**SharePoint** Development Training

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**-: Day 1:-**

**Introduction to SharePoint:**

SharePoint is a platform to support collaboration and Content management system.

It is a central web-based portal where we can manage document, social activities, data and information.

**SharePoint can do:**

1.) Collaboration -> Collaboration is the strong theme of SharePoint which means bringing people together through different types of collaboration such as enterprise content management.

2.) Interoperability -> Capability to build and deploy secure and custom solutions that integrate line of business data with SharePoint and Office.

3.) Platform->SharePoint is also a platform that supports extensibility, through a rich object model, a solid set of developer tools, and a growing developer community.

**History of SharePoint:**

SharePoint first launched in 2001 By Microsoft after that some more versions launched like:

* SharePoint 2001: In 2001, Microsoft launched SