated");

        // uncomment To active feature at root level site
        // clientContext.Site.RootWeb.DeactivateFeature(featureId);

        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

SharePoint Server publishing feature is deactivated. The feature can be identified on the site features page. The site features page URL of the example site is https://nakkeerann.sharepoint.com/sites/learning/SubSite2/_layouts/15/ManageFeatures.aspx. The snapshot depicts the deactivated feature.



5.12 How to check if site feature is active (web scoped)

In this example, you will learn how to check if the Web feature is activated or not, using PnP Core CSOM library.

In this example, we will see how to check OOB Web feature, which is active on the site. The OOB features list along with the feature ID of SharePoint 2013 is available on [MSDN blog site](#). The required Web scoped feature Id can be copied from the list. You will learn how to check SharePoint Server Publishing feature is active on the sub site. The ID of SharePoint Server Publishing feature is 94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the operation is the feature ID. The operation is triggered, using necessary Web object.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // SharePoint Server Publishing Feature Id
        Guid featureId = new Guid("94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb");
```

```
// Checks if web feature is active
bool isActive = clientContext.Site.OpenWeb("SubSite2").IsFeatureActive(featureId);
if (isActive) {
    Console.WriteLine("Web Feature is Active");
}
else
{
    Console.WriteLine("Web Feature is Not Active");
}

Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The feature can be identified manually on the site features page. The site features page URL of the example site is https://nakkeerann.sharepoint.com/sites/learning/SubSite2/_layouts/15/ManageFeatures.aspx.

6 SharePoint List Operations

In this section, you will learn how to work with the lists, using PnP Core CSOM library. Some of the basic operations like create, retrieve, update and check list operations are explained.

The methods given below are used for the examples explained in this section.

- CreateList
- ListExists
- GetListByTitle
- GetListByUrl
- GetListId
- SetListPermission
- UpdateListVersioning

6.1 How to create a list

In this example, you will learn how to create a list on SharePoint site, using PnP Core CSOM library. The list is created, using CreateList from the Web object. SharePoint custom list will be created, using the steps given below.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters are given below.
 - List template
 - List name
 - Enable versioning (Boolean)
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

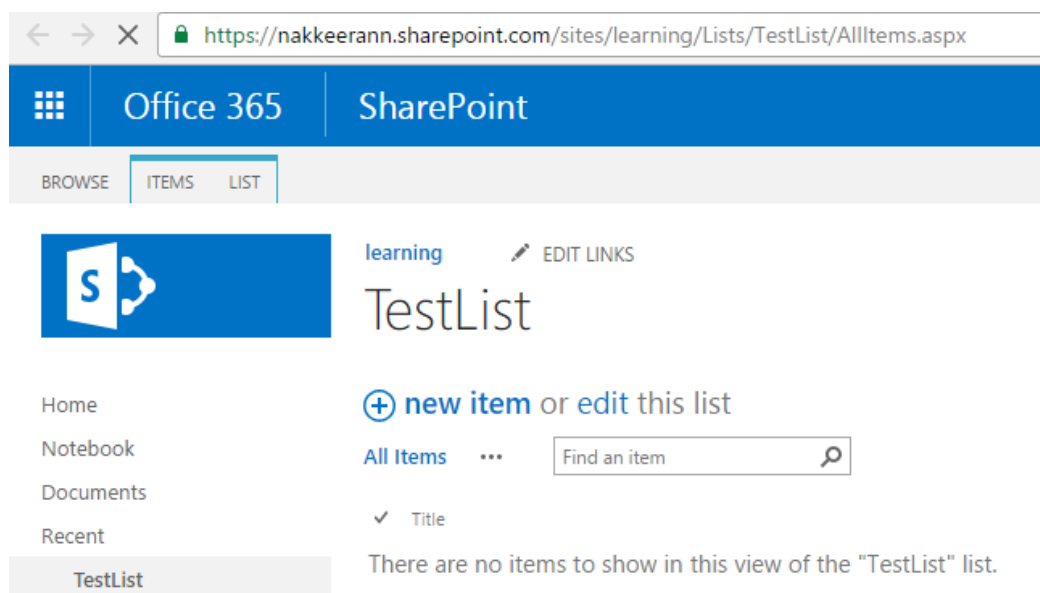
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
```

```
{
    // List Name
    string listName = "TestList";
    // Custom List Template
    ListTemplateType listTemplate = ListTemplateType.GenericList;
    // No versioning
    bool enableVersioning = false;
    // Creates List
    List list = clientContext.Site.RootWeb.CreateList(listTemplate, listName,
enableVersioning);

    Console.WriteLine("The list has been created.");
    Console.WriteLine("List Name : " + list.Title);
    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The snapshot given below shows the list created on the site.



6.2 How to check if the list exists using the list name

In this example, you will learn how to check, if a list exists or not, using PnP Core CSOM library by the list name. ListExists method is used for the operation.

Steps involved

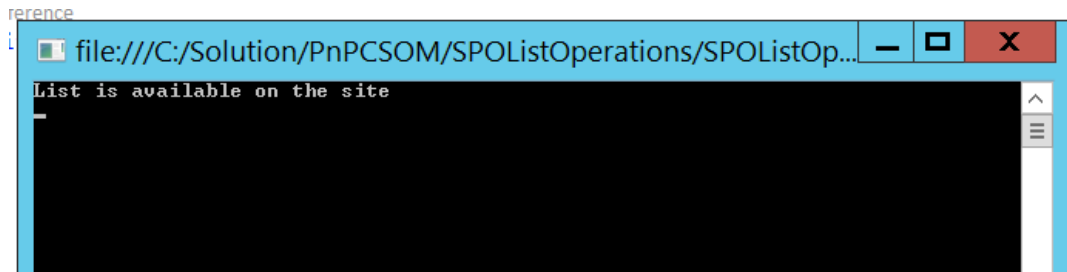
- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the operation is the list name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // List name input
        string listName = "TestList";
        // Checks the list exists
        bool listExists = clientContext.Site.RootWeb.ListExists(listName);
        if (listExists)
        {
            Console.WriteLine("List is available on the site");
        }
        else
        {
            Console.WriteLine("List is not available on the site");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



6.3 How to check if list exists using list ID

In this example, you will learn how to check, if a list exists or not, using PnP Core CSOM library by list Id. ListExists method is used for the operation.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter is the list ID.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // List Id input
        Guid listId = new Guid("F36FCF25-AE26-4CAA-933B-A9F52376ED4F");
        // Checks the list exists using list Id
        bool listExists = clientContext.Site.RootWeb.ListExists(listId);
        if (listExists)
        {
            Console.WriteLine("List is available on the site");
        }
        else
        {

```

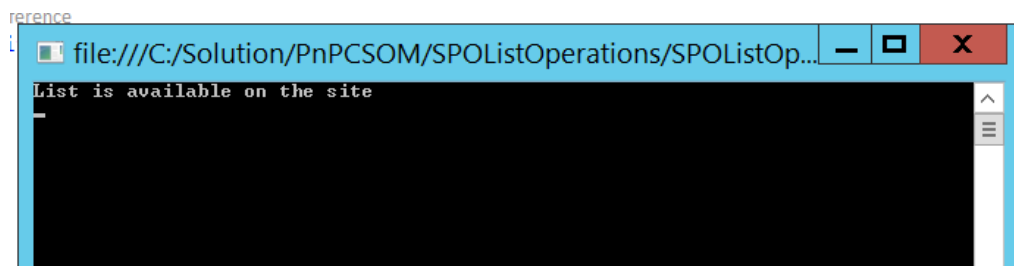
```

        Console.WriteLine("List is not available on the site");
    }
    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The results will be displayed on the console, as shown below.



6.4 How to retrieve a list using list name

In this example, you will learn how to retrieve a particular list from the current SharePoint site by the list name, using PnP core CSOM library. Here, we will see how we can retrieve the list object, using the list name. GetListByTitle method is used for the operation.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter is the list name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```

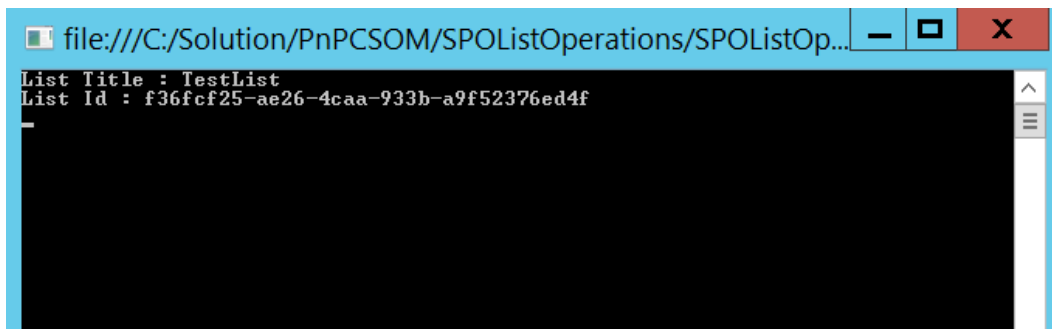
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided

```

```
using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
{
    // List Name input
    string listName = "TestList";
    // Retrieves list object using title
    List list = clientContext.Site.RootWeb.GetListByTitle(listName);
    if (list != null)
    {
        // Displays required result
        Console.WriteLine("List Title : " + list.Title);
        Console.WriteLine("List Id : " + list.Id);
    }
    else
    {
        Console.WriteLine("List is not available on the site");
    }
    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



```
file:///C:/Solution/PnPCSOM/SPOListOperations/SPOListOp...
List Title : TestList
List Id : f36fcf25-ae26-4caa-933b-a9f52376ed4f
```

6.5 How to retrieve a list using the list URL

In this example, you will learn how to retrieve a particular list from the current SharePoint site by the list URL, using PnP core CSOM library. `GetListByUrl` method is used for the operation.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the operation is the list URL.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

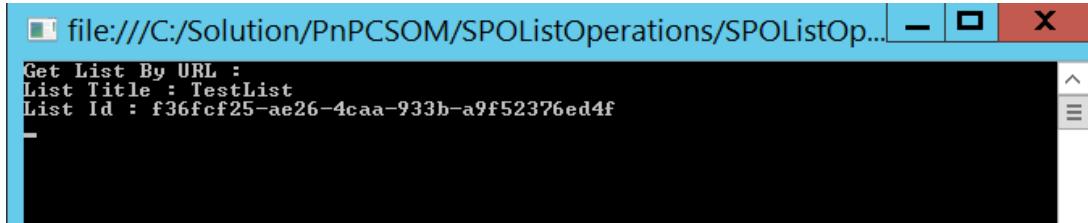
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        Console.WriteLine("Get List By URL :");

        // List Url Input
        string listUrl = "/Lists/TestList";

        // Gets list object using the list Url
        List list = clientContext.Site.RootWeb.GetListByUrl(listUrl);
        if (list != null)
        {
            // Displays the list details
            Console.WriteLine("List Title : " + list.Title);
            Console.WriteLine("List Id : " + list.Id);
        }
        else
        {
            Console.WriteLine("List is not available on the site");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



6.6 How to retrieve a list ID

In this example, you will learn how to retrieve a particular list ID from the current SharePoint site, using PnP core CSOM library. ID of the list can be retrieved, using `GetListID` method.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the method is the list name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        Console.WriteLine("Get List Id By List Name ::");

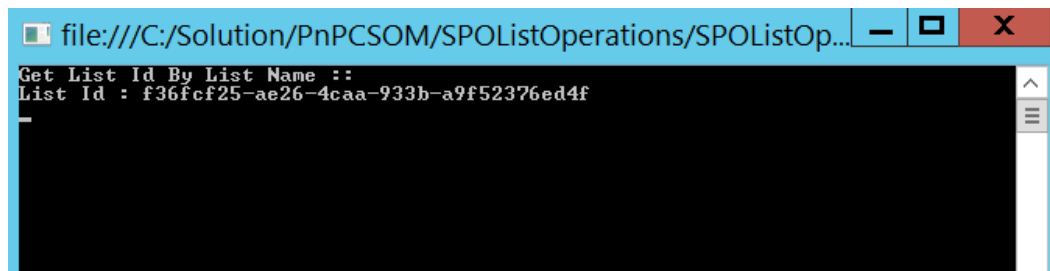
        // List Name
        string listName = "TestList";
        // Get List Id
        Guid listId = clientContext.Site.RootWeb.GetListID(listName);

        Console.WriteLine("List Id : " + listId);
        Console.ReadKey();
    }
}
```

```
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



```
file:///C:/Solution/PnPCSOM/SPOListOperations/SPOListOp...
Get List Id By List Name ::
List Id : f36fcf25-ae26-4caa-933b-a9f52376ed4f
```

6.7 How to update permissions on the list

In this example, you will learn how to set/update permissions for a SharePoint list, using PnP core CSOM library. The permissions of the list are updated, using SetListPermission method.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method are given below.
 - Identity
 - Rolt type
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // List name input
        string listName = "TestList";
```

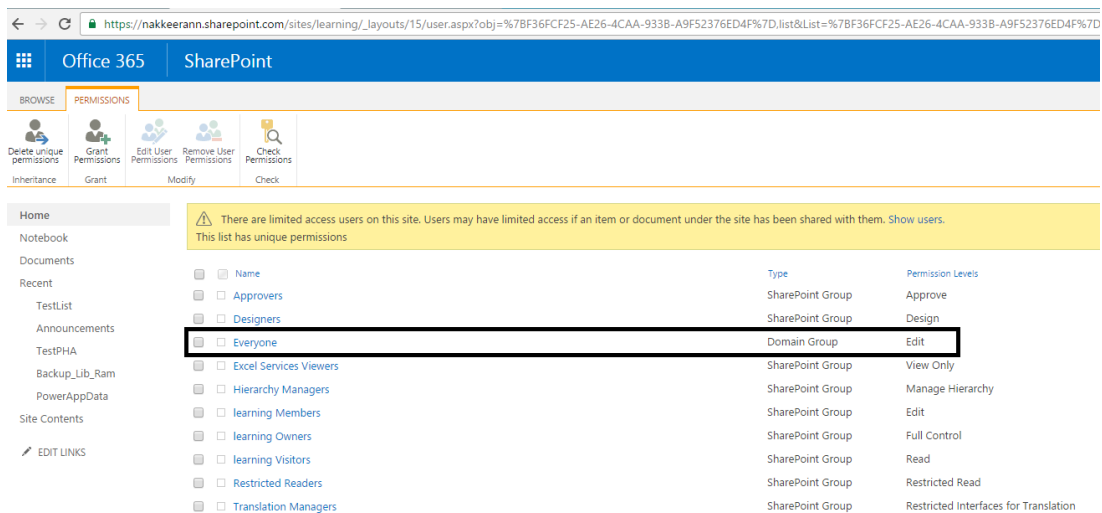
```
// Get List
List list = clientContext.Site.RootWeb.GetListByTitle(listName);
// Set Permissions
list.SetListPermission(OfficeDevPnP.Core.Enums.BuiltInIdentity.Everyone,
RoleType.Editor);

Console.WriteLine("Provided permission");
Console.ReadKey();
}
}
catch (Exception ex)
{
Console.WriteLine("Error Message: " + ex.Message);
Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. Navigate to the List -> List Settings -> Permissions. Subsequently, check the permissions granted. In this example, if the list is inheriting the permissions from the parent site, the inheritance will break and unique permissions will be provided to the list.

The snapshot given below shows the list with the unique permissions and provides the access.



The screenshot shows the SharePoint 'PERMISSIONS' page for a specific list. A yellow warning banner at the top states: 'There are limited access users on this site. Users may have limited access if an item or document under the site has been shared with them. Show users. This list has unique permissions'. Below this, a table lists the permissions granted to the list. The 'Everyone' permission is highlighted with a red box.

Name	Type	Permission Levels
Approvers	SharePoint Group	Approve
Designers	SharePoint Group	Design
Everyone	Domain Group	Edit
Excel Services Viewers	SharePoint Group	View Only
Hierarchy Managers	SharePoint Group	Manage Hierarchy
learning Members	SharePoint Group	Edit
learning Owners	SharePoint Group	Full Control
learning Visitors	SharePoint Group	Read
Restricted Readers	SharePoint Group	Restricted Read
Translation Managers	SharePoint Group	Restricted Interfaces for Translation

6.8 How to enable versioning for the list

In this example, you will learn how to enable versioning for a SharePoint list, using PnP core CSOM library. The versioning for the list can be updated, using `UpdateListVersioning` method.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method are given below.
 - Enable versioning (Boolean).
 - Enable minor versioning (Boolean).
 - Update Query (Boolean).
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // List name input
        string listName = "TestList";
        // Get List
        List list = clientContext.Site.RootWeb.GetListByTitle(listName);
        // Parameters for enabling versioning
        bool enableVersioning = true; // Enable, false for disabling
        bool enableMinorVersioning = false;
        bool updateQuery = true;
        // Enables versioning
        list.UpdateListVersioning(enableVersioning, enableMinorVersioning, updateQuery);

        Console.WriteLine("Versioning enabled");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. Navigate to the List -> List Settings -> Version Settings and check the settings.

7 SharePoint List View Operations

In this section, you will learn how to work with the views, using PnP Core CSOM library. Some of the basic operations like, create, retrieve, update and delete view operations are explained.

The commands given below are used in the examples, which are explained in this section.

- CreateView
- GetViewByName
- GetViewById

7.1 How to create a list view

In this example, you will learn how to create a list view for a list on SharePoint site, using PnP Core CSOM library. SharePoint Custom list views will be created, using CreateView method.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method are given below.
 - View name
 - View fields array
 - Default view (Boolean)
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // List Name
        string listName = "TestList";

        // Get List
        List list = clientContext.Site.RootWeb.GetListByTitle(listName);
```

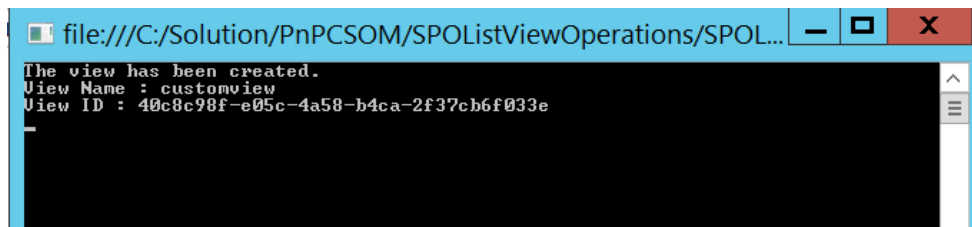
```
// Input for creating view
string viewName = "customview";
string[] viewFields = { "Title", "Author" };
bool defaultView = true;

// Creates New View
View newView = list.CreateView(viewName, ViewType.None, viewFields, 30,
defaultView, null, false, false);

// Output
Console.WriteLine("The view has been created.");
Console.WriteLine("View Name : " + newView.Title);
Console.WriteLine("View ID : " + newView.Id);
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

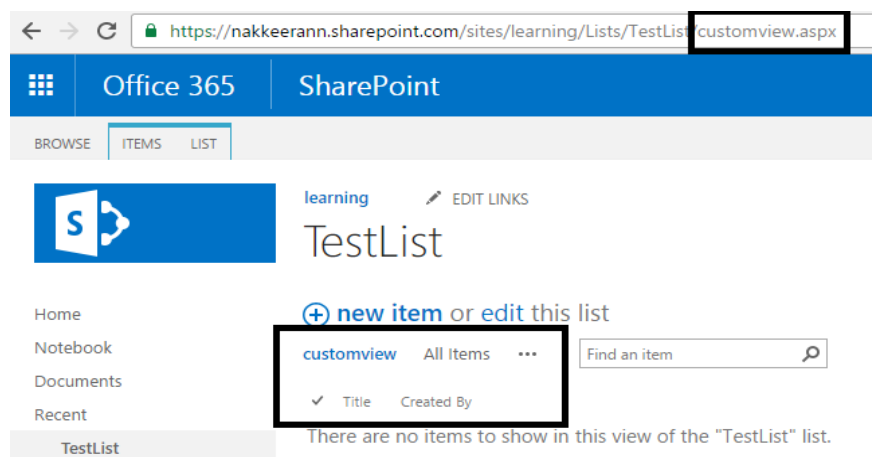
Result

The results will be displayed on the console, as shown below.



```
file:///C:/Solution/PnPCSOM/SPOListViewOperations/SPOL...
The view has been created.
View Name : customview
View ID : 40c8c98f-e05c-4a58-b4ca-2f37cb6f033e
```

The snapshot given below shows the list view created for a SharePoint list on the site.



7.2 How to retrieve a list view using view name

In this example, you will learn how to retrieve a particular list view by the view name from a SharePoint list on SharePoint site, using PnP Core CSOM library. Here, we will see how we can retrieve the list view object, using the list view name.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the method is the view name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

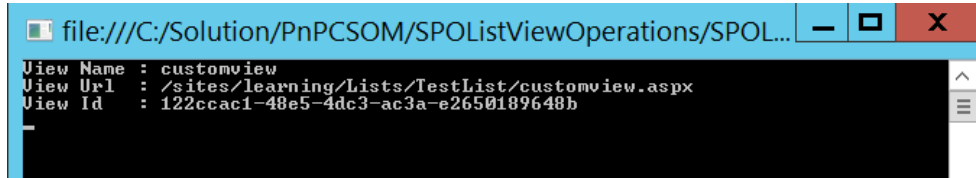
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Input Parameters
        // List Name
        string listName = "TestList";
        // View Name
        string viewName = "customview";

        // Get View By Name
        View view =
clientContext.Site.RootWeb.GetListByTitle(listName).GetViewByName(viewName);

        // Output
        Console.WriteLine("View Name : " + view.Title);
        Console.WriteLine("View Url : " + view.ServerRelativeUrl);
        Console.WriteLine("View Id : " + view.Id);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



```
file:///C:/Solution/PnP-CSOM/SPOListViewOperations/SPOL...
View Name : customview
View Url  : /sites/learning/Lists/TestList/customview.aspx
View Id   : 122ccac1-48e5-4dc3-ac3a-e2650189648b
```

7.3 How to retrieve a list view using view Id

In this example, you will learn how to retrieve a particular list view by Id from a SharePoint list, using PnP Core CSOM library. Here, we will see, how we can retrieve the list view object, using list name and view Id.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter for the method is the view ID.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {
        // Input Parameters
        // List Name
        string listName = "TestList";
        // View Id
        Guid viewId = new Guid("122ccac1-48e5-4dc3-ac3a-e2650189648b");

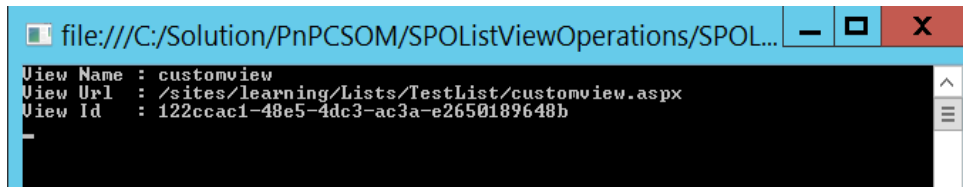
        // Get View By Id
        View view =
clientContext.Site.RootWeb.GetListByTitle(listName).GetViewById(viewId);

        // Output
```

```
        Console.WriteLine("View Name : " + view.Title);  
        Console.WriteLine("View Url : " + view.ServerRelativeUrl);  
        Console.WriteLine("View Id : " + view.Id);  
        Console.ReadKey();  
    }  
}  
catch (Exception ex)  
{  
    Console.WriteLine("Error Message: " + ex.Message);  
    Console.ReadKey();  
}
```

Result

The results will be displayed on the console, as shown below.



```
file:///C:/Solution/PnPCSOM/SPOListViewOperations/SPOL...  
View Name : customview  
View Url : /sites/learning/Lists/TestList/customview.aspx  
View Id : 122ccac1-48e5-4dc3-ac3a-e2650189648b
```

8 SharePoint Content Type Operations

In this section, you will learn how to complete the basic content type operations available, using PnP Core CSOM library. Some of the basic operations like create, retrieve, check if exists, add to the list and delete the content type operations, which are explained.

The content types can be accessed from the portal using

https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngctype.aspx.

The commands given below are used in the examples, which are explained in this section.

- `GetContentTypeByName`
- `GetContentTypeById`
- `CreateContentType`
- `CreateContentTypeFromXMLString`
- `ContentTypeExistsByName`
- `ContentTypeExistsById`
- `AddContentTypeToList`
- `AddContentTypeToListById`
- `AddContentTypeToListByName`
- `ContentTypeExistsByName`
- `ContentTypeExistsById`
- `DeleteContentTypeByName`
- `DeleteContentTypeById`

8.1 How to retrieve site content type by content type name

In this section, you will learn how to retrieve the required site content type from SharePoint site, using PnP Core CSOM library with the help of an existing content type name. The Item Site content type is retrieved in the example given below, using `GetContentTypeByName` method.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method is the content type name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

        // Input Parameters
        string contentTypeName = "Item";

        // Retrieves content type by name
        ContentType contentType =
clientContext.Site.RootWeb.GetContentTypeByName(contentTypeName);

        // Output
        Console.WriteLine("Content Type Details:");
        Console.WriteLine("Name : " + contentType.Name);
        Console.WriteLine("ID : " + contentType.Id);
        Console.WriteLine("Description : " + contentType.Description);
        Console.WriteLine("Group : " + contentType.Group);
        Console.WriteLine("XML : " + contentType.SchemaXml);
        Console.ReadKey();

    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. The item site content type information is retrieved from the site.

```
file:///C:/Solution/PnP/SOM/ContentTypeOperations/Cont...
Content Type Details:
Name : Item
ID : 0x01
Description : Create a new list item.
Group : List Content Types
XML : <ContentType ID="0x01" Name="Item" Group="List Content Types" Description="
"Create a new list item." Version="0" FeatureId="{695b6570-a48b-4a8e-8ea5-26ea7f
c1d162}"><Folder TargetName="_cts/Item" /><Fields><Field ID="{c042a256-787d-4a6f
-8a8a-cf6ab767f12d}" Name="ContentType" SourceID="http://schemas.microsoft.com/s
harepoint/v3" StaticName="ContentType" Group="_Hidden" Type="Computed" DisplayNa
me="Content Type" Sealed="TRUE" Sortable="FALSE" RenderXMLUsingPattern="TRUE" PI
Target="MicrosoftWindowsSharePointServices" PIAttribute="ContentTypeID" Customiz
ation=""><FieldRefs><FieldRef ID="{03e45e84-1992-4d42-9116-26f756012634}" Name="
ContentTypeID" /></FieldRefs><DisplayPattern><MapToContentType><Column Name="Con
tentTypeId" /></MapToContentType></DisplayPattern></Field><Field ID="{fa564e0f-0
c70-4ab9-b863-0177e6ddd247}" Name="Title" SourceID="http://schemas.microsoft.com
/sharepoint/v3" StaticName="Title" Group="_Hidden" Type="Text" DisplayName="Titl
e" Required="TRUE" FromBaseType="TRUE" Customization="" ShowInNewForm="TRUE" Sho
wInEditForm="TRUE"></Field></Fields><XmlDocuments><XmlDocument NamespaceURI="htt
p://schemas.microsoft.com/sharepoint/v3/contenttype/forms"><FormTemplates xmlns=
"http://schemas.microsoft.com/sharepoint/v3/contenttype/forms"><Display>ListForm
</Display><Edit>ListForm</Edit><New>ListForm</New></FormTemplates></XmlDocument>
</XmlDocuments></ContentType>
```

Note

Similarly, other field values can also be retrieved.

8.2 How to retrieve site content type by content type Id

In this section, you will learn how to retrieve the required site content type by content type Id from SharePoint site, using PnP Core CSOM library. Item Site content type is retrieved in the example given below, using `GetContentTypeById` method.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameter is the content type ID. The default item content type id is 0x01.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
```

```
{
    // Input Parameters
    string contentTypeId = "0x01";

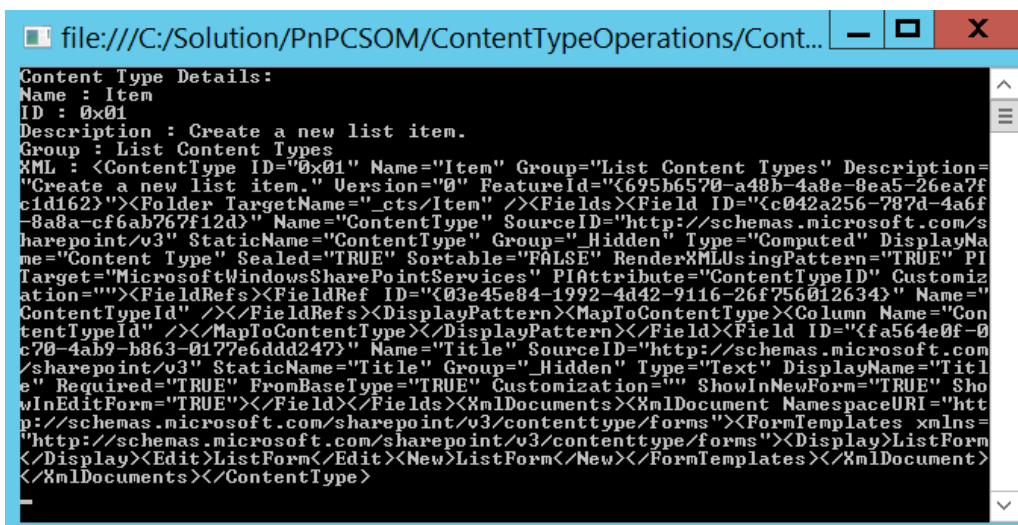
    // Retrieves content type by id
    ContentType contentType =
clientContext.Site.RootWeb.GetContentTypeById(contentTypeId);

    // Output
    Console.WriteLine("Content Type Details:");
    Console.WriteLine("Name : " + contentType.Name);
    Console.WriteLine("ID : " + contentType.Id);
    Console.WriteLine("Description : " + contentType.Description);
    Console.WriteLine("Group : " + contentType.Group);
    Console.WriteLine("XML : " + contentType.SchemaXml);
    Console.ReadKey();

}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. The item site content type information is retrieved from the site.



8.3 How to create a site content type

In this section, you will learn how to create a new site content type on SharePoint site using PnP Core CSOM library. The scope of the content type is defined in the Web object, where the method is initiated. The scope can also be identified, which is based on the site URL.

In the example given below, site collection URL is used for setting the context. The create site content type method is initiated, using the root level Web object. Thus, the scope of the content type to be created is the site.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The parent content type should be retrieved for assigning it to the new content type, since the parent content type will be considered as base content type.
- The required input parameters should be set.
 - Content type name
 - Content type description
 - Content type Id – If not decided, then it can be null
 - Content type group
 - Parent content type
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

        // Input Parameters
        string contentTypeName = "NakkeeranCT";
        string contentTypeDesc = "Test Content Type using PnP CSOM";
        string contentTypeId = null;
        string contentTypeGroup = "CustomCTGroup";

        // Input Parameter - Item Content Type Id
        string parentContentTypeId = "0x01";

        // Retrieves content type by id
```

```

        ContentType parentContentType =
clientContext.Site.RootWeb.GetContentTypeById(parentContentTypeId);

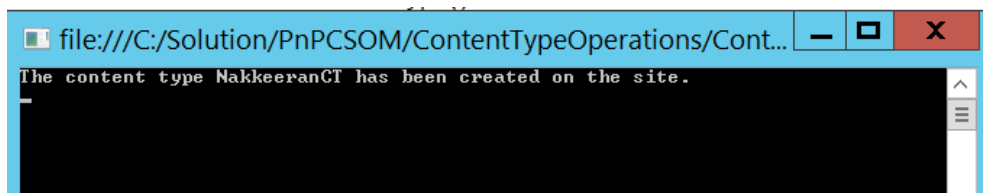
        // Creates New Content Type
        ContentType contentType =
clientContext.Site.RootWeb.CreateContentType(contentTypeName, contentTypeDesc,
contentTypeId, contentTypeGroup, parentContentType);

        // Output
        Console.WriteLine("The content type " + contentType.Name + " has been created on the
site.");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

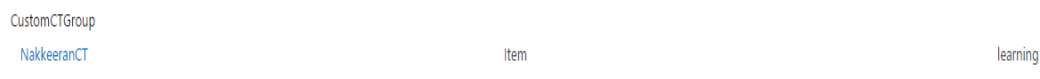
```

Result

The results will be displayed on the console, as shown below.



The snapshot given below shows the content type created on the site. The content types can be found on the site content type's page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngctype.aspx).



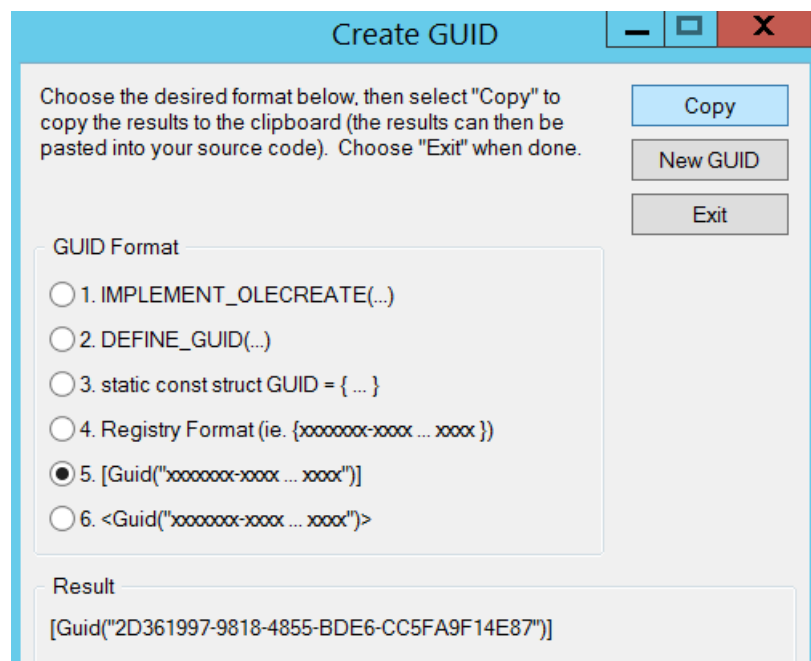
8.4 How to create a site content type using content type XML

In this section, you will learn how to create a new site content type on SharePoint site by content type XML, using PnP Core CSOM library.

In the example given below, site collection URL is used for setting the context. The content type XML is built and is used as an input parameter to create the content type in this approach. The create site content type method is initiated, using the root level Web object. Thus, the scope of the content type to be created is the site.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The content type XML is built. The content type Id is generated with the help of Visual Studio.
 - From the Visual Studio menu options, click on Tools → Create GUID.
 - GUID dialog box will open, as shown below. Leave the default option, click on New GUID and then click on copy. The GUID will be copied to the clipboard.



- The GUID will be [Guid("2D361997-9818-4855-BDE6-CC5FA9F14E87")]. Select only the alpha numeric data and append it to the parent content type Id. The content type ID will be 0x01002D36199798184855BDE6CC5FA9F14E87 in this example.

- The content type XML will be, as shown below.

```
<ContentType ID ='0x01002D36199798184855BDE6CC5FA9F14E87'  
Name='NakkeeranCT1' Group='CustomCTGroup' Description='Content Type using  
XML' Inherits='TRUE'>  
    <FieldRefs>  
    </FieldRefs>  
</ContentType>
```

The necessary fields can be inserted within the FieldRefs tag, using field tags.

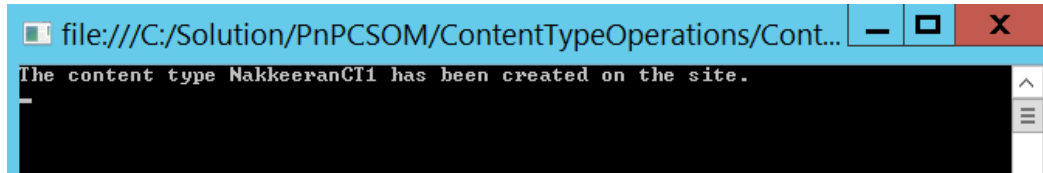
- In the program file, insert the code mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

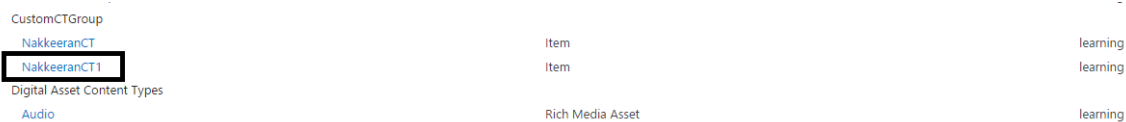
```
try  
{  
    // Get and set the client context  
    // Connects to SharePoint online site using inputs provided  
    using (var clientContext =  
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))  
    {  
        // Input Parameters - Content Type XML  
        string contentTypeXml = "<ContentType ID =  
'0x01002D36199798184855BDE6CC5FA9F14E87' Name='NakkeeranCT1'  
Group='CustomCTGroup' Description='Content Type using XML' Inherits='TRUE'>"  
            + "<FieldRefs>"  
            + "</FieldRefs>"  
            + "</ContentType>";  
        // Creates Content Type using XML string  
        ContentType contentType =  
clientContext.Site.RootWeb.CreateContentTypeFromXMLString(contentTypeXml);  
  
        // Output  
        Console.WriteLine("The content type " + contentType.Name + " has been created on the  
site.");  
        Console.ReadKey();  
    }  
}  
catch (Exception ex)  
{  
    Console.WriteLine("Error Message: " + ex.Message);  
    Console.ReadKey();  
}
```

Result

The results will be displayed on the console, as shown below.



The snapshot given below shows the content type created on the site. The content types can be found on the site content type's page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngctype.aspx).



Note

This approach can only be used for the development purposes, since the content type Id is generated manually.

8.5 How to check if a site content type already exists by content type name

In this section, you will learn how to check, if a site content type already exists by content type name on SharePoint site, using PnP Core CSOM library.

In the example given below, site collection URL is used for setting the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters should be set.
 - Content type name
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

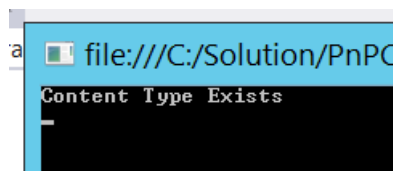
        // Input Parameters – Content Type Name
        string contentTypeName = "NakkeeranCT";

        // Checks if Content Type exists by Name
        bool contentTypeExists =
clientContext.Site.RootWeb.ContentTypeExistsByName(contentTypeName);

        // Output
        if (contentTypeExists)
        {
            Console.WriteLine("Content Type Exists");
        }
        else
        {
            Console.WriteLine("Content Type doesn't exists");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



The content types can be found on the site content type's page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngctype.aspx).

8.6 How to check if a site content type already exists by content type Id

In this section, you will learn how to check, if a site content type already exists by the content type Id on SharePoint site, using PnP Core CSOM library.

In the following example, site collection URL is used for setting the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters should be set.
 - Content type Id
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

        // Input Parameters – Content Type Id
        string contentTypeId = "0x0100C6D6909EB935D44F88615A32BB60B9F9";

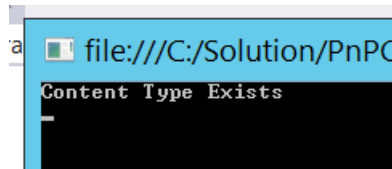
        // Checks if Content Type exists by Id
        bool contentTypeExists =
clientContext.Site.RootWeb.ContentTypeExistsById(contentTypeId);

        // Output
        if (contentTypeExists)
        {
            Console.WriteLine("Content Type Exists");
        }
        else
        {
            Console.WriteLine("Content Type doesn't exists");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
}
```

```
Console.WriteLine("Error Message: " + ex.Message);
Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



The content types can be found on the site content type's page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngctype.aspx).

8.7 How to add site content type to List

In this section, you will learn how to add the site content type (created above) to SharePoint list available on the site, using PnP Core CSOM library.

1. Retrieve the list where the content type is being added.
2. Retrieve the content type.
3. Add the content type to list.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method are given below.
 - Content type
 - Is default content type
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
}
```

```
using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
{
    // Input Parameters - Content Type Name
    string contentTypeName = "NakkeeranCT";
    string listUrl = "Lists/TestList"; // List Server Relative Url
    bool defaultContentType = true; // Set it as default content type on list

    // Get Content type by name
    ContentType contentType =
clientContext.Site.RootWeb.GetContentTypeByName(contentTypeName);
    // Get List by Url
    List list = clientContext.Site.RootWeb.GetListByUrl(listUrl);
    // Add Content Type to List
    bool contentTypeAdd = list.AddContentTypeToList(contentType, defaultContentType);

    if (contentTypeAdd)
    {
        Console.WriteLine("Content Type Added to List");
    }
    else
    {
        Console.WriteLine("Content Type Not added");
    }
    // Output
    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. You can navigate to the list settings to check, if the content type is added successfully or not.

TestList ▸ Settings

List Information

Name:

TestList

Web Address:

https://nakkeerann.sharepoint.com/sites/learning/Lists/TestList/AllItems.aspx

Description:

General Settings

List name, description and navigation

Versioning settings

Advanced settings

Validation settings

Audience targeting settings

Rating settings

Form settings

Permissions and Management

Delete this list

Save list as template

Permissions for this list

Workflow Settings

Generate file plan report

Enterprise Metadata and Keywords Settings

Information management policy settings

Communications

RSS settings

Content Types

This list is configured to allow multiple content types. Use content types to specify the information you want to display about an item, in addition to its policies, workflows, or other behavior. The following content types are currently available in this list:

Content Type	Visible on New Button	Default Content Type
NakkeerannCT	✓	✓
Item	✓	

8.8 How to add site content type to List By content type Id

In this section, you will learn how to add the site content type (created above) to the SharePoint list available on the site by content type Id, using PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method are given below.
 - Content type Id.
 - Is default content type.
- In the program file, insert the code given below and save the file.

Code

```

try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // Input Parameters - Content Type Id
        string contentTypeId = "0x0100C6D6909EB935D44F88615A32BB60B9F9";
        bool defaultCT = false;
        string listUrl = "Lists/TestList"; // List Server Relative Url
        // Get List by Url
        List list = clientContext.Site.RootWeb.GetListByUrl(listUrl);
    }
}

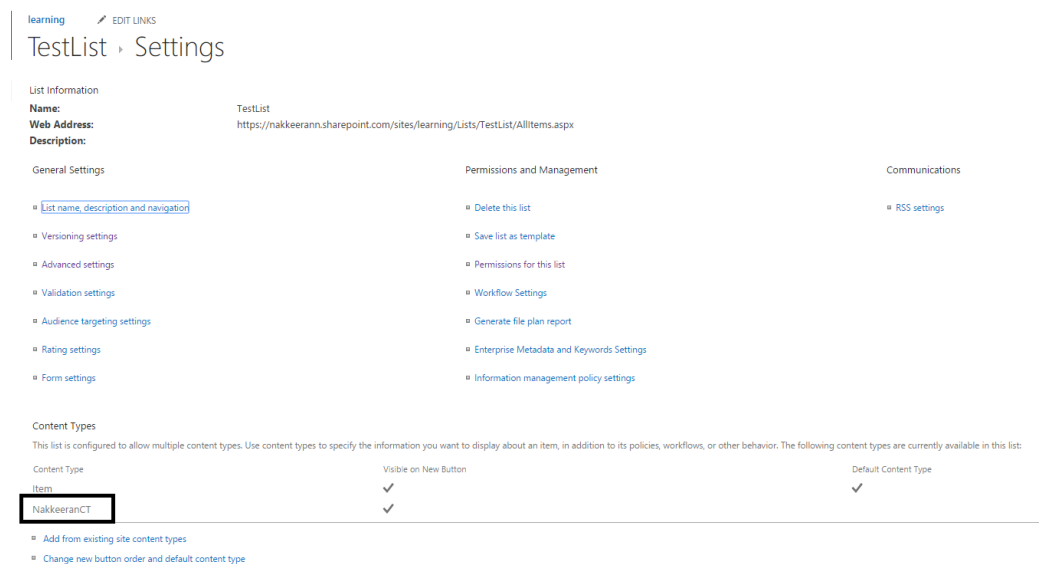
```

```
// Add Content Type to List by Id
bool contentTypeAdd = list.AddContentTypeToListById(contentTypeId, defaultCT);
if (contentTypeAdd)
{
    Console.WriteLine("Content Type Added to List");
}
else
{
    Console.WriteLine("Content Type Not added");
}
// Output
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

- Run the code from Debug menu or by pressing F5.

Result

The results will be displayed on the console. You can navigate to the list settings to check, if the content type is added successfully.



learning EDIT LINKS

TestList > Settings

List Information

Name: TestList
Web Address: https://nakkeerann.sharepoint.com/sites/learning/Lists/TestList/AllItems.aspx
Description:

General Settings Permissions and Management Communications

- List name, description and navigation
- Versioning settings
- Advanced settings
- Validation settings
- Audience targeting settings
- Rating settings
- Form settings
- Delete this list
- Save list as template
- Permissions for this list
- Workflow Settings
- Generate file plan report
- Enterprise Metadata and Keywords Settings
- Information management policy settings
- RSS settings

Content Types

This list is configured to allow multiple content types. Use content types to specify the information you want to display about an item, in addition to its policies, workflows, or other behavior. The following content types are currently available in this list:

Content Type	Visible on New Button	Default Content Type
Item	✓	✓
NakkeeranCT	✓	

- Add from existing site content types
- Change new button order and default content type

8.9 How to add site content type to List By content type name

In this section, you will learn how to add the site content type (created above) to SharePoint list available on the site by content type name using PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters for the method are given below.
 - Content type Id.
 - Is default content type.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

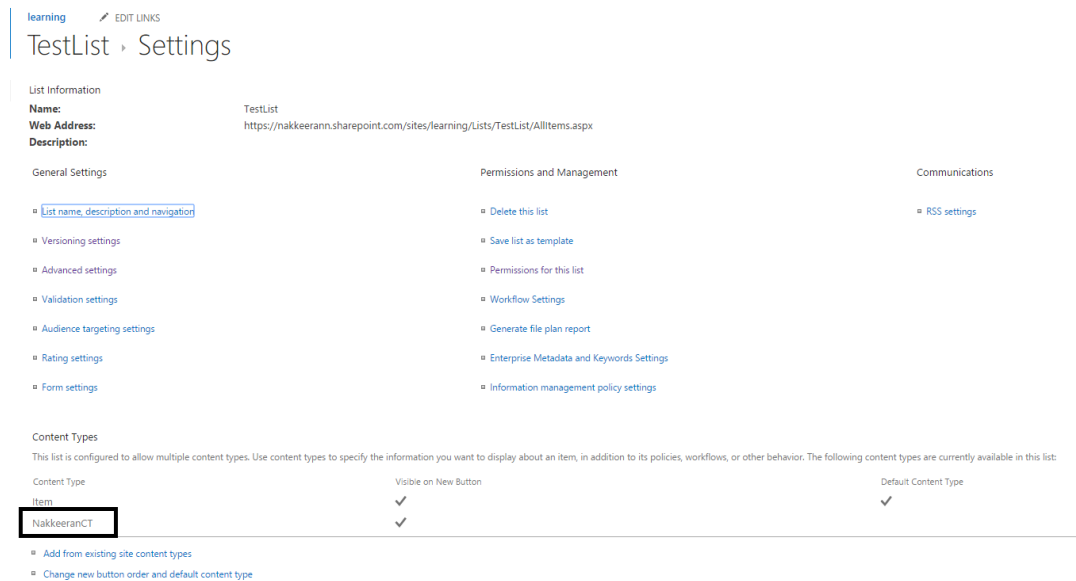
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

        // Input Parameters - Content Type Name
        string contentTypeName = "NakkeeranCT";
        bool defaultCT = false;
        string listUrl = "Lists/TestList"; // List Server Relative Url
        // Get List by Url
        List list = clientContext.Site.RootWeb.GetListByUrl(listUrl);
        // Add Content Type to List by Name
        bool contentTypeAdd = list.AddContentTypeToListByName(contentTypeName,
defaultCT);
        if (contentTypeAdd)
        {
            Console.WriteLine("Content Type Added to List");
        }
        else
        {
            Console.WriteLine("Content Type Not added");
        }
        // Output
        Console.ReadKey();
    }
}
catch (Exception ex)
```

```
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. You can navigate to the list settings to check, if the content type is added successfully or not.



The screenshot shows the 'TestList' settings page in SharePoint. The 'Content Types' section is expanded, showing a table of content types. The 'NakkeeranCT' content type is listed with a checkmark in the 'Visible on New Button' column and a checkmark in the 'Default Content Type' column. The 'Item' content type is also listed with a checkmark in the 'Visible on New Button' column.

Content Type	Visible on New Button	Default Content Type
Item	✓	✓
NakkeeranCT	✓	✓

8.10 How to check if a content type already exists on a list by the content type name

In this section, you will learn how to check, if a site content type already exists by the content type name on SharePoint list, using PnP Core CSOM library.

In the example given below, site collection URL is used for setting the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters should be set.
 - List URL.
 - Content type name.
- In the program file, insert the code given below and save the file.

- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

        // Input Parameters
        string contentTypeName = "NakkeeranCT";
        string listUrl = "Lists/TestList"; // List Server Relative Url

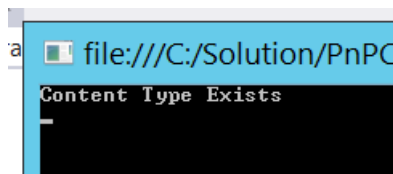
        // Get List by Url
        List list = clientContext.Site.RootWeb.GetListByUrl(listUrl);

        // Checks if content type exists in list by content type name
        bool contentTypeExists = list.ContentTypeExistsByName(contentTypeName);

        // Output
        if (contentTypeExists)
        {
            Console.WriteLine("Content Type Exists");
        }
        else
        {
            Console.WriteLine("Content Type doesn't exists");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. The same can be checked by navigating to the list settings.



8.11 How to check if a content type already exists on a List by content type Id

In this section, you will learn how to check, if a site content type already exists by the content type Id on SharePoint list, using PnP Core CSOM library.

In the example given below, the site collection URL is used to set the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters should be set.
 - List URL
 - Content type Id
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext =
authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl, userName, password))
    {

        // Input Parameters
        string contentTypeId = "0x0100C6D6909EB935D44F88615A32BB60B9F9";
        string listUrl = "Lists/TestList"; // List Server Relative Url

        // Get List by Url
        List list = clientContext.Site.RootWeb.GetListByUrl(listUrl);

        // Checks if content type exists in list by content type Id
        bool contentTypeExists = list.ContentTypeExistsById(contentTypeId);

        // Output
    }
}
```

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

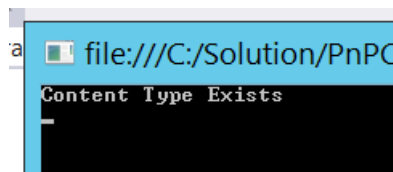
```

        if (contentTypeExists)
        {
            Console.WriteLine("Content Type Exists");
        }
        else
        {
            Console.WriteLine("Content Type doesn't exists");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
}

```

Result

The results will be displayed on the console, as shown below. The same can be checked by navigating to the list settings.



8.12 How to delete a content type by the name

In this section, you will learn how to delete a content type from the SharePoint site by the content type name, using PnP Core CSOM library.

In the example given below, the site collection URL is used to set the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The required input parameters should be set.
 - Content type name.

- In the program file, insert the code mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {

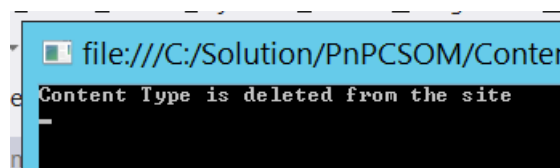
        // Input Parameters - Content Type Name
        string contentTypeName = "NakkeeranCT";

        // Deletes Content Type by content type Name
        clientContext.Site.RootWeb.DeleteContentTypeByName(contentTypeName);

        Console.WriteLine("Content Type is deleted from the site");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. The same can be checked by navigating to the content type page.



8.13 How to delete a content type by Id

In this section, you will learn how to delete a content type from the SharePoint site by content type Id, using PnP Core CSOM library.

In the example given below, the site collection URL is used to set the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book for importing the packages and creating authentication manager object.
- The required input parameters should be set.
 - Content type Id
- In the program file, insert the following code and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {

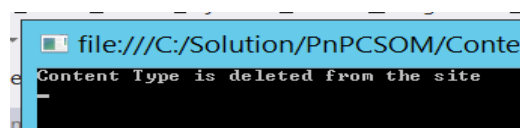
        // Input Parameters - Content Type Id
        string contentTypeId = "0x0100210247D8D96D9F4CAC902CF835C12919";

        // Deletes Content Type by content type Id
        clientContext.Site.RootWeb.DeleteContentTypeById(contentTypeId);

        Console.WriteLine("Content Type is deleted from the site");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below. The same can be checked by navigating to the content type page.



9 SharePoint Field Operations

In this topic, you will learn how to perform the basic field operations like add, retrieve or delete to/from the sites/content types, using PnP Core CSOM library. Fields are otherwise called as columns.

The methods used for the operations are,

- CreateField
- CreateFieldsFromXMLString
- GetFieldById
- FieldExistsById
- FieldExistsByName
- AddFieldToContentType
- AddFieldToContentTypeById
- AddFieldToContentTypeByName
- FieldExistsByNameInContentType

The fields can be accessed manually from the site columns page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngfield.aspx).

9.1 How to create a field on SharePoint site

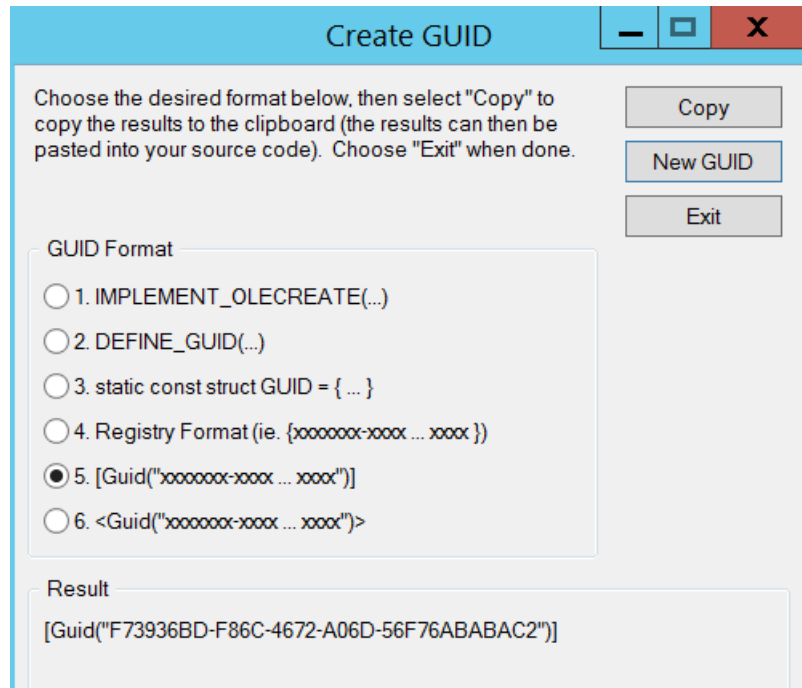
In this section, you will learn how to create a site column (field) on SharePoint site, using PnP Core CSOM library. CreateField method is used to create columns with the necessary input parameters.

In the example given below, the site collection URL is used for setting the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in section 4 (Connect to SharePoint site) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Display Name
 - Group Name
 - Field Id
 - Field Type
- The field GUID needs to be generated.

- From Visual Studio menu options, click on Tools → Create GUID.
- GUID dialog box will open as shown below. Leave the default option, click on New GUID and then click on copy. GUID will be copied to the clip board.



- GUID will be [Guid("F73936BD-F86C-4672-A06D-56F76ABABAC2")].
Select only the alpha numeric data and assign it to a variable. The field ID will be F73936BD-F86C-4672-A06D-56F76ABABAC2 in this example.
- The field creation information has to be created, which will contain the necessary input parameters to create the field/column.
- Create Field method is used from the Web object to create the fields, where field creation information object is used as an input parameter.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string displayName = "NakkeeranColumn1";
```

```

string internalName = "NakkeeranColumn1";
string groupName = "NavColumnGroup";
string fieldId = "F73936BD-F86C-4672-A06D-56F76ABABAC2";
FieldType fieldType = FieldType.Text;

// Input - New field creation info object
OfficeDevPnP.Core.Entities.FieldCreationInformation newFieldInfo = new
OfficeDevPnP.Core.Entities.FieldCreationInformation(fieldType)
{
    DisplayName = displayName,
    InternalName = internalName,
    Id = new Guid(fieldId),
    Group = groupName
};

// Creates new Field
Field newField = clientContext.Site.RootWeb.CreateField(newFieldInfo);

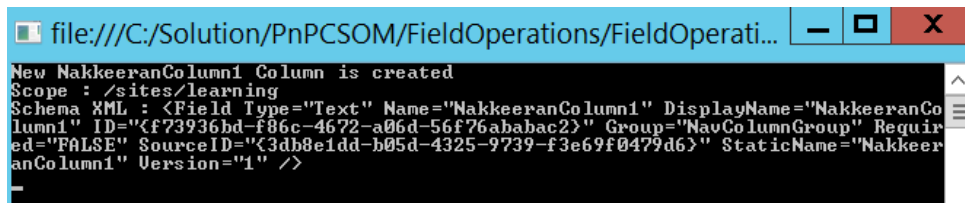
// Output
Console.WriteLine("New " + newField.Title + " Column is created");
Console.WriteLine("Scope : " + newField.Scope);
Console.WriteLine("Schema XML : " + newField.SchemaXml);

Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
}

```

Result

The results will be displayed on the console, as shown below.



```

file:///C:/Solution/PnPCSOM/FieldOperations/FieldOperati...
New NakkeeranColumn1 Column is created
Scope : /sites/learning
Schema XML : <Field Type='Text' Name='NakkeeranColumn1' DisplayName='NakkeeranColumn1' ID='{f73936bd-f86c-4672-a06d-56f76ababac2}' Group='NavColumnGroup' Required='FALSE' SourceID='{3db8e1dd-b05d-4325-9739-f3e69f0479d6}' StaticName='NakkeeranColumn1' Version='1' />

```

The created field can be viewed on the site columns page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngfield.aspx). The snapshot given below shows the field created on the site column page.

Target List Template ID	Single line of text	learning
Target Scope	Single line of text	learning
NavColumnGroup		
NakkeeranColumn1	Single line of text	learning
Page Layout Columns		
Byline	Single line of text	learning

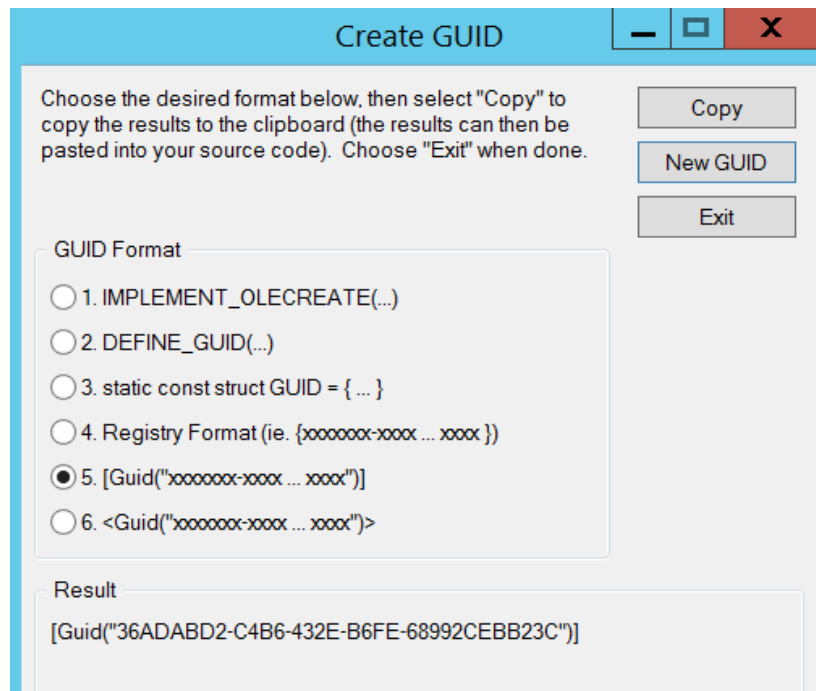
9.2 How to create a field on SharePoint site using Field XML

In this section, you will learn how to create a site column (field) on SharePoint site, using PnP Core CSOM library by field XML. CreateFieldsFromXMLString method is used to create the columns with the necessary XML.

In the example given below, the site collection URL is used for setting the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The field XML should contain the following.
 - Title or name.
 - Display name.
 - Group name.
 - Type.
 - GUID.
- The field GUID needs to be generated.
 - From Visual Studio menu options, click on Tools → Create GUID.
 - GUID dialog box will open, as shown below. Leave the default option, click on New GUID and then click on copy. GUID will be copied to the clip board.



- The GUID will be [Guid("36ADABD2-C4B6-432E-B6FE-68992CEBB23C")]. Select only the alpha numeric data and assign it to a variable. The field ID will be 36ADABD2-C4B6-432E-B6FE-68992CEBB23C in this example.
- Create Field method is used from the Web object to create the fields, where the field XML is used as an input parameter.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter - New Field XML as string
        string fieldXML = "<Field Type =\"Text\" Name=\"NakkeeranColumn4\"
        DisplayName=\"NakkeeranColumn4\" ID=\"{ 36ADABD2-C4B6-432E-B6FE-68992CEBB23C}\"
        Group =\"NavColumnGroup\" Required=\"FALSE\" />";

        // Creates new Field
        clientContext.Site.RootWeb.CreateFieldsFromXMLString(fieldXML);

        // Output
        Console.WriteLine("New Column is created");
    }
}
```

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

```

    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The success/failure results will be displayed on the console.

The created field can be viewed on the site columns page (https://nakkeerann.sharepoint.com/sites/learning/_layouts/15/mngfield.aspx). The snapshot given below shows the field created on the site column page.

Target Scope	Single line of text	learning
NavColumnGroup		
NakkeeranColumn1	Single line of text	learning
NakkeeranColumn4	Single line of text	learning
Page Layout Columns		
Byline	Single line of text	learning

9.3 How to retrieve a field using field Id

In this section, you will learn how to retrieve the required field (columns), using the field Id from SharePoint site by PnP Core CSOM library.

Steps involved

- Open Visual Studio as administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Id as GUID variable.
- The field can be retrieved, using GetFieldById method.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {

        // Input Parameter
        string fieldIdStr = "F73936BD-F86C-4672-A06D-56F76ABABAC2";
        // As Guid
        Guid fieldId = new Guid(fieldIdStr);

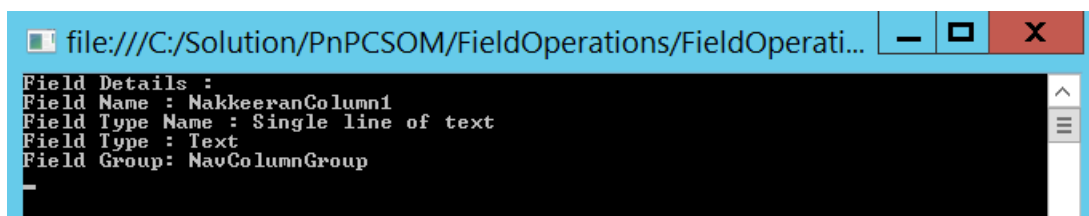
        // Retrieves Field
        Field field = clientContext.Site.RootWeb.GetFieldById<Field>(fieldId);

        // Output
        Console.WriteLine("Field Details : ");
        Console.WriteLine("Field Name : " + field.Title);
        Console.WriteLine("Field Type Name : " + field.TypeDisplayName);
        Console.WriteLine("Field Type : " + field.FieldTypeKind);
        Console.WriteLine("Field Group: "+field.Group);
        Console.ReadKey();

    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



9.4 How to check if a field exists using field Id

In this section, you will learn how to check, if the field is already available, using the field Id from SharePoint site by PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Id as GUID variable.
- The existing field can be checked, using FieldExistsById method, using the Web object.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string fieldIdStr = "F73936BD-F86C-4672-A06D-56F76ABABAC2";
        // As Guid
        Guid fieldId = new Guid(fieldIdStr);

        // Checks Field
        bool fieldExists = clientContext.Site.RootWeb.FieldExistsById(fieldId);

        // Output
        if (fieldExists)
        {
            Console.WriteLine("Field is available");
        }
        else
        {
            Console.WriteLine("Field is not available");
        }
        Console.ReadKey();
    }
}
```

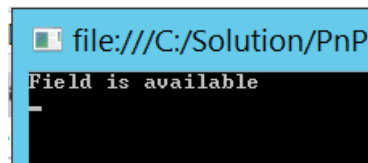
©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

```
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



9.5 How to check if a field exists using field Name

In this section, you will learn how to check, if the field is already available, using field title/name from SharePoint site by PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Title.
- The existing field can be checked, using FieldExistsByName method, using the Web object.
- In the program file, insert the code snippet given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
```

```

string fieldName = "NakkeeranColumn1";

// Checks Field
bool fieldExists = clientContext.Site.RootWeb.FieldExistsByName(fieldName);

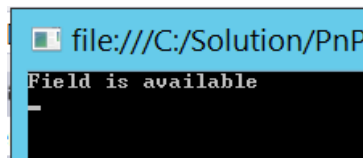
// Output
if (fieldExists)
{
    Console.WriteLine("Field is available");
}
else
{
    Console.WriteLine("Field is not available");
}
Console.ReadKey();

}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The results will be displayed on the console, as shown below.



9.6 How to add a field to site content type

In this section, you will learn how to add an existing site column to the site content type by the field object and the content type object, using PnP Core CSOM library. The existing site column is added to the existing site content type in the example given below.

In the example given below, the site collection URL is used to set the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Id – Field to be added.
 - Content Type Name – Content type, where the field is added.
 - Required – Field needs to be mandatory column?
 - Hidden – Field needs to be hidden on the lists/content types?
- Field is retrieved, using the field Id.
- Content Type is retrieved, using the content type name.
- Then AddFieldToContentType method is used to add the field to the content type with the help of the field and the content type objects retrieved.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string fieldIdStr = "F73936BD-F86C-4672-A06D-56F76ABABAC2";
        string contentTypeName = "NakkeeranCT";
        bool isRequired = false;
        bool isHidden = false;

        // Field As Guid
        Guid fieldId = new Guid(fieldIdStr);
        // Retrieves Field
        Field field = clientContext.Site.RootWeb.GetFieldById<Field>(fieldId);
```



```
// Retrieves Content Type
ContentType contentType =
clientContext.Site.RootWeb.GetContentTypeByName(contentTypeName);

// Adds Field to Content Type using Name
clientContext.Site.RootWeb.AddFieldToContentType(contentType, field, isRequired, isHidden);

// Output
Console.WriteLine("Field is added to Content Type");
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console given below. The result can be viewed by opening the “NakkeeranCT” from the site content type’s page. The URL for the content type’s page is https://nakkeerann.sharepoint.com/_layouts/15/mngctype.aspx. The snapshot given below shows the field added to the content type.

Site Content Types › Site Content Type

Site Content Type Information
Name: NakkeeranCT
Description: Test Content Type using PnP CSOM
Parent: Item
Group: CustomCTGroup

Settings

- ▣ Name, description, and group
- ▣ Advanced settings
- ▣ Workflow settings
- ▣ Delete this site content type
- ▣ Information management policy settings

Columns

Name	Type	Status	Source
Title	Single line of text	Required	Item
NakkeeranColumn1	Single line of text	Optional	

- ▣ Add from existing site columns
- ▣ Add from new site column
- ▣ Column order

9.7 How to add a field to site content type using Content type Id

In this section, you will learn how to add an existing site column to the site content type directly with the help of the field Id and the content type Id, using PnP Core CSOM library. The existing site column is added to the existing site content type in the example given below.

In the example given below, the site collection URL is used for setting the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Id – Field to be added
 - Content Type Id – Content type, where the field is added
 - Required – Field needs to be mandatory column
 - Hidden – Field needs to be hidden on the lists/content types
- Subsequently, `AddFieldToContentTypeById` method is used to add the field to the content type, using the field Id with the help of the web object.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string fieldId = "F73936BD-F86C-4672-A06D-56F76ABABAC2";
        string contentTypeId = "0x01006699DA2F8D486148BB11DE9FEFA816FA";
        bool isRequired = false;
        bool isHidden = false;

        // Adds Field to Content Type using ID

        clientContext.Site.RootWeb.AddFieldToContentTypeById(contentTypeId,fieldId,isRequired,isHidden);

        // Output
    }
}
```

```

        Console.WriteLine("Field is added to Content Type");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The results will be displayed on the console. The result can be viewed by opening the “NakkeeranCT” from the site content type’s page. The URL for the content type’s page is https://nakkeerann.sharepoint.com/_layouts/15/mngctype.aspx. The snapshot given below shows the field added to the content type.

Site Content Types › Site Content Type

Site Content Type Information
Name: NakkeeranCT
Description: Test Content Type using PnP CSOM
Parent: Item
Group: CustomCTGroup

Settings

- ▀ Name, description, and group
- ▀ Advanced settings
- ▀ Workflow settings
- ▀ Delete this site content type
- ▀ Information management policy settings

Columns

Name

Title

NakkeeranColumn1

▀ Add from existing site columns

▀ Add from new site column

▀ Column order

Type

Single line of text

Single line of text

Status

Required

Optional

Source

Item

9.8 How to add a field to site content type using Content type name

In this section, you will learn how to add existing site column to the site content type with the help of the field Id and the content type name, using PnP Core CSOM library. The existing site column is added to the existing site content type in the example given below.

In the example given below, the site collection URL is used to set the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Id – Field to be added.
 - Content Type Name – Content type, where the field is added
 - Required – Field needs to be the mandatory column?
 - Hidden – Field needs to be hidden on the lists/content types?
- Then AddFieldToContentTypeByName method is used to add the field to the content type, using field Id with the help of the Web object.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string fieldIdStr = "F73936BD-F86C-4672-A06D-56F76ABABAC2";
        string contentTypeName = "NakkeeranCT";
        bool isRequired = false;
        bool isHidden = false;

        // Field As Guid
        Guid fieldId = new Guid(fieldIdStr);

        // Adds Field to Content Type using Name
        clientContext.Site.RootWeb.AddFieldToContentTypeByName(contentTypeName, fieldId,
            isRequired, isHidden);
    }
}
```

```
// Output
Console.WriteLine("Field is added to Content Type");
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The result can be viewed by opening the “NakkeeranCT” from the site content type page. The URL for the content type’s page is https://nakkeerann.sharepoint.com/_layouts/15/mngctype.aspx. The snapshot given below shows the field added to the content type.

Site Content Types · Site Content Type

Site Content Type Information

Name: NakkeeranCT
Description: Test Content Type using PnP CSOM
Parent: Item
Group: CustomCTGroup

Settings

- Name, description, and group
- Advanced settings
- Workflow settings
- Delete this site content type
- Information management policy settings

Columns

Name	Type	Status	Source
Title	Single line of text	Required	Item
NakkeeranColumn1	Single line of text	Optional	

- Add from existing site columns
- Add from new site column
- Column order

9.9 How to check if a field exists in the Content type

In this section, you will learn how to check if the field already exists in the content type, using PnP Core CSOM library.

In the example given below, the site collection URL is used to set the context.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Field Name.
 - Content Type Name.
- Then `FieldExistsByNameInContentType` method is used to check, if the field already exists with the help of the `Web` object.
- In the program file, insert the code, mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {

        // Input Parameter
        string fieldName = "NakkeeranColumn1";
        string contentTypeName = "NakkeeranCT";
        // Checks Field exists in CT
        bool fieldExists =
            clientContext.Site.RootWeb.FieldExistsByNameInContentType(contentTypeName, fieldName);

        // Output
        if (fieldExists)
        {
            Console.WriteLine("Field is available");
        }
        else
        {
            Console.WriteLine("Field is not available");
        }
        Console.ReadKey();
    }
}
```

```

    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The results will be displayed on the console. The result can be viewed by opening the “NakkeeranCT” from the site content type page. The URL for the content type’s page is https://nakkeerann.sharepoint.com/_layouts/15/mngctype.aspx. The snapshot given below shows the field added to the content type.

Site Content Types · Site Content Type

Site Content Type Information
Name: NakkeeranCT
Description: Test Content Type using PnP CSOM
Parent: Item
Group: CustomCTGroup

Settings

- ▀ Name, description, and group
- ▀ Advanced settings
- ▀ Workflow settings
- ▀ Delete this site content type
- ▀ Information management policy settings

Columns

Name	Type	Status	Source
Title	Single line of text	Required	Item
NakkeeranColumn1	Single line of text	Optional	

▀ Add from existing site columns

▀ Add from new site column

▀ Column order

10 SharePoint Folder tasks

In this topic, you will learn how to perform the basic folder operations like add, retrieve or delete to/from libraries, using PnP Core CSOM library.

The methods given below are used for the folder operations are

- CreateFolder
- FolderExists
- ResolveSubFolder
- EnsureFolder
- EnsureFolderPath

10.1 How to create a folder on a library

In this section, you will learn how to create the folders on SharePoint library, using PnP Core CSOM library. CreateFolder method is used to create a folder on the document library.

The example given below creates a folder “Folder1”, under “Documents” library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - List Name.
 - Folder Name.
- The list is retrieved from the root level site object, using the list title.
- Subsequently, using the list object, the folders are created. The folder is created, using CreateFolder method with the new folder name as an input.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
```



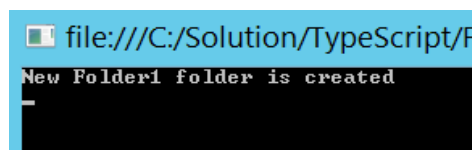
```
// Input Parameter
string listName = "TestList";
string folderName = "Folder1";
// Get Library
List list = clientContext.Site.RootWeb.GetListByTitle(listName);
// Create Folder
Folder folder = list.RootFolder.CreateFolder(folderName);

// Output
Console.WriteLine("New " + folder.Name + " folder is created");
Console.ReadKey();

}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

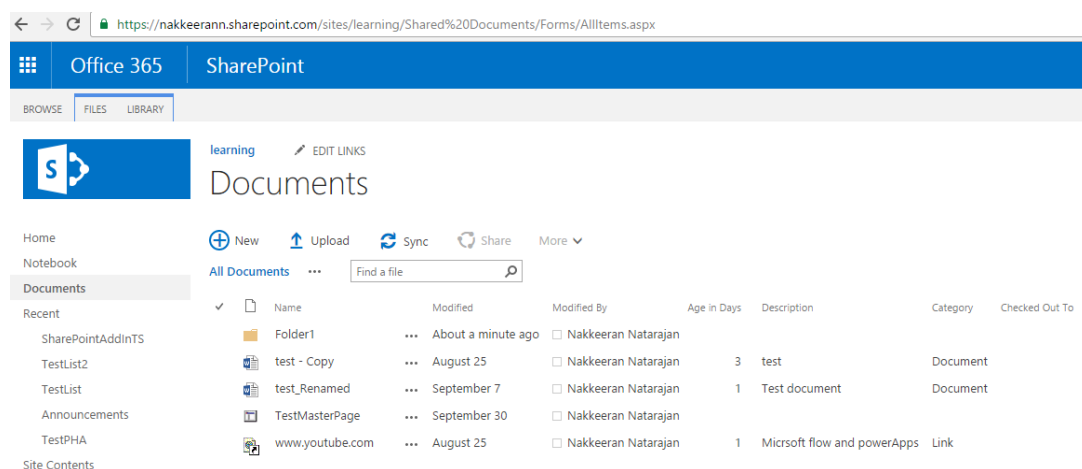
Result

The results will be displayed on the console, as shown below.



The created folder can be viewed on the library

(<https://nakkeerann.sharepoint.com/sites/learning/Shared%20Documents/Forms/AllItems.aspx>). The snapshot given below shows the folder created on the site library.



10.2 How to check if a folder exists on a library using folder name

In this section, you will learn how to check if a folder already exists on a SharePoint library using PnP Core CSOM library with folder name. FolderExists method is used to create a folder on the document library.

The example given below checks for a folder “Folder1”, under “Documents” library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - List name.
 - Folder name.
- The list is retrieved from the root level site object, using the list title.
- Subsequently, using the list object, the folder is checked. The folder is checked, using FolderExists method with the folder name as an input. The return type is Boolean.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string listName = "Documents";
        string folderName = "Folder1";
        // Get Library
        List list = clientContext.Site.RootWeb.GetListByTitle(listName);
        // Checks for folder
        bool folderExists = list.RootFolder.FolderExists(folderName);

        // Output
        if (folderExists)
        {
            Console.WriteLine("Folder already exists on the library");
        }
        else
        {

```

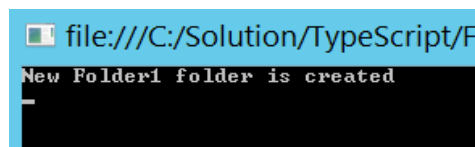
```

        Console.WriteLine("Folder doesn't exists on the library");
    }
    Console.ReadKey();
}
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
}

```

Result

The results will be displayed on the console, as shown below.



The folder can be viewed manually on the library

(<https://nakkeerann.sharepoint.com/sites/learning/Shared%20Documents/Forms/AllItems.aspx>).

10.3 How to check if a folder exists on a library using folder path

In this section, you will learn how to check, if a folder already exists on a SharePoint library , using PnP Core CSOM library with the folder Server relative path. DoesFolderExists method is used to check a folder on the document library with the help of the Web object.

The example given below checks for a folder “Folder1”, under “Documents” library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Folder Server relative URL.
- Using the Web object, the folder is checked. The folder is checked, using DoesFolderExists method with the folder URL as an input. The return type is Boolean.

- In the program file, insert the code give below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

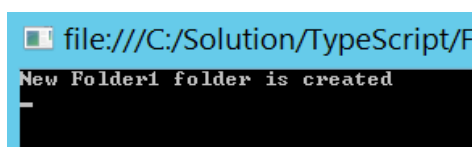
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string folderUrl = "/sites/learning/Shared Documents";

        // checks folder from site level
        bool folderExists = clientContext.Site.RootWeb.DoesFolderExists(folderUrl);

        // Output
        if (folderExists)
        {
            Console.WriteLine("Folder Exists");
        }
        else
        {
            Console.WriteLine("Folder doesn't exists");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



The folder can be viewed manually on the library

(<https://nakkeerann.sharepoint.com/sites/learning/Shared%20Documents/Forms/AllItems.aspx>).

10.4 How to retrieve the folder from the library

In this section, you will learn how to retrieve the required folder from SharePoint library , using PnP Core CSOM library. ResolveSubFolder method is used to retrieve the folder.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - List Name.
 - Folder Name.
- The list is retrieved from the root level site object, using the list title.
- Subsequently, using the list object, the folders are retrieved with the list and the folder names as an input.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
// PnP component to set context
AuthenticationManager authManager = new AuthenticationManager();
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string listName = "Documents";
        string folderName = "Folder1";
        // Get Library
        List list = clientContext.Site.RootWeb.GetListByTitle(listName);

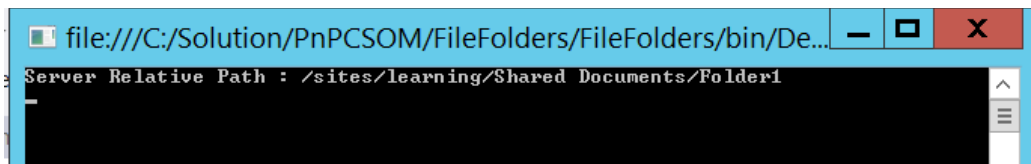
        // Retrieves Folder
        Folder folder = list.RootFolder.ResolveSubFolder(folderName);

        // Output
        Console.WriteLine("Server Relative Path : " + folder.ServerRelativeUrl);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
}
```

```
Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



10.5 How to check and create a folder on a library using folder name

In this section, you will learn how to check, if the folder exists and if it doesn't exist, the method will create the necessary folder, using the folder name. This can be done in a single method call on SharePoint library, using PnP Core CSOM library. EnsureFolder method is used to check and create a folder on the document library.

The example given below checks and creates a folder "Folder2", under "Documents" library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - List Name
 - Folder Name
- The list is retrieved from the root level site object, using the list title.
- Now, using the list object, the folders are checked and created. The folder is created , using EnsureFolder method with the new folder name as an input.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
```

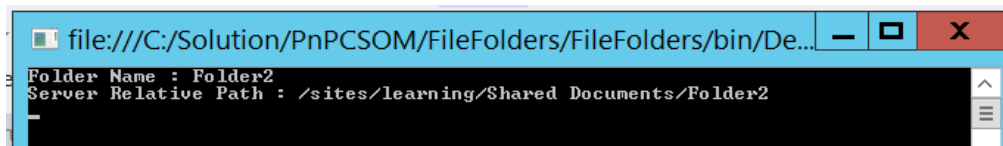
```
{
    // Input Parameter
    string listName = "Documents";
    string folderName = "Folder2";
    // Get Library
    List list = clientContext.Site.RootWeb.GetListByTitle(listName);

    // Retrieves and Creates Folder if not exists
    Folder folder = list.RootFolder.EnsureFolder(folderName);

    // Output
    Console.WriteLine("Folder Name : " + folder.Name);
    Console.WriteLine("Server Relative Path : " + folder.ServerRelativeUrl);
    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

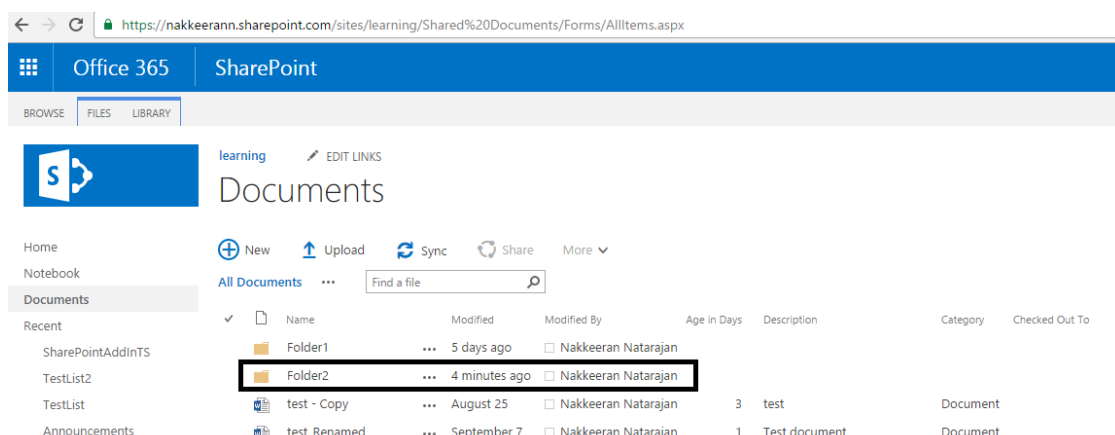
The results will be displayed on the console, as shown below.



```
file:///C:/Solution/PnPCSOM/FileFolders/FileFolders/bin/De...
Folder Name : Folder2
Server Relative Path : /sites/learning/Shared Documents/Folder2
```

The created folder can be viewed on the library

(<https://nakkeerann.sharepoint.com/sites/learning/Shared%20Documents/Forms/AllItems.aspx>). The snapshot given below shows the folder created on site library.



10.6 How to check and create a folder on a library using folder URL

In this section, you will learn how to check if folder exists and if it doesn't exist, the method will create the necessary folder with the help of the folder URL as an input. This can be done in a single method call on SharePoint library, using PnP Core CSOM library. EnsureFolderPath method is used to check and create a folder on the document library, using folder relative URL.

The following example checks for Folder3 and if it doesn't exist, creates a folder "Folder3", under "Documents" library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Folder Server relative URL
- Using the Web object, the folders are checked for the existence and are created. If the folder doesn't exist, it is created using EnsureFolderPath method.
- In the program file, insert the code, mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string folderUrl = "/Shared Documents/Folder3";

        // checks folder from site level
        Folder folder = clientContext.Site.RootWeb.EnsureFolderPath(folderUrl);

        // Output
        Console.WriteLine("Folder Name : " + folder.Name);
        Console.WriteLine("Server Relative Path : " + folder.ServerRelativeUrl);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
}
```



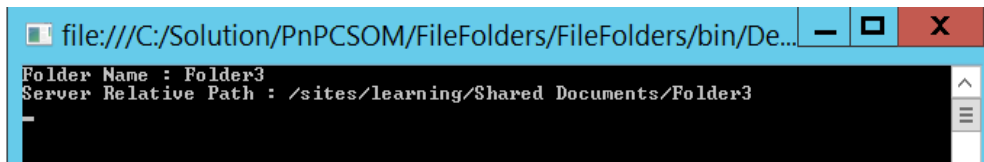
```

Console.WriteLine("Error Message: " + ex.Message);
Console.ReadKey();
}

```

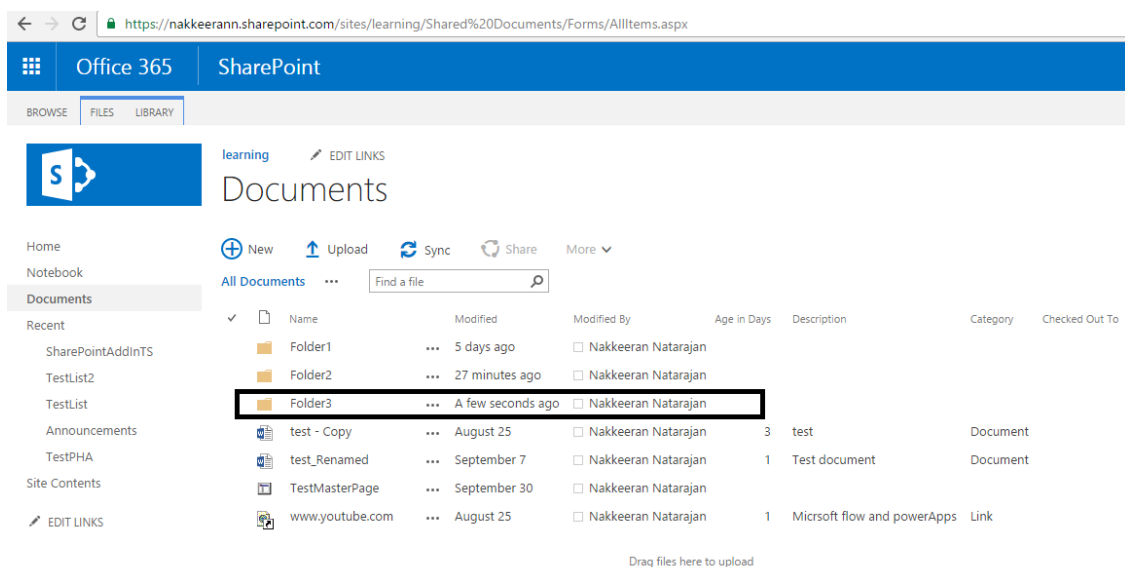
Result

The results will be displayed on the console, as shown below.



The created folder can be viewed on the library

(<https://nakkeerann.sharepoint.com/sites/learning/Shared%20Documents/Forms/AllItems.aspx>). The snapshot given below shows the created folder on the site library.



11 SharePoint Basic File tasks

In this topic, you will learn how to perform the basic file operations like add, find, retrieve, download or delete to/from the libraries, using PnP Core CSOM library.

The methods given below are used for the operations.

- UploadFile
- GetFile
- SaveFileToLocal
- CheckOutFile
- CheckInFile
- SetFileProperties

11.1 How to upload a file

In this section, you will learn how to upload a file to SharePoint library, using PnP Core CSOM library. UploadFile method is used to upload a file from the local system to SharePoint document library. Local file path, file name and library object are used to upload the documents.

The example given below adds a file to SharePoint library folder.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - List Name
 - Folder Name, where the file is uploaded
 - File Name on the site
 - Local file path
- The list is retrieved from the root level site object, using the list title.
- Subsequently, using the list object, the appropriate folder is accessed. With the help of the folder object, the file is uploaded to the library, using UploadFile method.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

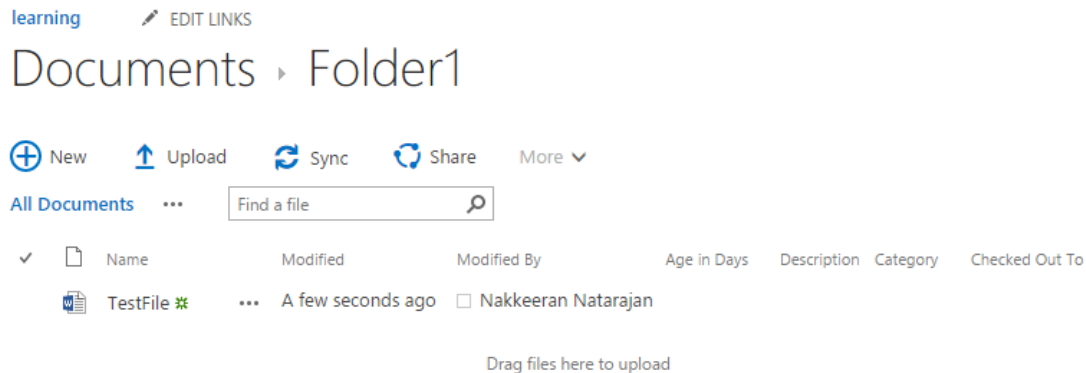
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string listName = "Documents";
        string folderName = "Folder1";
        string fileName = "TestFile.docx";
        string localPath = "C:/Solution/TestFile.docx";
        // Get Library
        List list = clientContext.Site.RootWeb.GetListByTitle(listName);
        // Retrieves Folder
        Folder folder = list.RootFolder.ResolveSubFolder(folderName);
        // Uploads File
        File newFile = folder.UploadFile(fileName, localPath, true);

        // Output
        Console.WriteLine(newFile.Title + " is uploaded to the site");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The file is successfully added to the folder under SharePoint library. The snapshot given below shows the library with the added file.



11.2 How to retrieve a file

In this section, you will learn how to retrieve a file from SharePoint library, using PnP Core CSOM library. GetFile method is used to retrieve the file, using the folder name and the file name.

The example given below shows retrieving file from a library or a folder.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - List Name
 - Folder Name, where file is uploaded
 - File Name on the site
- The list is retrieved from the root level site object, using the list title.
- Using the list object, the appropriate folder is accessed. With the help of the folder object, the file is retrieved from the library, using GetFile method.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
```

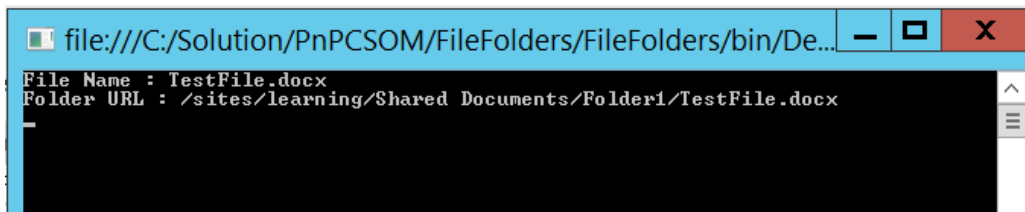
```
// Input Parameter
string listName = "Documents";
string folderName = "Folder1";
string fileName = "TestFile.docx";
// Get Library
List list = clientContext.Site.RootWeb.GetListByTitle(listName);
// Create Folder
Folder folder = list.RootFolder.ResolveSubFolder(folderName);
File file = folder.GetFile(fileName);

// Output
Console.WriteLine("File Name : " + file.Name);
Console.WriteLine("Folder URL : " + file.ServerRelativeUrl);
Console.ReadKey();

}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



11.3 How to download a file

In this section, you will learn how to download a file from SharePoint library, using PnP Core CSOM library. The Server file path, local file path and local file name are the required parameters for downloading a file.

The example given below downloads a file from SharePoint library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Server relative URL of the file
 - Local path
 - Local file name
- Using the Web object, download the file, using SaveFileToLocal method with the help of the input parameters, mentioned above.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {

        // Input Parameter
        string fileServerRelativeUrl = "/sites/learning/Shared Documents/Folder1/TestFile.docx";
        string localPath = "C:/Solution";
        string localFileName = "TestFileNew.docx";

        // Downloads file to local
        clientContext.Site.RootWeb.SaveFileToLocal(fileServerRelativeUrl, localPath, localFileName);

        // Output
        Console.WriteLine("File Downloaded");
        Console.ReadKey();
    }
}
```

```
}  
catch (Exception ex)  
{  
    Console.WriteLine("Error Message: " + ex.Message);  
    Console.ReadKey();  
}
```

Result

The results will be displayed on the console. The file is successfully downloaded. Similarly, the files can be downloaded from the library/subfolders on the library. The respective folder URL should be given.

11.4 How to checkout a file

In this section, you will learn how to checkout a file from SharePoint library, using PnP Core CSOM library. CheckOutFile method is used to checkout a file from SharePoint document library. Server relative URL of file is the required parameter to check out a file.

The example given below checks out a file from SharePoint library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Server relative URL of the file.
- Using the Web object, check out the file, using CheckOutFile method with the help of the input parameter given above.
- In the program file, insert the code given above and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try  
{  
    // Get and set the client context  
    // Connects to SharePoint online site using inputs provided  
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,  
        userName, password))  
    {  
        // Input Parameter
```

```
string fileServerRelativeUrl = "/sites/learning/Shared Documents/Folder1/TestFile.docx";

// Check Out file
clientContext.Site.RootWeb.CheckOutFile(fileServerRelativeUrl);

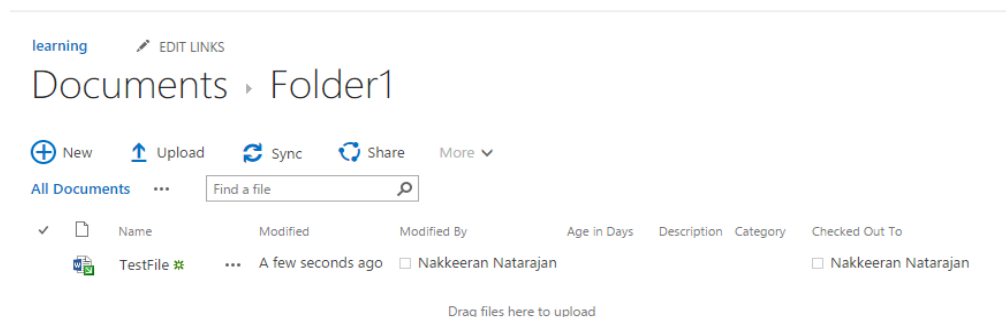
// Output
Console.WriteLine("File checked out");
Console.ReadKey();

}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The file is successfully checked out for editing on SharePoint library. Similarly, the files can be checked out from the folders/subfolders on the library.

The snapshot given below shows the file checked out from SP library.



11.5 How to checkin a file

In this section, you will learn how to checkin a file on SharePoint library, using PnP Core CSOM library. CheckInFile method is used to check in a file on SharePoint document library.

The example given below checks in a file on SharePoint library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- The input parameters should be set.
 - Server relative URL of the file.
 - Check-in type.
 - Comments.
- Using the Web object, check in the file, using CheckInFile method with the help of the input parameters stated above.
- In the program file, insert the following code and save the file.
- Run the code from Debug menu or by pressing F5.

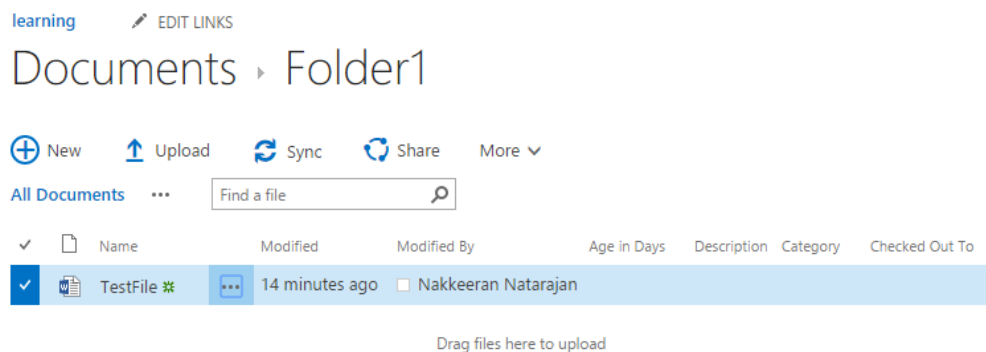
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string fileServerRelativeUrl = "/sites/learning/Shared Documents/Folder1/TestFile.docx";
        CheckinType checkinType = CheckinType.MajorCheckIn;
        string comment = "Checked in through PnP Programming";
        // Check In file
        clientContext.Site.RootWeb.CheckInFile(fileServerRelativeUrl, checkinType, comment);

        // Output
        Console.WriteLine("File checked in");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The file is successfully checked in on SharePoint library. Similarly, the files can be checked in to the folders/subfolders on the library. The snapshot given below shows the file checked in details on SP library. The version history of the file shows the recent check in information.



11.6 How to update the file properties

In this section, you will learn how to update the file properties on SharePoint library, using PnP Core CSOM library. SetFileProperties method is used to update the file properties

The example given below updates properties of a file on SharePoint library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the library with the library name and then retrieve the root folder or the sub folder. Retrieve the file, using GetFile method.
- The input parameters should be set.
 - File Properties as a dictionary variable
 - Checkout Boolean value, if required (optional)
- Using the file object, update the file properties, using the dictionary variable. Here, the dictionary variable contains the title and description values to be updated.
- In the program file, insert the code shown below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameter
        string listName = "Documents";
        string folderName = "Folder1";
        string fileName = "TestFile.docx";
        // Get Library
        List list = clientContext.Site.RootWeb.GetListByTitle(listName);
        // Create Folder
        Folder folder = list.RootFolder.ResolveSubFolder(folderName);
        File file = folder.GetFile(fileName);
        Dictionary<string, string> fileProperties = new Dictionary<string, string>();
        fileProperties.Add("Title", "Test File");
        fileProperties.Add("Description0", "Properties updated using PnP Programming");
        file.SetFileProperties(fileProperties);
        // Output
        Console.WriteLine("File properties updated");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The file is successfully updated with title and description properties. The snapshot given below shows the updated properties.

Name *	<input type="text" value="TestFile"/> .docx
Title	<input type="text" value="Test File"/>
Age in Days	<input type="text"/>
Description	<div>Properties updated using PnP Programming</div>
Category	<input type="text" value="▼"/>

12 SharePoint Page Tasks

In this topic, you will learn how to perform basic page operations on SharePoint site, using PnP Core CSOM library.

The methods used for the page operations are shown below.

- AddPublishingPage
- GetPublishingPage
- AddWikiPage
- AddWikiPageByUrl
- GetWikiPageContent
- EnsureWikiPage
- AddHtmlToWikiPage
- AddLayoutToWikiPage
- GetWebParts
- AddWebPartToWebPartPage
- DeleteWebPart
- GetWebPartProperties
- SetWebPartProperties
- AddWebPartToWikiPage

12.1 How to create a publishing page

In this section, you will learn how to create a publishing page on SharePoint site, using PnP Core CSOM library.

SharePoint publishing features should be activated at both site and Web scopes, prior to running this operation.

The example given below creates a page on SharePoint pages library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, create the page, using AddPublishingPage method. The required input parameters are given below.
 - Page Name

- Page Template Name
- Title for page
- In the program file, insert the code, mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

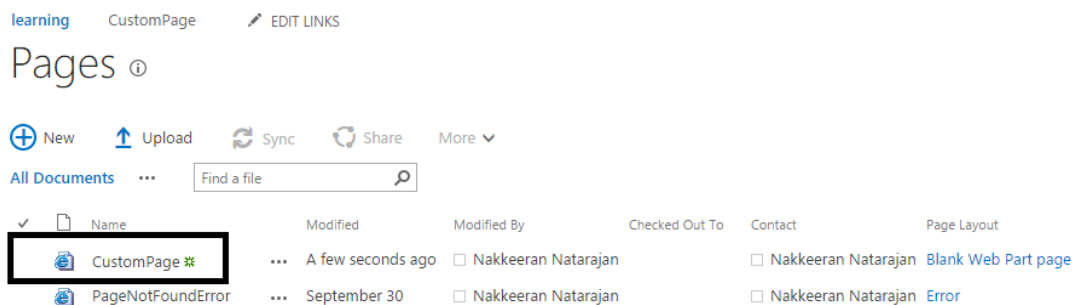
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageName = "CustomPage";
        string pageTemplate = "BlankWebPartPage";
        string pageTitle = "CustomPage";

        // Adds Publishing Page
        clientContext.Site.RootWeb.AddPublishingPage(pageName, pageTemplate, pageTitle);

        // Output
        Console.WriteLine("Publishing Page Created");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The file is successfully created on the pages library. The snapshot given below shows the pages library with the created page.



12.2 How to retrieve a publishing page

In this section, you will learn how to retrieve a page from pages library, using PnP Core CSOM library.

The example given below retrieves the pages from pages library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the page, using GetPublishingPage method. The required input parameter is given below.
 - Page Leaf URL.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string fileUrl = "CustomPage1.aspx";

        // Gets Publishing Page
        PublishingPage page = clientContext.Site.RootWeb.GetPublishingPage(fileUrl);

        // Output
        if (page != null)
        {
            // Process the Page
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.

Note

The publishing pages library should be available before executing the operation shown above.

12.3 How to create a wiki page using page name

In this section, you will learn how to create a wiki page on SharePoint site using PnP Core CSOM library. AddWikiPage method is used to create a wiki page, using wiki page name and library name.

The example given below creates a wiki page on the site, using the page content.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, create the page, using the input parameters,
 - Wiki Page Name
 - Wiki Pages Library Name
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string libraryName = "WikiPages";
        string pageName = "CustomPage";

        // Adds Publishing Page
        string wikiPage = clientContext.Site.RootWeb.AddWikiPage(libraryName, pageName);

        // Output
        Console.WriteLine("Wiki Page Created. Page URL : " + wikiPage);
        Console.ReadKey();
    }
}
```



```
    }  
}  
catch (Exception ex)  
{  
    Console.WriteLine("Error Message: " + ex.Message);  
    Console.ReadKey();  
}
```

Result

The results will be displayed on the console. The page is successfully created on SharePoint site wiki page's library.

Note

The wiki pages library should be created before executing the operation shown above.

12.4 How to create a wiki page using URL and the file content

In this section, you will learn how to create a wiki page on SharePoint site, using the page URL with the page content by using PnP Core CSOM library. AddWikiPageByUrl method is used to create a wiki page on the site. Page name and page content are passed as the parameters to create a page.

The example given below creates a wiki page on the site, using page content.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, create the page, using the input parameters.
 - Wiki page URL.
 - Wiki page content.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

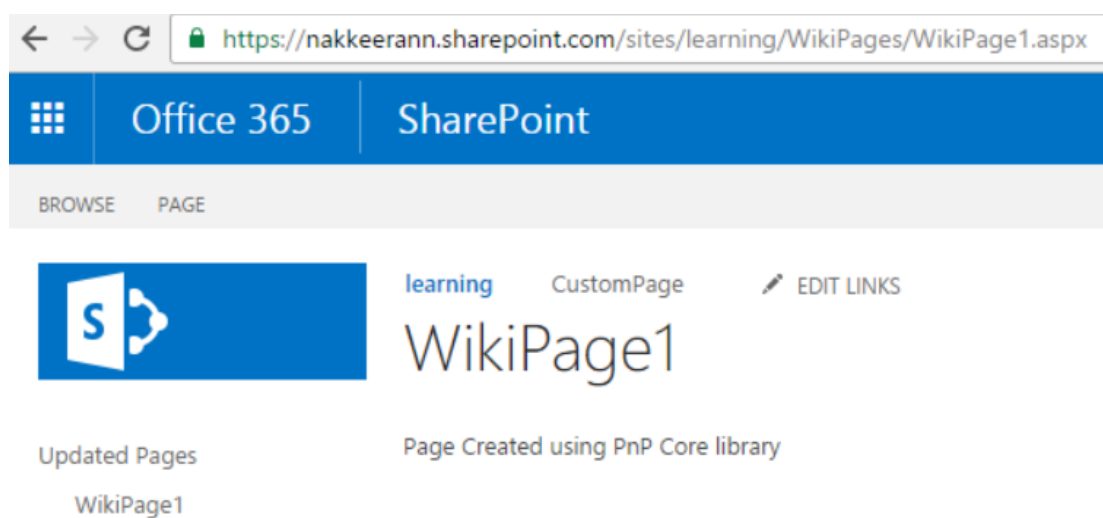
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageContent = "Page Created using PnP Core library";
        string pageUrl = "/sites/learning/WikiPages/WikiPage1.aspx";

        // Adds Wiki Page using Url with content
        clientContext.Site.RootWeb.AddWikiPageByUrl(pageUrl, pageContent);

        // Output
        Console.WriteLine("Wiki Page Created");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The page is successfully created on SharePoint site wiki page's library. The snapshot given below shows the page with the page content.



Note

The wiki pages library should be created before executing the operation shown above.

12.5 How to retrieve wiki page content

In this section, you will learn how to retrieve wiki page content, using PnP Core CSOM library. GetWikiPageContent method is used to retrieve the page content.

The example given below retrieves the pages from the pages library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the page content, using wiki page Server relative URL. The required input parameter is given below.
 - Wiki Page URL
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

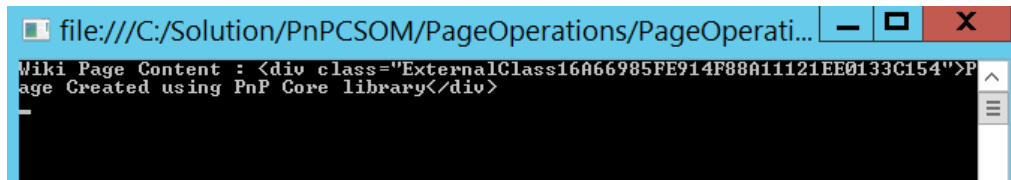
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageUrl = "/sites/learning/WikiPages/WikiPage1.aspx";

        // Gets Wiki Page using Url
        string pageContent = clientContext.Site.RootWeb.GetWikiPageContent(pageUrl);

        // Output
        Console.WriteLine("Wiki Page Content : " + pageContent);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The snapshot given below shows the page content retrieved.



Note

The wiki pages library should be available before executing the operation given below.

12.6 How to ensure a wiki page

In this section, you will learn how to check, if wiki page exists or not. If it's not present, the same method will create a wiki page on the specified library. This can be accomplished, using single method by PnP Core CSOM library. EnsureWikiPage method is used for the operation.

The example given below checks and creates a wiki page on the site.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, create the page, using the input parameters.
 - Wiki page library name
 - Wiki page name
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
```

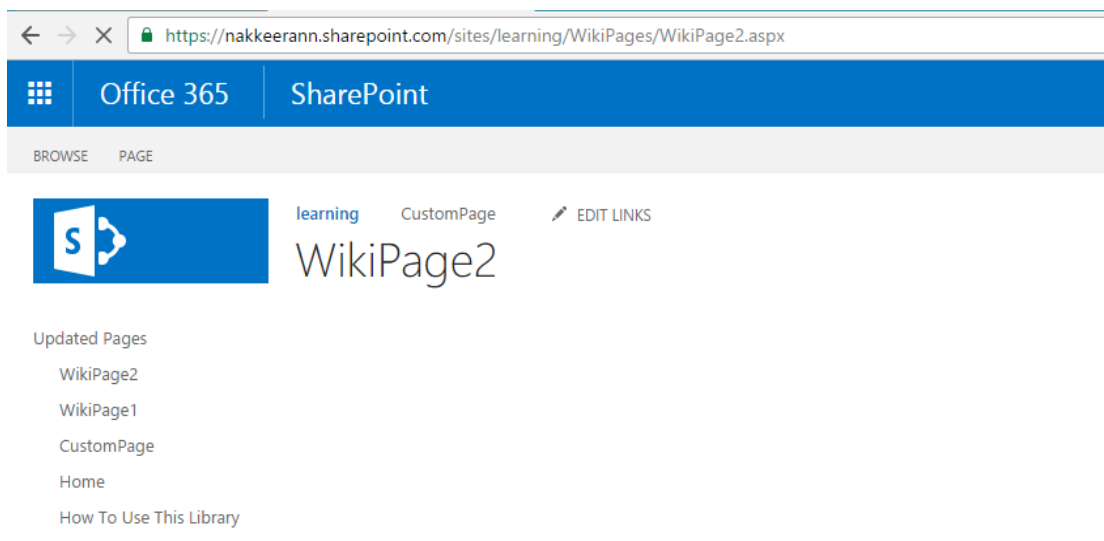
```
// Input Parameters
string libraryName = "WikiPages";
string pageName = "WikiPage2.aspx";

// Checks and adds Wiki Page
string wikiPage = clientContext.Site.RootWeb.EnsureWikiPage(libraryName, pageName);

// Output
Console.WriteLine("Wiki Page is Availalbe");
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The page is successfully ensured on SharePoint site wiki page's library. The snapshot given below shows the wiki page.



Note

The wiki pages library should be created before executing the operation given below.

12.7 How to update a wiki page content

In this section, you will learn how to update the page content of wiki page, using PnP Core CSOM library. AddHtmlToWikiPage method is used for updating the wiki page content. The wiki page URL and content are passed as the parameters for updating.

The example given below retrieves the page content of wiki page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, update the page content, using wiki page Server relative URL. The required input parameter is given below.
 - Wiki page URL.
 - HTML content to be updated.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

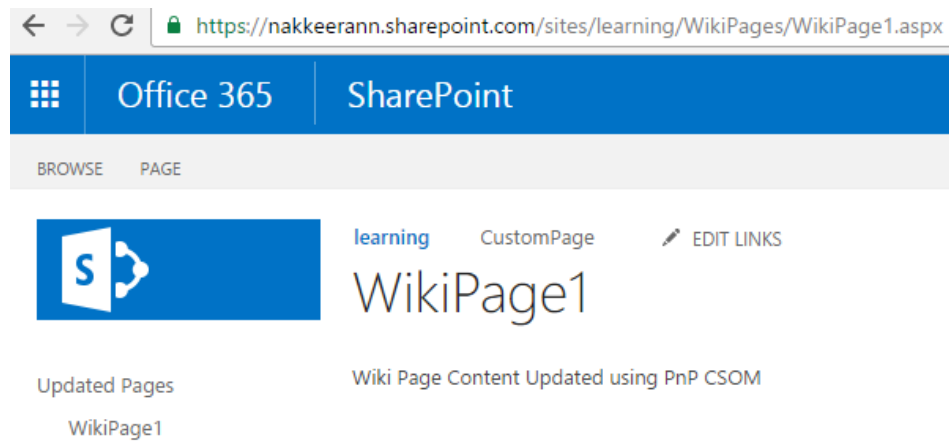
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageContent = "Wiki Page Content Updated using PnP CSOM";
        string pageUrl = "/sites/learning/WikiPages/WikiPage1.aspx";

        // Adds content to Wiki Page
        clientContext.Site.RootWeb.AddHtmlToWikiPage(pageUrl, pageContent);

        // Output
        Console.WriteLine("Wiki Page Content Updated");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The snapshot given below shows the page content updated.



12.8 How to add layout to a wiki page

In this section, you will learn how to apply existing page layout to a wiki page, using PnP Core CSOM library. `AddLayoutToWikiPage` method is used to update wiki page layout.

The example given below applies the page layout to wiki page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, apply the page layout. The page layouts can be accessed, using `WikiPageLayout` object. The required input parameters are given below.
 - Wiki Page URL
 - Page Layout
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

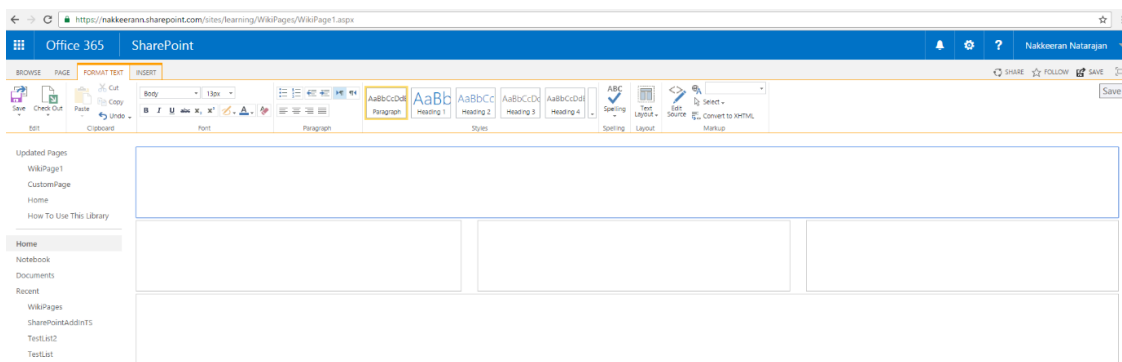
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageUrl = "/sites/learning/WikiPages/WikiPage1.aspx";
        WikiPageLayout pageLayout = WikiPageLayout.ThreeColumnsHeaderFooter;

        // Adds layout to Wiki Page
        clientContext.Site.RootWeb.AddLayoutToWikiPage(pageLayout, pageUrl);

        // Output
        Console.WriteLine("Wiki Page Content Updated");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The snapshot given below shows the page layout is applied.



12.9 How to retrieve web parts from a page

In this section, you will learn how to retrieve the Web parts from a page, using PnP Core CSOM library. GetWebParts method is used to retrieve the Webparts from a specified page.

The example given below applies the page layout to wiki page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the Web parts from a page, using the page relative URL. The required parameter is given below.
 - Page URL
- In the program file, insert the code mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageUrl = "/sites/learning/Pages/CustomPage.aspx";

        // Retrieves web parts
        var webparts = clientContext.Site.RootWeb.GetWebParts(pageUrl);
        // Output
        foreach (WebPartDefinition webpart in webparts)
        {
            Console.WriteLine(webpart.WebPart.Title);
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on console.

12.10 How to add the Web part to a publishing page

In this section, you will learn how to add a Web part to a publishing page, using PnP Core CSOM library. `AddWebPartToWebPartPage` method is used to adding Webpart to a specified page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, add the Web part to a page, using page relative URL. The required parameter is
 - Page URL
 - Web part XML
- The Web part XML defines the Web part properties, which needs to be added to the page. The sample given below shows the Web part XML of the content editor Web part, which will be used for adding it to the page.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Web Part XML

```
<?xml version="1.0" encoding="utf-8"?>
<WebPart xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/WebPart/v2">
  <Title>Custom Content Editor</Title>
  <FrameType>Default</FrameType>
  <Description>Allows authors to enter rich text content.</Description>
  <IsIncluded>true</IsIncluded>
  <ZoneID>Header</ZoneID>
  <PartOrder>0</PartOrder>
  <FrameState>Normal</FrameState>
  <Height />
  <Width />
  <AllowRemove>true</AllowRemove>
  <AllowZoneChange>true</AllowZoneChange>
  <AllowMinimize>true</AllowMinimize>
  <AllowConnect>true</AllowConnect>
  <AllowEdit>true</AllowEdit>
  <AllowHide>true</AllowHide>
  <IsVisible>true</IsVisible>
  <DetailLink />
  <HelpLink />
  <HelpMode>Modeless</HelpMode>
  <Dir>Default</Dir>
  <PartImageSmall />
  <MissingAssembly>Cannot import this Web Part.</MissingAssembly>
  <PartImageLarge>/_layouts/15/images/mscontl.gif</PartImageLarge>
  <IsIncludedFilter />
  <Assembly>Microsoft.SharePoint, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c</Assembly>
  <TypeName>Microsoft.SharePoint.WebPartPages.ContentEditorWebPart</TypeName>
  <ContentLink xmlns="http://schemas.microsoft.com/WebPart/v2/ContentEditor" />
  <Content xmlns="http://schemas.microsoft.com/WebPart/v2/ContentEditor" />
  <PartStorage xmlns="http://schemas.microsoft.com/WebPart/v2/ContentEditor" />
</WebPart>
```

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // Input Parameters
        string webPartXml = @"C:\Solution\PnPCSOM\PageOperations\webpart.xml";
        string pageUrl = "/sites/learning/Pages/CustomPage.aspx";
```

```
WebPartEntity webpart = new WebPartEntity();
webpart.WebPartXml = System.IO.File.ReadAllText(webPartXml);
webpart.WebPartZone = "Right";
// Adds web part
clientContext.Site.RootWeb.AddWebPartToWebPartPage(pageUrl, webpart);

Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The page will be updated with the Web part.

12.11 How to delete the Web part from a page

In this section, you will learn how to remove a Web part from the page, using PnP Core CSOM library. DeleteWebPart method is used to remove the Webpart from a specified page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, remove the Web part from a page, using page relative URL. The required parameters are given below.
 - Page URL.
 - Web part title.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
```

```
{
    // Input Parameters
    string pageUrl = "/sites/learning/Pages/CustomPage.aspx";
    string webpartTitle = "Custom Content Editor";

    // Adds web part
    clientContext.Site.RootWeb.DeleteWebPart(pageUrl, webpartTitle);

    Console.WriteLine("Web part is deleted from page");
    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The Web part will be removed successfully from the page.

12.12 How to retrieve web part properties

In this section, you will learn how to retrieve the properties of a Web part, using PnP Core CSOM library. GetWebPartProperties method is used for retrieving the properties of a Web part from a specified page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the properties, using the input parameters given below.
 - Web part GUID
 - Page URL
- In the program file, insert the code, mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

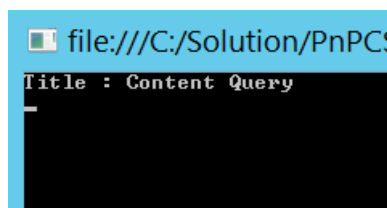
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageUrl = "/sites/learning/Pages/CustomPage.aspx";
        Guid webpartId = new Guid("97825521-36a8-4302-a0d5-7b4911dfbc0c");
        // Retrieves web part properties
        PropertyValues webpartProps = clientContext.Site.RootWeb.GetWebPartProperties(webpartId,
            pageUrl);

        // Output
        Console.WriteLine("Title : "+webpartProps["Title"]);
        // Similarly other properties can also be accessed
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.



Note

In the above sample, title of a web part is accessed. Similarly, other required properties can be accessed.

12.13 How to update the Web part properties

In this section, you will learn how to update the properties of a Web part, using PnP Core CSOM library. SetWebPartProperties method is used to update the properties of a Web part from a specified page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the properties, using the input parameters given below.
 - Web part GUID
 - Page URL
 - Web part property key
 - Web part property value
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string pageUrl = "/sites/learning/Pages/CustomPage.aspx";
        Guid webpartId = new Guid("97825521-36a8-4302-a0d5-7b4911dfbc0c");
        string propertyName = "Description";
        string propertyValue = "Content query web part used to query the SP lists";

        // Update web part properties
        clientContext.Site.RootWeb.SetWebPartProperty(propertyName,propertyValue, webpartId,
            pageUrl);
        // Similarly other properties can also be updated

        // Output
        Console.WriteLine("Web Part Properties updated, Similarly other properties can be updated.");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
}
```

```
Console.ReadKey();  
}
```

Result

The results will be displayed on console.

Note

1. To check the updated values, export the Web part from a page manually and check the Web part description file.
2. In the sample given above, description of a Web part is updated. Similarly, other required properties can be updated.

12.14 How to add the web part to a wiki page

In this section, you will learn how to add a Web part to a wiki page, using PnP Core CSOM library. AddWebPartToWikiPage method is used to add the Web part to a specified page.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, add the Web part to a page, using page relative URL. The required parameters are given below.
 - Page URL
 - Web part XML
 - Row – Page table row value
 - Column – Page table column value
 - Space (Boolean) – Blank line after the Web part in the page
- The Web part XML defines the Web part properties, which needs to be added to the page. Please refer to [section 12.10 \(How to add web part to a publishing page\)](#) section for the Web part sample XML content.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string webPartXml = @"C:\Solution\PnPCSOM\PageOperations\webpart.xml";
        string pageUrl = "/sites/learning/WikiPages/WikiPage1.aspx";

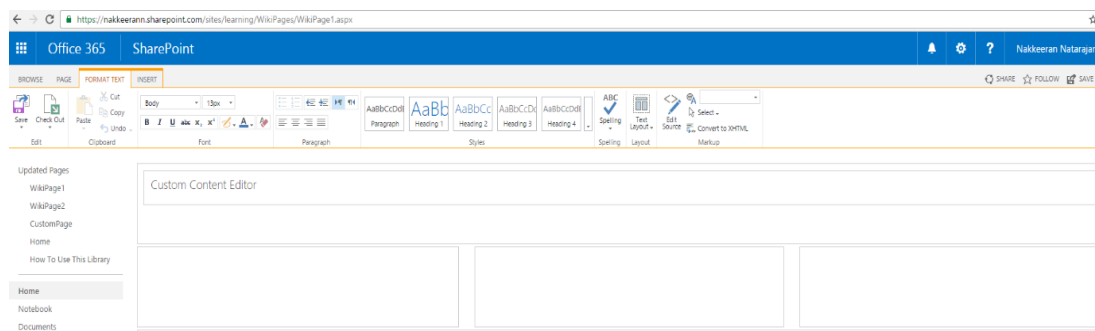
        WebPartEntity webpart = new WebPartEntity();
        webpart.WebPartXml = System.IO.File.ReadAllText(webPartXml);
        webpart.WebPartZone = "Right";
        int webpartRow = 1; // Wiki Table Row
        int webpartColumn = 1; // Wiki Table Column
        bool space = true; // Blank Line after the web part

        // Adds web part
        clientContext.Site.RootWeb.AddWebPartToWikiPage(pageUrl, webpart, webpartRow,
            webpartColumn, space);

        // Output
        Console.WriteLine("Web part is successfully added to Wiki Page");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The page will be updated with the Web part.



13 SharePoint User and Group Operations

In this topic, you will learn how to perform basic SharePoint user and group that can be performed on SharePoint site, using PnP Core CSOM library.

The methods used for the user and group operations are given below.

- GetAdministrators
- AddAdministrators
- AddGroup
- GetGroupID
- GroupExists
- AddUserToGroup
- AddPermissionLevelToGroup
- RemovePermissionLevelFromGroup
- AddPermissionLevelToUser
- AddPermissionLevelToPrincipal
- RemovePermissionLevelFromUser
- RemovePermissionLevelFromPrincipal
- AddReaderAccess

13.1 How to retrieve site collection administrator

In this section, you will learn how to retrieve the site collection administrators from a SharePoint site, using PnP Core CSOM library. GetAdministrators method is used to retrieve all the site collection administrators of a site.

The example given below retrieves the administrators.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the administrators, using GetAdministrators method. The input parameters are not required for this process.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

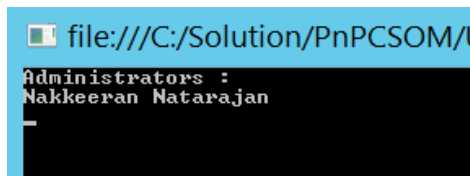
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Gets Site Collection Admins
        List<UserEntity> admins = clientContext.Site.RootWeb.GetAdministrators();

        foreach (UserEntity admin in admins)
        {
            Console.WriteLine("Admin Name : " + admin.Title);
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.



Note

The above operation is applicable only for SharePoint online sites.

13.2 How to add the site collection administrator

In this section, you will learn how to add a user as a site collection administrator to a SharePoint site, using PnP Core CSOM library. AddAdministrators method is used to add the site collection administrators to a site.

The example given below adds the administrators.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book for importing the packages and creating authentication manager object.
- Using the web object, add the administrators, using AddAdministrators method. The input parameters are given below.
 - User Entity List – List with the set of users
 - Add to owners group (Boolean) – The users to be added to the site group owners.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        List<UserEntity> admins = new List<UserEntity>();
        UserEntity admin = new UserEntity();
        admin.LoginName = "nirmal";
        admins.Add(admin);
        // Multiple users can be added as admins with the help of above list

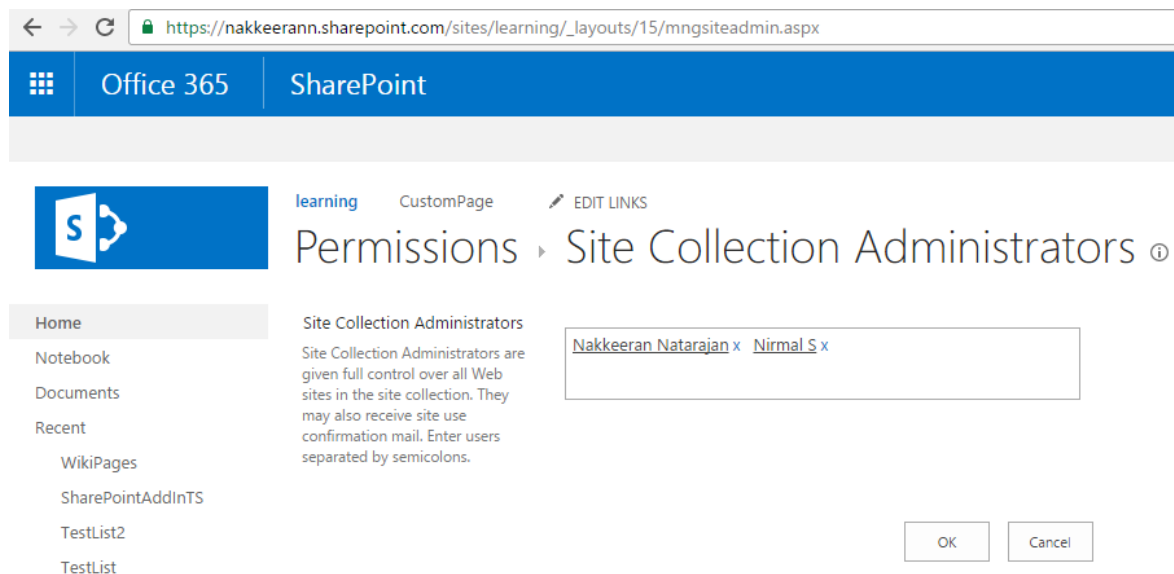
        // Adds Site Collection Admins
        clientContext.Site.RootWeb.AddAdministrators(admins,true);

        Console.WriteLine("Users added as Site Collection Admins");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
}
```

```
Console.ReadKey();
}
```

Result

The results will be displayed on the console. Navigate to the site collection administrators' page to check the output. The snapshot given below shows the site collection administrators' page.



Note

The operation given above is applicable only for SharePoint online sites.

13.3 How to create a SharePoint group

In this section, you will learn how to create a SharePoint group on SharePoint site, using PnP Core CSOM library. AddGroup method is used to create a group on the site. Group name and group description are required to create a group.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, create/add the group, using AddGroup method. The input parameters are given below.
 - Group Name
 - Group Description

- Site Ownership for group
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

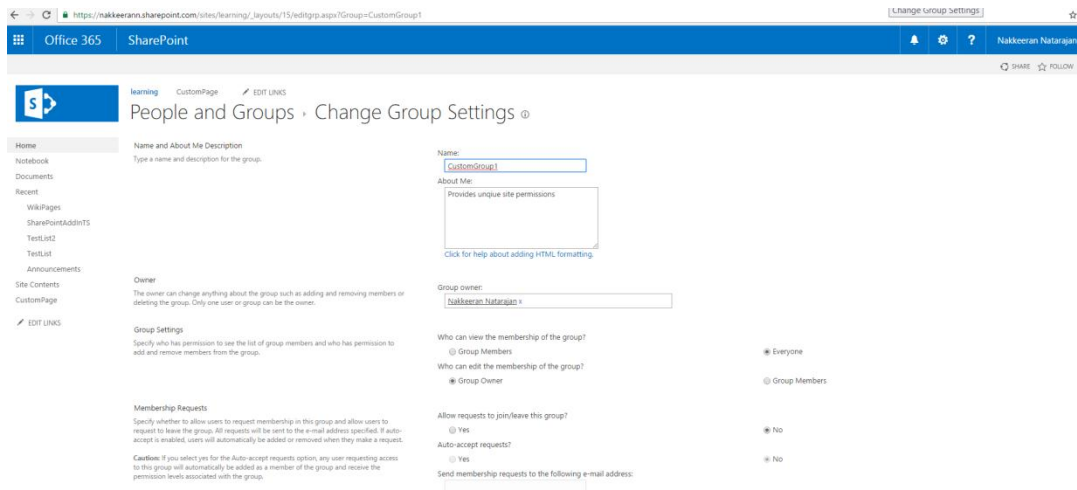
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        string groupName = "CustomGroup1";
        string groupDesc = "Provides unique site permissions";
        bool siteOwnership = false;
        // Adds group to site
        Group customGroup = clientContext.Site.RootWeb.AddGroup(groupName, groupDesc,
            siteOwnership);
        // Other properties / Users can be added to group directly
        Console.WriteLine("Group is created and added to site");

        Console.WriteLine("Title : " + customGroup.Title);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The snapshot given below shows the SharePoint group created on SharePoint portal.



13.4 How to retrieve a SharePoint group ID

In this section, you will learn how to retrieve ID of a SharePoint group from SharePoint site , using PnP Core CSOM library. GetGroupID method is used to retrieve the ID of a group.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, retrieve the Id, using GetGroupID method. The input parameter is given below.
 - Group Name.
- In the program file, insert the code, mentioned below and save the file.
- Run the code from Debug menu or by pressing F5.

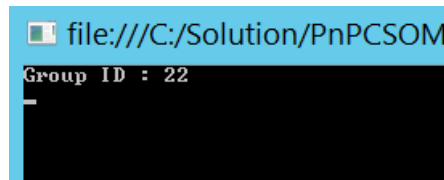
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string groupName = "CustomGroup1";
        // Retrieves group ID using group name
        int groupId = clientContext.Site.RootWeb.GetGroupID(groupName);
    }
}
```

```
//Output
Console.WriteLine("Group ID : " + groupId);
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.



13.5 How to check if a SharePoint group exists on Site or not

In this section, you will learn how to check, if the group already exists on the site or not, using PnP Core CSOM library. GroupExists method is used for this operation.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, check if the group exists, using GroupExists method. The input parameter is given below.
 - Group Name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
```



```
string groupName = "CustomGroup1";
// Retrieves group ID using group name
bool groupExists = clientContext.Site.RootWeb.GroupExists(groupName);

//Output
if (groupExists)
{
    Console.WriteLine("Group exists");
}
else
{
    Console.WriteLine("Group doesnt exists");
}
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.



13.6 How to add user to a SharePoint group

In this section, you will learn how to add a user to SharePoint group, using PnP Core CSOM library. AddUserToGroup method is used for this operation.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, add the user to group, using AddUserToGroup method. The input parameters are given below.
 - Group Name
 - User Login Name

- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

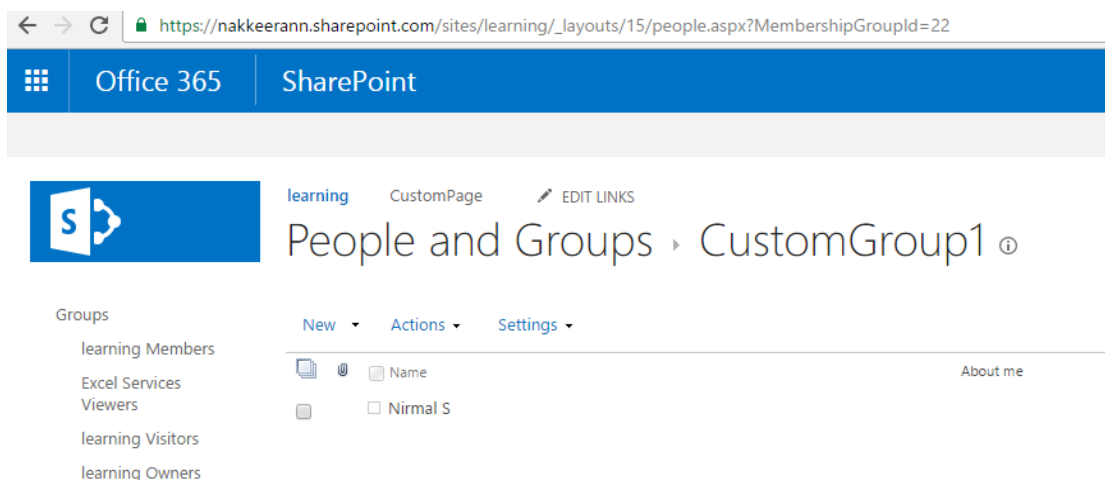
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string groupName = "CustomGroup1";
        string userLoginName = "Nirmal";
        // Adds user to group
        clientContext.Site.RootWeb.AddUserToGroup(groupName, userLoginName);

        // Output
        Console.WriteLine("Added user to group");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The snapshot given below shows the group with the added users.



13.7 Add permissions to SharePoint group / add groups with permissions to the site (or list or item)

In this section, you will learn how to add the permissions to SharePoint group, using PnP Core CSOM library. AddPermissionLevelToGroup method is used for updating the permissions. Adding permissions to the group will literally add the group to the site permissions.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, add the permissions to the group, using AddPermissionLevelToGroup method. The input parameters are given below.
 - Group Name.
 - Role Type.
 - Remove Existing Permissions (Boolean).
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

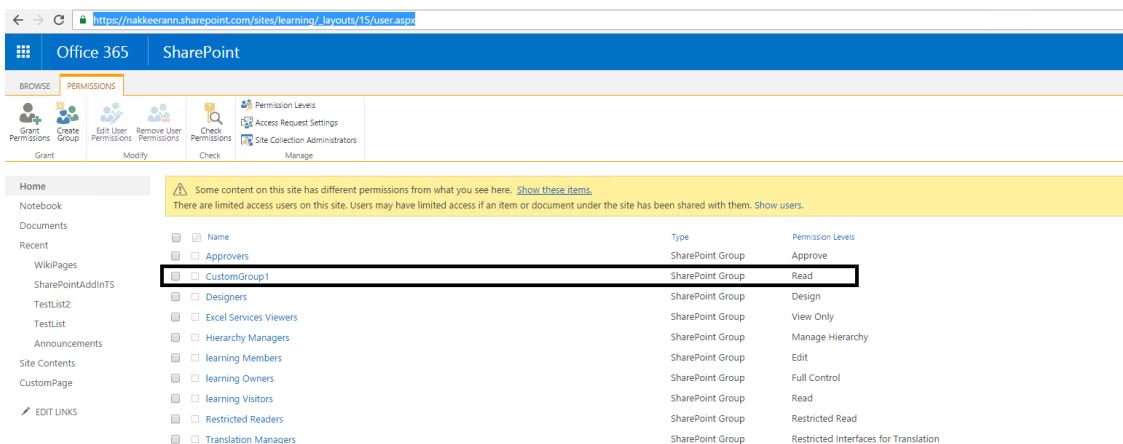
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string groupName = "CustomGroup1";
        RoleType roleType = RoleType.Reader;
        bool removeExistingPermission = true;
        // Updates group permissions
        clientContext.Site.RootWeb.AddPermissionLevelToGroup(groupName, roleType,
            removeExistingPermission);

        // Output
        Console.WriteLine("Group permissions updated");
        Console.ReadKey();
    }
}
catch (Exception ex)
```

```
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

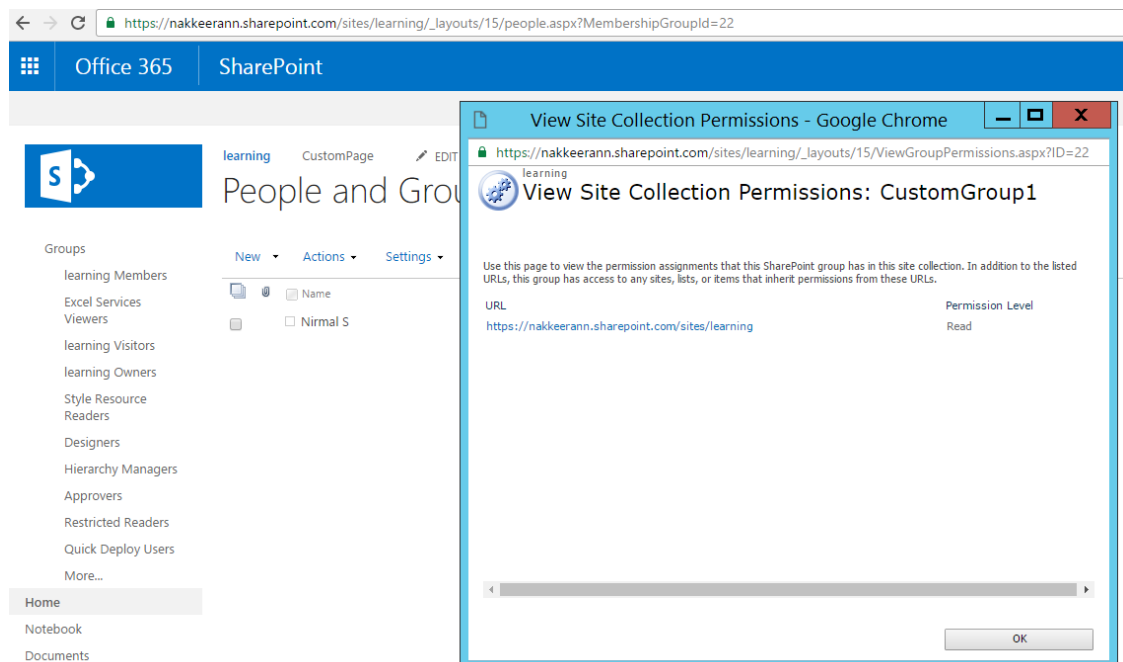
The results will be displayed on the console. To check the permission manually, navigate to Site Settings → Site Permissions. The snapshot given below shows the group with the added permissions.



The screenshot shows the SharePoint 'PERMISSIONS' page for a site. A table lists the permissions assigned to various groups. The 'CustomGroup1' group is highlighted, showing it has 'Read' permission.

Name	Type	Permission Levels
Approvers	SharePoint Group	Approve
CustomGroup1	SharePoint Group	Read
Designers	SharePoint Group	Design
Excel Services Viewers	SharePoint Group	View Only
Hierarchy Managers	SharePoint Group	Manage Hierarchy
Learning Members	SharePoint Group	Edit
Learning Owners	SharePoint Group	Full Control
Learning Visitors	SharePoint Group	Read
Restricted Readers	SharePoint Group	Restricted Read
Translation Managers	SharePoint Group	Restricted Interfaces for Translation

The permissions of a group can be accessed from the respective group page by navigating to Site Settings → People and Group → Groups → Respective Group → Settings → Group Permissions. The snapshot given below depicts the same.



The screenshot shows the 'People and Groups' page in SharePoint. A dialog box titled 'View Site Collection Permissions - Google Chrome' is open, displaying the permissions for 'CustomGroup1'. The dialog shows a table with the URL 'https://nakkeerann.sharepoint.com/sites/learning' and the permission level 'Read'.

URL	Permission Level
https://nakkeerann.sharepoint.com/sites/learning	Read

Note

1. In the sample given above, the group permissions is with respect to the sites. If the method `AddPermissionLevelToGroup` is accessed from the lists/permissions instead from the Web object, the permissions will be updated for the respective objects.
2. If you are going to perform the operation with the group, object exists instead of the group name, then the same operation can be carried out, using `AddPermissionLevelToPrincipal` method. The code snippet given below shows the logic.

```
clientContext.Site.RootWeb.AddPermissionLevelToPrincipal(group, roleType,
removeExistingPermission);
```

13.8 Removes permissions from SharePoint group / Removes group with permissions from site (or list or item)

In this section, you will learn how to remove the permissions from SharePoint group, using PnP Core CSOM library. `RemovePermissionLevelFromGroup` method is used to remove the permissions. Removing the permissions from the group will literally remove the group from the site permissions, if no other permissions exists.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, remove the permissions from the group, using `RemovePermissionLevelFromGroup` method. The input parameters are given below.
 - Group Name
 - Role Type
 - Remove All Permissions (Boolean)
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // Input Parameters
```

```
string groupName = "CustomGroup1";
RoleType roleType = RoleType.Reader;
bool removeAllPermissions = true;
// Removes group permissions
clientContext.Site.RootWeb.RemovePermissionLevelFromGroup(groupName, roleType,
removeAllPermissions);

// Output
Console.WriteLine("Group permissions removed");
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. To check the permission manually, navigate to Site Settings → Site Permissions.

The permissions of a group can be accessed from the respective group page by navigating to Site Settings → People and Group → Groups → Respective Group → Settings → Group Permissions.

Note

1. In the sample given above, the group permissions are with respect to the site. If the method `RemovePermissionLevelFromGroup` is accessed from the lists/permissions instead from the Web object, the permissions will be removed for the respective objects.
2. If the group object exists, the same operation can be carried out, using `RemovePermissionLevelFromPrincipal` method. The code snippet given below shows the logic.

```
clientContext.Site.RootWeb.RemovePermissionLevelFromPrincipal(group, roleType,
removeExistingPermission);
```

13.9 Add the permissions to SharePoint User / add the user with permissions to the site (or list or item)

In this section, you will learn how to add the permissions to SharePoint user, using PnP Core CSOM library. `AddPermissionLevelToUser` method is used to update the permissions. Adding permissions to the user will literally add the user to the site permissions.

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

Note- In the sample given below, the permissions are respect to the site/Web. If the method AddPermissionLevelToUser is accessed from the lists/permissions instead of the Web object, the permissions will be updated for the respective objects.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, add the user to group, using AddPermissionLevelToUser method. The input parameters are given below.
 - User Login Name
 - Role Type
 - Remove Existing Permissions (Boolean)
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

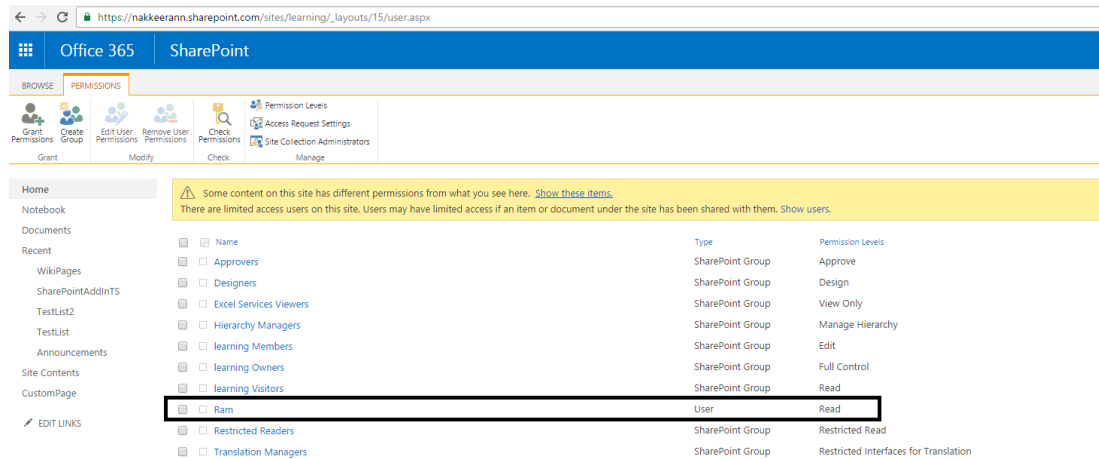
Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string userLoginName = "ram";
        RoleType roleType = RoleType.Reader;
        bool removeExistingPermission = true;
        // Updates user permissions
        clientContext.Site.RootWeb.AddPermissionLevelToUser(userLoginName, roleType,
            removeExistingPermission);

        // Output
        Console.WriteLine("Added user to site with read permission");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. To check the permission manually, navigate to Site Settings → Site Permissions. The snapshot given below shows the group with the added permissions.



Note

If the user object exists, then the same operation can be carried out, using AddPermissionLevelToPrincipal method. The code snippet given below shows the logic.

```
clientContext.Site.RootWeb.AddPermissionLevelToPrincipal(user, roleType,
removeExistingPermission);
```

13.10 Remove the permissions for SharePoint user / remove the user with the permissions from the site (or list or item)

In this section, you will learn how to remove the permissions for SharePoint user, using PnP Core CSOM library. RemovePermissionLevelFromUser method is used to update the permissions. Adding permissions to the user will literally remove the user from the site permissions, if no other permissions exists for the user.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, add the user to the group, using RemovePermissionLevelFromUser method. The input parameters are given below.
 - User Login Name
 - Role Type

- Remove All Permissions (Boolean)
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string userLoginName = "ram";
        RoleType roleType = RoleType.Reader;
        bool removeAllPermission = true;
        // Removes user permissions
        clientContext.Site.RootWeb.RemovePermissionLevelFromUser(userLoginName, roleType,
            removeAllPermission);

        // Output
        Console.WriteLine("Removed user from the site permissions");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. To check the permission manually, navigate to Site Settings → Site Permissions.

Note

1. In the sample given above, the permissions are respect to the site/Web. If the method RemovePermissionLevelFromUser is accessed from the lists/permissions instead of the Web object, the user permissions will be removed for the respective objects.
2. If you are using the user object instead of the user name, the same operation can be carried out, using RemovePermissionLevelFromPrincipal method. The code snippet given below shows the logic.

```
clientContext.Site.RootWeb.RemovePermissionLevelFromPrincipal(user, roleType,
removeExistingPermission);
```

13.11 Grant read permissions to everyone except external users

In this section, you will learn how to provide read level permissions to all SharePoint users , using PnP Core CSOM library. AddReaderAccess method is used to provide the permissions. This will not include the permissions for the external users. For On-Premise sites, the input parameter is required.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the Web object, provide the permissions, using AddReaderAccess method. No input parameters are required.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
    {
        // Provides read permissions to all users except external users
        clientContext.Site.RootWeb.AddReaderAccess();

        // Output
        Console.WriteLine("Read permission granted for all users (except external users)");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

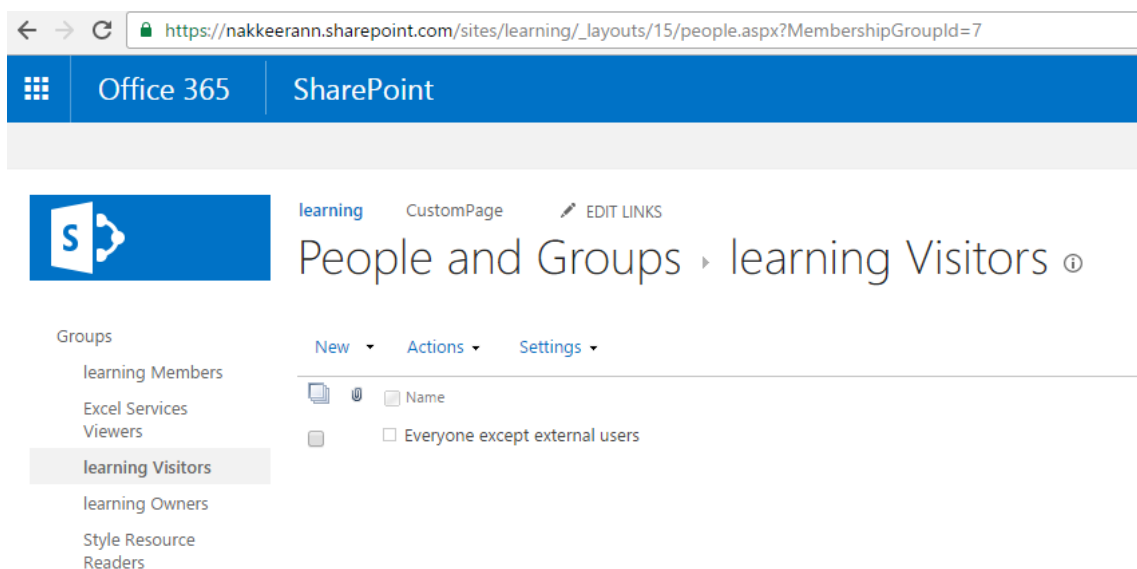
The results will be displayed on the console. The same can be accomplished with the input parameter.

```
clientContext.Site.RootWeb.AddReaderAccess(OfficeDevPnP.Core.Enums.BuiltInIdentity.EveryoneButExternalUsers)
```

For On-Premise sites, only the method given below will work.

```
clientContext.Site.RootWeb.AddReaderAccess(OfficeDevPnP.Core.Enums.BuiltInIdentity.Everyone)
```

To check the permission manually, navigate to Site Settings → Site Permissions → Visitors group. The snapshot given below shows the group with the added permissions.



14 Taxonomy Tasks

In this topic, you will learn how to perform taxonomy operations, using PnP Core CSOM Library.

The taxonomy term store of a site can be accessed on term stores page. For example, the term store URL is https://nakkeerann.sharepoint.com/_layouts/15/termstoremanager.aspx.

Please make sure, you are a term store administrator before executing any of the operations. It can be changed on the tenant site. The URL will be https://nakkeerann-admin.sharepoint.com/_layouts/15/termstoremanager.aspx.

The methods used for taxonomy operations are given below.

- CreateTermGroup
- EnsureTermGroup
- GetTermGroupName
- GetTermGroupById
- EnsureTermSet
- GetTermSetsByName
- AddTermToTermSet
- GetTermByName
- ImportTerms
- ExportTermSet
- ExportAllTerms
- GetTaxonomyItemByPath

14.1 How to create Term Group

In this section, you will learn how to create a term group on SharePoint taxonomy, using PnP Core CSOM library.

The example given below creates a term group called Cities on the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Retrieve the taxonomy session and appropriate taxonomy term store, using `GetDefaultSiteCollectionTermStore` method. Subsequently, create the term group ,

using CreateTermGroup method. The input parameters, which are required are given below.

- Term Store
- Group Name
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

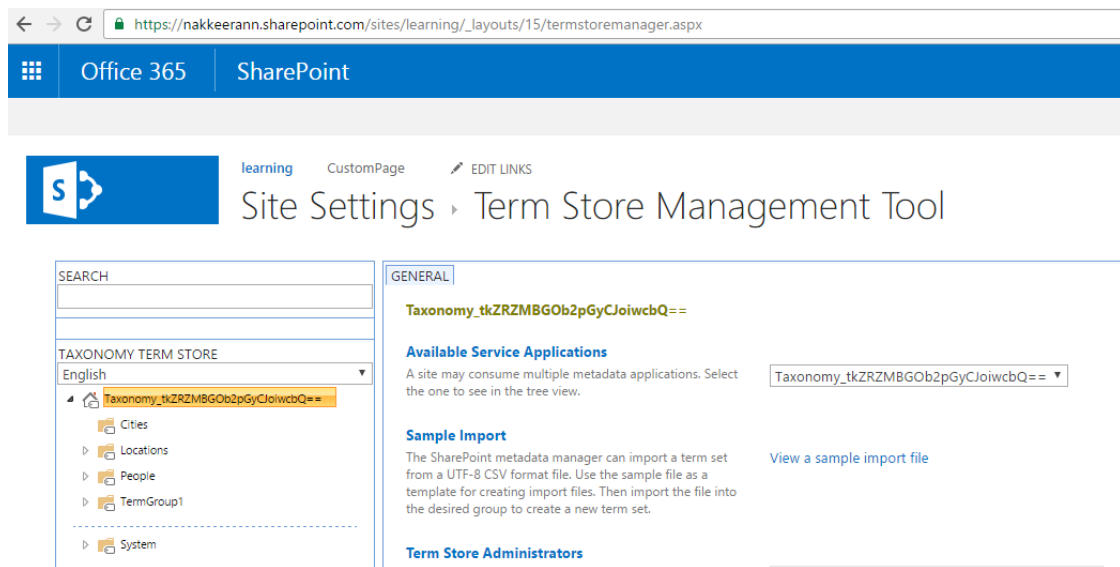
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string groupName = "Cities";

        // Retrieves taxonomy session using PnP method
        TaxonomySession taxonomySession =
        TaxonomyExtensions.GetTaxonomySession(clientContext.Site);
        if (taxonomySession != null)
        {
            // Retrieves term store
            TermStore termStore = taxonomySession.GetDefaultSiteCollectionTermStore();
            // PnP CSOM method to create term group
            TaxonomyExtensions.CreateTermGroup(termStore, groupName);
        }

        // Output
        Console.WriteLine("Term Group Created");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console. The snapshot given below shows the taxonomy page with the created term group.



14.2 How to ensure Term Group

In this section, you will learn how to ensure whether a term group exists or not. If it doesn't exist, the term group will be created on the term set of portal using PnP Core CSOM library.

The example given below ensures a term group called Cities on the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the `TaxonomyExtensions` class, term group is ensured, using `EnsureTermGroup` method. The required parameters are given below.
 - Site Object – Retrieved using client context.
 - Group Name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
```

```
// Input Parameters
string groupName = "Cities";

// PnP CSOM method to check & create term group
TermGroup termGroup = TaxonomyExtensions.EnsureTermGroup(clientContext.Site,
groupName);

// Output
Console.WriteLine("The following term group exists on the site : " + termGroup.Name);
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.

14.3 How to retrieve Term Group by Name

In this section, you will learn how to retrieve the term group by name from term store, using PnP Core CSOM library.

The example given below retrieves a term group called Cities from the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the TaxonomyExtensions class, term group is retrieved, using GetTermGroupNameBy method. The required parameters are given below.
 - Site Object – Retrieved using the client context.
 - Group Name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

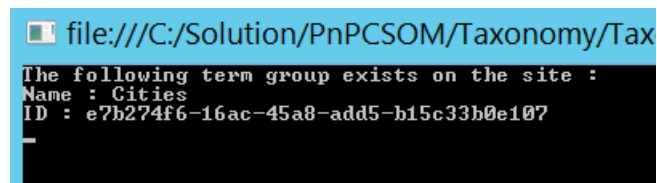
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string groupName = "Cities";

        // PnP CSOM method to retrieve term group
        TermGroup termGroup = TaxonomyExtensions.GetTermGroupByName(clientContext.Site,
            groupName);

        // Output
        Console.WriteLine("The following term group exists on the site : ");
        Console.WriteLine("Name : " + termGroup.Name);
        Console.WriteLine("ID : " + termGroup.Id);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console, as shown below.



14.4 How to retrieve Term Group by ID

In this section, you will learn how to retrieve the term group by ID from term store, using PnP Core CSOM library.

The following example retrieves a term group called Cities from the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the TaxonomyExtensions class, term group is ensured, using GetTermGroupId method. The required parameters are given below.
 - Site Object – Retrieved using client context.
 - Group ID – GUID.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        Guid groupId = new Guid("e7b274f6-16ac-45a8-add5-b15c33b0e107");

        // PnP CSOM method to retrieve term group by ID
        TermGroup termGroup = TaxonomyExtensions.GetTermGroupId(clientContext.Site,
            groupId);

        // Output
        Console.WriteLine("The following term group exists on the site : ");
        Console.WriteLine("Name : " + termGroup.Name);
        Console.WriteLine("ID : " + termGroup.Id);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.

14.5 How to create/ensure Term Set

In this section, you will learn how to create or ensure the term set on term store, using PnP Core CSOM library.

The example given below creates/ensures a term set called “Vellore” on term group “Cities” from the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the `TaxonomyExtensions` class, retrieve the term group and term set is created/ensured, using `EnsureTermSet` method. The required parameters are given below.
 - Term group name.
 - Term set name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string groupName = "Cities";
        string termSetName = "Vellore";
        // PnP CSOM method to retrieve term group
        TermGroup termGroup = TaxonomyExtensions.GetTermGroupByName(clientContext.Site,
            groupName);

        // Creates Term Set
        TermSet termSet = TaxonomyExtensions.EnsureTermSet(termGroup, termSetName);
        // Output
        Console.WriteLine("The following term set has been created/exists on the site term store : ");
    }
}
```

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

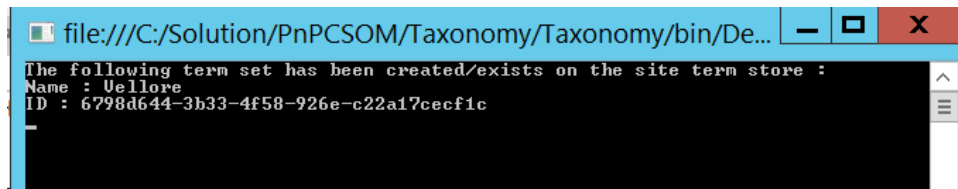
```

        Console.WriteLine("Name : " + termSet.Name);
        Console.WriteLine("ID : " + termSet.Id);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The results will be displayed on the console, as shown below.



14.6 How to retrieve the term sets

In this section, you will learn how to retrieve the term sets from term store, using PnP Core CSOM library.

The example given below retrieves a term set called “Vellore” from term group “Cities”.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the TaxonomyExtensions class, retrieve the term sets, using GetTermSetsByName method. The required parameters are given below.
 - Site object.
 - Term set name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

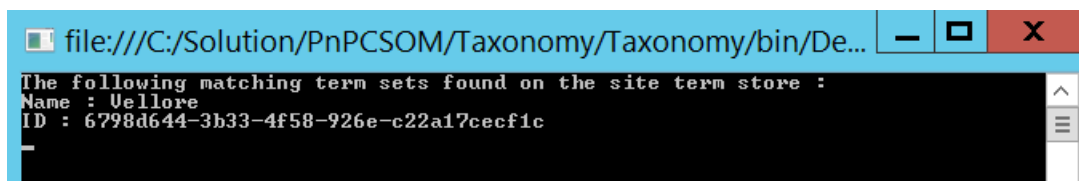
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string termSetName = "Vellore";

        // Retrieves Term Sets
        TermSetCollection termSets = TaxonomyExtensions.GetTermSetsByName(clientContext.Site,
            termSetName);
        // Output
        Console.WriteLine("The following matching term sets found on the site term store : ");
        foreach (TermSet termSet in termSets)
        {
            Console.WriteLine("Name : " + termSet.Name);
            Console.WriteLine("ID : " + termSet.Id);
        }

        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The snapshot given below shows the result on the console.



```
file:///C:/Solution/PnPCSOM/Taxonomy/Taxonomy/bin/De...
The following matching term sets found on the site term store :
Name : Uellore
ID : 6798d644-3b33-4f58-926e-c22a17cecf1c
```

14.7 How to create a Term on Term Set

In this section, you will learn how to create/add term on taxonomy term set, using PnP Core CSOM library.

The example given below creates a term called “Term1” on term set “Vellore” of the term group “Cities” from the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the `TaxonomyExtensions` class, add the terms, using `AddTermToTermset` method. The required parameters are given below.
 - Site object.
 - Source Term Set ID.
 - New Term Name.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string termName = "Term1";
        Guid termSetId = new Guid("6798d644-3b33-4f58-926e-c22a17cecf1c"); // Term Set Id of
        Vellore

        // Creates Term on Term Set
        Term term = TaxonomyExtensions.AddTermToTermset(clientContext.Site, termSetId,
            termName);
        // Output
        Console.WriteLine("The following term has been added to term set : ");

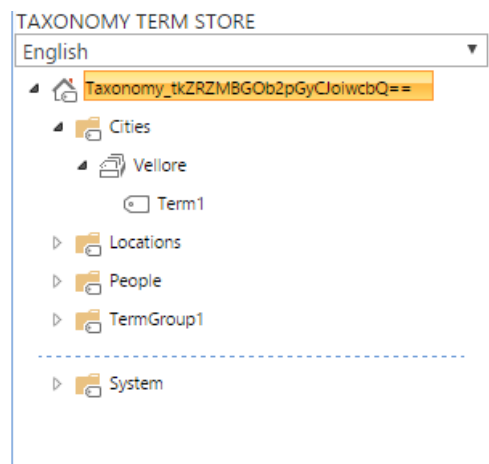
        Console.WriteLine("Name : " + term.Name);
        Console.WriteLine("ID : " + term.Id);

        Console.ReadKey();
    }
}
```

```
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console. The snapshot given below shows the term added to the term set.



14.8 How to retrieve the term from term set

In this section, you will learn how to retrieve the term from taxonomy term set, using PnP Core CSOM library.

The example given below retrieves a term called “Term1” from the term set “Vellore” of the term group “Cities” on the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the TaxonomyExtensions class, retrieve the term, using GetTermByName method. The required parameters are given below.
 - Site object.
 - Source Term Set ID.
 - Term Name.
- In the program file, insert the code given below and save the file.

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string termName = "Term1";
        Guid termSetId = new Guid("6798d644-3b33-4f58-926e-c22a17cecf1c"); // Term Set Id of
        Vellore

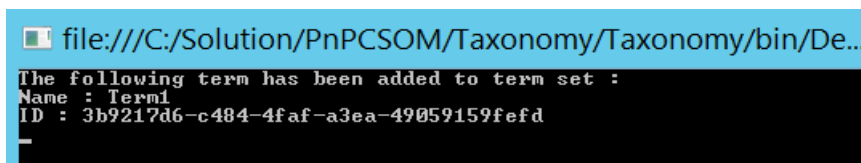
        // Retrieves Term from Term Set
        Term term = TaxonomyExtensions.GetTermByName(clientContext.Site, termSetId, termName);
        // Output
        Console.WriteLine("The following term has been added to term set : ");

        Console.WriteLine("Name : " + term.Name);
        Console.WriteLine("ID : " + term.Id);

        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console, as shown below.



```
file:///C:/Solution/PnPCSOM/Taxonomy/Taxonomy/bin/De...
The following term has been added to term set :
Name : Term1
ID : 3b9217d6-c484-4faf-a3ea-49059159fefb
```

14.9 How to retrieve the term from the term store by the path

In this section, you will learn how to retrieve term from taxonomy term store by the path, using PnP Core CSOM library.

The example given below retrieves a term called “Term1” from the term set “Vellore” of the term group “Cities” on the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the `TaxonomyExtensions` class, retrieve the term, using `GetTaxonomyItemByPath` method. The required parameters are given below.
 - Site object.
 - Term path separated by delimiters.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Term path on term store
        string termPath = "Cities|Vellore|Term1";
        // Retrieve term by path
        TaxonomyItem taxonomyItem =
        TaxonomyExtensions.GetTaxonomyItemByPath(clientContext.Site, termPath);
        // Output
        Console.WriteLine("Term details :");
        Console.WriteLine("ID : " + taxonomyItem.Id);
        Console.WriteLine("Name : " + taxonomyItem.Name);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```


Result

The result will be displayed on the console, as shown below.

```
file:///C:/Solution/PnPSCOM/Taxonomy/Ta
Term details :
ID : 3b9217d6-c484-4faf-a3ea-49059159fefc
Name : Term1
-
```

14.10 How to import the terms

In this section, you will learn how to import terms into the existing taxonomy term sets, using PnP Core CSOM library.

The example given below imports few terms into the term set “Vellore” of the term group “Cities” on the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the `TaxonomyExtensions` class, add the terms to the term set, using `ImportTerms` method. The required parameters are given below.
 - Site object.
 - Terms – String array which contains the terms defined as `TermGroup|TermSet|Term`
 - Locale value.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

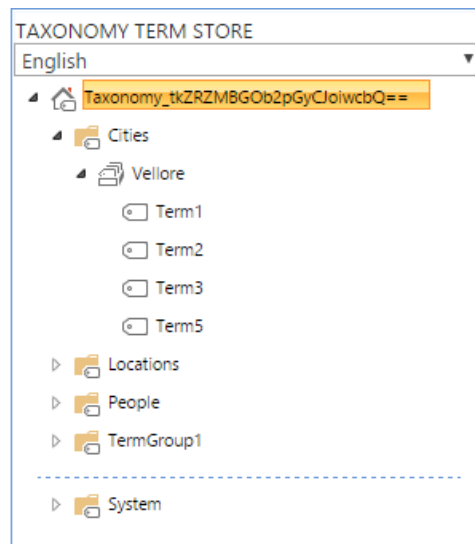
```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string[] terms = { "Cities|Vellore|Term2", "Cities|Vellore|Term3", "Cities|Vellore|Term5" };
        // Imports Terms
        TaxonomyExtensions.ImportTerms(clientContext.Site, terms, 1033);
    }
}
```

```
// Output
Console.WriteLine("The terms has been imported.");

Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console. The terms which are import can be viewed from the term store page.



14.11 How to export the term set

In this section, you will learn how to export the term set with the terms from taxonomy term sets, using PnP Core CSOM library.

The example given below exports the terms from the term set “Vellore” of the term group “Cities” on the taxonomy term store.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the `TaxonomyExtensions` class, export the term sets, using `ExportTermSet` method. The required parameters are given below.
 - Site object
 - Term Set ID
 - IncludeId - Export ID's as well (Boolean)
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        Guid termSetId = new Guid("6798d644-3b33-4f58-926e-c22a17cecf1c"); // Term Set Id of
        Vellore
        // Export Term Set
        List<string> termSets = TaxonomyExtensions.ExportTermSet(clientContext.Site, termSetId,
            true);
        // Output
        foreach(string term in termSets)
        {
            Console.WriteLine(term);
        }

        Console.ReadKey();
    }
}
catch (Exception ex)
{
}
```

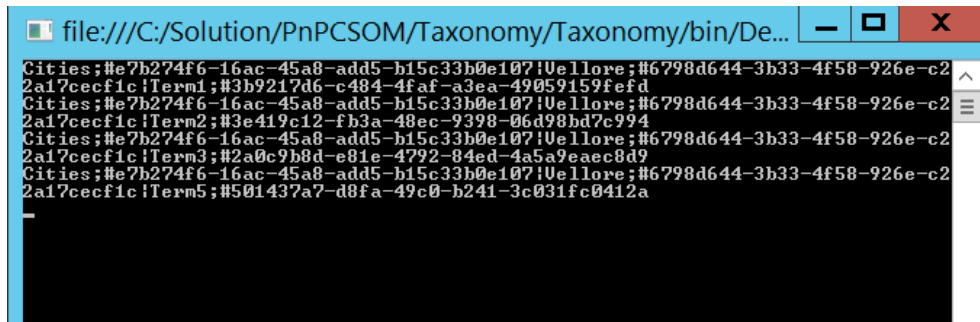
```

Console.WriteLine("Error Message: " + ex.Message);
Console.ReadKey();
}

```

Result

The result will be displayed on the console.



Note

To get only the names in the result, Include Id parameter, which should be set to false.

14.12 How to export all Terms

In this section, you will learn how to export all the terms from taxonomy term store, using PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Using the TaxonomyExtensions class, export the term sets, using ExportAllTerms method. The required parameters are given below.
 - Site object.
 - IncludeId - Export IDs as well (Boolean).
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```

try
{
    // Get and set the client context
}

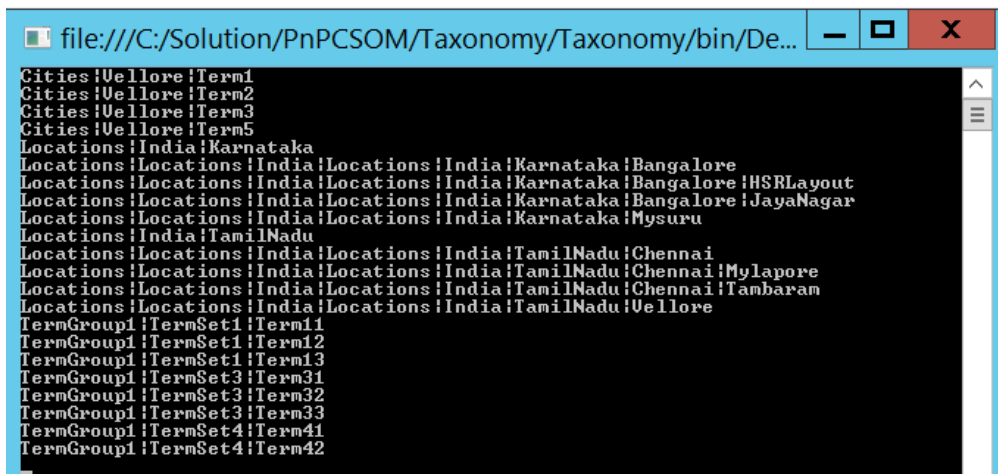
```

```
// Connects to SharePoint online site using inputs provided
using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
userName, password))
{
    List<string> terms = TaxonomyExtensions.ExportAllTerms(clientContext.Site, false);
    // Output
    foreach (string term in terms)
    {
        Console.WriteLine(term);
    }

    Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console.



```
file:///C:/Solution/PnPCSOM/Taxonomy/Taxonomy/bin/De...
Cities!Uellore!Term1
Cities!Uellore!Term2
Cities!Uellore!Term3
Cities!Uellore!Term5
Locations!India!Karnataka
Locations!Locations!India!Locations!India!Karnataka!Bangalore
Locations!Locations!India!Locations!India!Karnataka!Bangalore!HSRLayout
Locations!Locations!India!Locations!India!Karnataka!Bangalore!JayaNagar
Locations!Locations!India!Locations!India!Karnataka!Mysuru
Locations!India!TamilNadu
Locations!Locations!India!Locations!India!TamilNadu!Chennai
Locations!Locations!India!Locations!India!TamilNadu!Chennai!Mylapore
Locations!Locations!India!Locations!India!TamilNadu!Chennai!Tambaram
Locations!Locations!India!Locations!India!TamilNadu!Vellore
TermGroup1!TermSet1!Term11
TermGroup1!TermSet1!Term12
TermGroup1!TermSet1!Term13
TermGroup1!TermSet3!Term31
TermGroup1!TermSet3!Term32
TermGroup1!TermSet3!Term33
TermGroup1!TermSet4!Term41
TermGroup1!TermSet4!Term42
```

Note

To get term/term set GUIDs within the result, IncludeId parameter should be set to true.

14.13 How to create Taxonomy Field on Site/List

In this section, you will learn how to create a taxonomy field on SharePoint site or the list, using PnP Core CSOM library.

The example give below creates a taxonomy field on a site with the term store value.

Steps involved

- Open Visual Studio as administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Retrieve the term from term store, using section 14.9
- Using the `TaxonomyExtensions` class, create a taxonomy field, using `CreateTaxonomyField` method. The required parameters are given below.
 - Web object or list object.
 - Field Creation info, which contains the field details like GUID, internal name, display name, group and taxonomy item.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Term path on term store
        string termPath = "Cities|Vellore|Term1";
        // Retrieve term by path
        TaxonomyItem taxonomyItem =
        TaxonomyExtensions.GetTaxonomyItemByPath(clientContext.Site, termPath);

        // Input parameters for taxonomy field
        string displayName = "NakkeeranTaxColumn1";
        string internalName = "NakkeeranTaxColumn1";
        string groupName = "NavColumnGroup";
        string fieldId = "90EBCB00-DAF3-48C0-BD5D-120951CF8CF6";
        Guid fieldGuid = new Guid(fieldId);

        // Field creation info
        TaxonomyFieldCreationInformation fieldInfo = new TaxonomyFieldCreationInformation();
        fieldInfo.Id = fieldGuid;
        fieldInfo.InternalName = internalName;
```

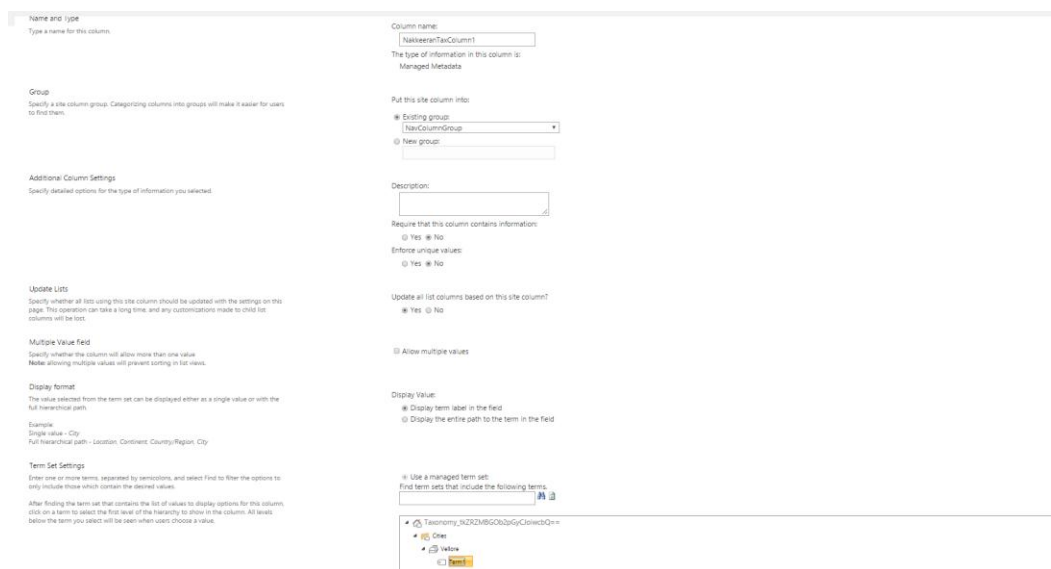
```

fieldInfo.DisplayName = displayName;
fieldInfo.Group = groupName;
fieldInfo.TaxonomyItem = taxonomyItem;
// Creates taxonomy field
Field taxField = TaxonomyExtensions.CreateTaxonomyField(clientContext.Site.RootWeb,
fieldInfo);
// Output
Console.WriteLine("Field Created details :");
Console.WriteLine("ID : " + taxField.Id);
Console.WriteLine("Name : " + taxField.InternalName);
Console.ReadKey();
}
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
}

```

Result

The results will be displayed on the console. The field can be viewed from the site columns/list column pages, as shown below.



14.14 How to update Taxonomy Field on List Item

In this section, you will learn how to update a taxonomy field value of the list item, using PnP Core CSOM library.

The example given below updates a list item with the term store value.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.
- Retrieve the list item to be updated.
- Using the `TaxonomyExtensions` class, update the taxonomy field value, using `SetTaxonomyFieldValueByTermPath` method. The required parameters are given below.
 - List item
 - Term path
 - Field GUID
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Term path on term store
        string termPath = "Cities|Vellore|Term2";
        // Input parameters for taxonomy field
        string fieldId = "C584C999-1E90-4D67-AA99-D05374E25C38";
        Guid fieldGuid = new Guid(fieldId);

        ListItem listItem = clientContext.Site.RootWeb.GetListByTitle("TestList").GetItemById(3);
        clientContext.Load(listItem);
        clientContext.ExecuteQuery();

        // Updates taxonomy field value
        TaxonomyExtensions.SetTaxonomyFieldValueByTermPath(listItem, termPath, fieldGuid);
        // Output
        Console.WriteLine("List item updated");
        Console.ReadKey();
    }
}
```



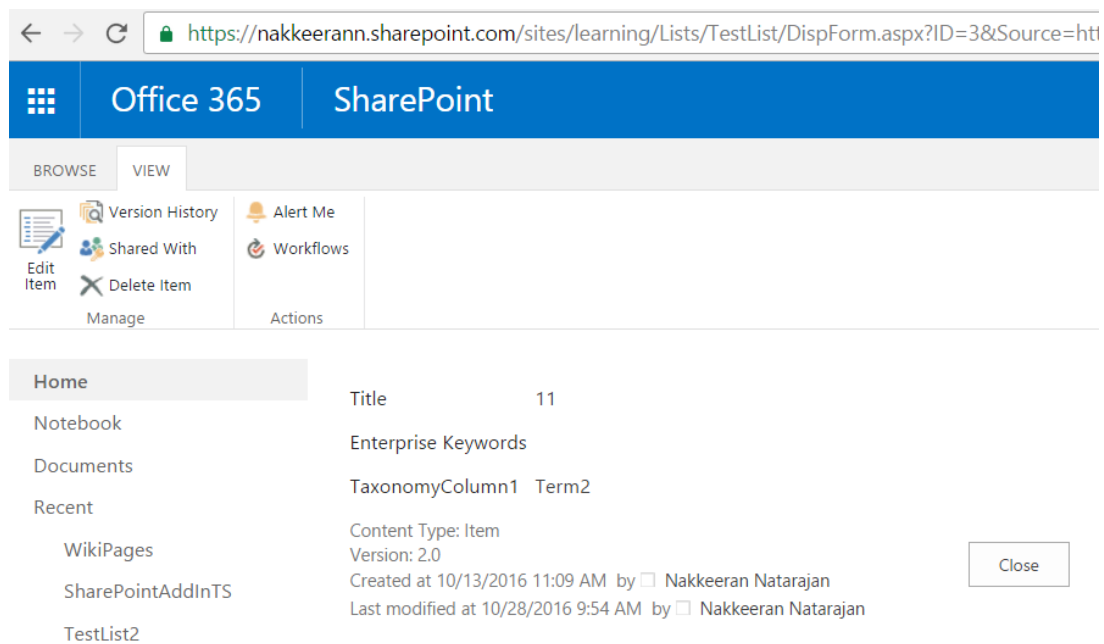
```

}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}

```

Result

The results will be displayed on the console. The snapshot given below shows the updated value.



14.15 How to remove Taxonomy Field from Site

In this section, you will learn how to remove a taxonomy field from SharePoint site, using PnP Core CSOM library.

The example given below removes a list item with the term store value.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object.

- Using the TaxonomyExtensions class, remove the field from the site, using RemoveTaxonomyFieldByInternalName method. The required parameters are given below.
 - Web object
 - Field name
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input parameters
        string fieldName = "NakkeeranTaxColumn1";

        // Removes taxonomy field
        TaxonomyExtensions.RemoveTaxonomyFieldByInternalName(clientContext.Site.RootWeb,
            fieldName);
        // Output
        Console.WriteLine("Field removed");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The results will be displayed on the console.

Note

The field can be removed from the site, using field ID as well. The method for removing using ID is RemoveTaxonomyFieldById(web,fieldGuid).

15 Office 365 Site Collection tasks

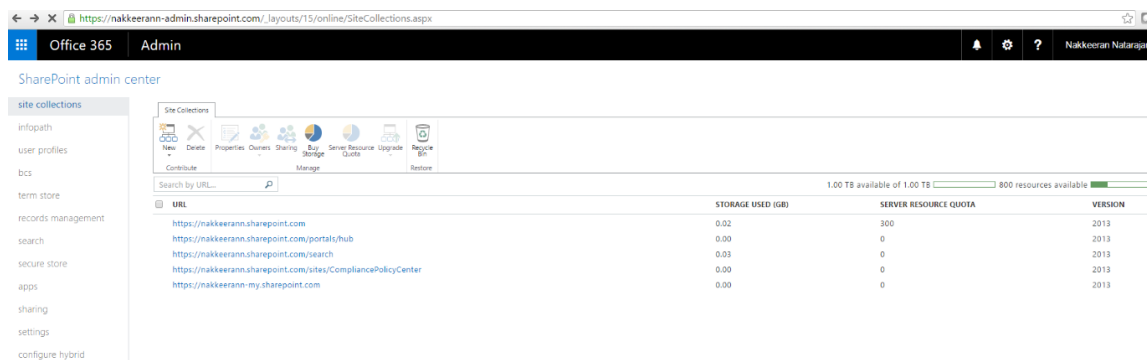
In this topic, you will learn how to perform the operations on Office 365 tenant site, using PnP Core CSOM library. These operations can be used only on Office 365 sites.

The methods given below are majorly used for the operations in this topic.

- CreateSiteCollection.
- GetSiteCollections.
- CheckIfSiteExists.
- DeleteSiteCollection.

Office 365 tenant site will look like https://***-admin.sharepoint.com.

The site collections can be accessed from https://***-admin.sharepoint.com/_layouts/15/online/SiteCollections.aspx. The snapshot given below shows the site collections page.



The screenshot shows the SharePoint admin center interface. The left sidebar contains navigation links: site collections, infopath, user profiles, bcs, term store, records management, search, secure store, apps, sharing, settings, and configure hybrid. The main content area is titled 'Site Collections' and includes a search bar, a table of site collections, and a progress bar showing storage usage.

URL	STORAGE USED (GB)	SERVER RESOURCE QUOTA	VERSION
https://nakkeerann.sharepoint.com	0.02	300	2013
https://nakkeerann.sharepoint.com/portals/hub	0.00	0	2013
https://nakkeerann.sharepoint.com/search	0.03	0	2013
https://nakkeerann.sharepoint.com/sites/CompliancePolicyCenter	0.00	0	2013
https://nakkeerann-my.sharepoint.com	0.00	0	2013

15.1 How to create a site collection on tenant site

In this section, you will learn how to create a site collection on Office 365 tenant site.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object. In place of site user details, URL (https://***-admin.sharepoint.com) of the tenant site should be passed along with the tenant user credentials.

- Using the client context object, create the site collection, using CreateSiteCollection method. The input parameters are initialized and passed, using SiteEntity class. The required parameters are given below.
 - Title
 - Template
 - URL
 - Owner
 - Maximum Storage
 - Maximum resources
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        var tenant = new Tenant(clientContext);
        SiteEntity siteEntity = new SiteEntity();
        siteEntity.Title = "Samplepnp";
        siteEntity.Template = "STS#0";
        siteEntity.Url = "https://***.sharepoint.com/sites/Samplepnp";
        siteEntity.SiteOwnerLogin = "abc@nakkeerann.onmicrosoft.com";
        siteEntity.UserCodeMaximumLevel = 10;
        siteEntity.StorageMaximumLevel = 30;

        Guid tenantSiteCollectionId = tenant.CreateSiteCollection(siteEntity);

        // Output
        Console.WriteLine("Site Collection is created " + tenantSiteCollectionId);
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console. The created site collection can be accessed from the tenant site (https://***-admin.sharepoint.com/_layouts/15/online/SiteCollections.aspx).

15.2 How to retrieve the site collections from the tenant site

In this section, you will learn how to retrieve the site collections from Office 365 tenant site.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book for importing the packages and creating authentication manager object. In place of site user details, URL (https://***-admin.sharepoint.com) of the tenant site should be passed along with the tenant user credentials.
- Using the client context object, retrieve the site collection, using `GetSiteCollections` method. The return object will be the list containing the site collection as the site entity type.
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        var tenant = new Tenant(clientContext);
        List<SiteEntity> siteCollections = tenant.GetSiteCollections() as List<SiteEntity>;

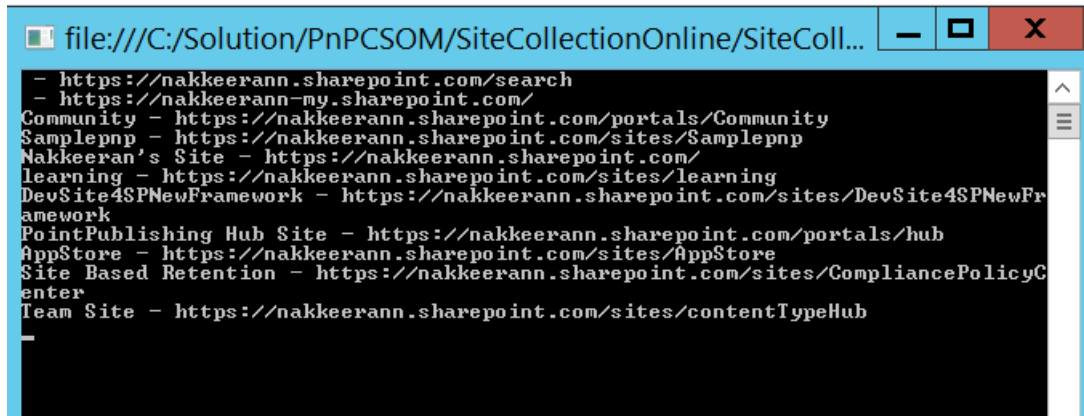
        // Output
        foreach (SiteEntity siteCollection in siteCollections)
        {
            Console.WriteLine(siteCollection.Title + " - " + siteCollection.Url);
        }

        Console.ReadKey();
    }
}
catch (Exception ex)
```

```
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console, as shown below.



```
file:///C:/Solution/PnP/PCSOM/SiteCollectionOnline/SiteColl...
- https://nakkeerann.sharepoint.com/search
- https://nakkeerann-my.sharepoint.com/
Community - https://nakkeerann.sharepoint.com/portals/Community
Samplepnp - https://nakkeerann.sharepoint.com/sites/Samplepnp
Nakkeeran's Site - https://nakkeerann.sharepoint.com/
learning - https://nakkeerann.sharepoint.com/sites/learning
DevSite4SPNewFramework - https://nakkeerann.sharepoint.com/sites/DevSite4SPNewFr
amework
PointPublishing Hub Site - https://nakkeerann.sharepoint.com/portals/hub
AppStore - https://nakkeerann.sharepoint.com/sites/AppStore
Site Based Retention - https://nakkeerann.sharepoint.com/sites/CompliancePolicyC
enter
Team Site - https://nakkeerann.sharepoint.com/sites/contentTypeHub
```

The site collections can be accessed from the tenant site (https://***-admin.sharepoint.com/layouts/15/online/SiteCollections.aspx).

15.3 How to check if a site collection exists on tenant site or not

In this section, you will learn how to check whether a site collection exists on a tenant site or not, using PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object. In place of site user details, URL (https://***-admin.sharepoint.com) of the tenant site should be passed along with the tenant user credentials.
- Using the client context object, check the status of site collection, using `CheckIfSiteExists` method. The return type is Boolean. The input parameters are given below.
 - Site Collection URL
 - Status (Active, Creating or Recycled)
- In the program file, insert the code given below and save the file.

- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string siteCollectionUrl = "https://nakkeerann.sharepoint.com/sites/Samplepnp";
        string siteStatus = "Active";
        var tenant = new Tenant(clientContext);

        bool siteExists = tenant.CheckIfSiteExists(siteCollectionUrl, siteStatus);

        // Output
        if (siteExists)
        {
            Console.WriteLine("Site is Active");
        }
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```

Result

The result will be displayed on the console.

15.4 How to delete a site collection

In this section, you will learn how to delete a site collection from the tenant site, using PnP Core CSOM library.

Steps involved

- Open Visual Studio as an administrator.
- Follow the steps explained in [section 4 \(Connect to SharePoint site\)](#) of the book to import the packages and create authentication manager object. In place of the site user details, the URL (https://***-admin.sharepoint.com) of the tenant site should be passed along with the tenant user credentials.
- Using the client context object, remove the site collection from the tenant site, using DeleteSiteCollection method. The required input parameters are given below.
 - Site collection URL
 - Use recycle bin (Boolean) – Whether to use recycle bin or not?
- In the program file, insert the code given below and save the file.
- Run the code from Debug menu or by pressing F5.

Code

```
try
{
    // Get and set the client context
    // Connects to SharePoint online site using inputs provided
    using (var clientContext = authManager.GetSharePointOnlineAuthenticatedContextTenant(siteUrl,
        userName, password))
    {
        // Input Parameters
        string siteCollectionUrl = "https://nakkeerann.sharepoint.com/sites/Samplepnp";
        bool useRecycleBin = true;
        var tenant = new Tenant(clientContext);

        // Deletes site collection
        tenant.DeleteSiteCollection(siteCollectionUrl, useRecycleBin);
        Console.WriteLine("Site Collection deleted");
        Console.ReadKey();
    }
}
catch (Exception ex)
{
    Console.WriteLine("Error Message: " + ex.Message);
    Console.ReadKey();
}
```


Result

The result will be displayed on the console. The site collection will be deleted and moved to recycle bin.

Note

To remove the site collection from recycle bin, `DeleteSiteCollectionFromRecycleBin` method is used.

16 Summary

Thus we have learnt programming on PnP Core CSOM library programming. The book covers all the basic operations required to access the various objects on SharePoint instances. The advanced or complex code can be developed and executed from the base samples given above. The operations explained in the sections mentioned above are compatible for SharePoint 2013, SharePoint 2016 and SharePoint online (Office 365) versions.

17 References

<https://github.com/OfficeDev/PnP-Sites-Core/blob/master/Core/Documentation.md>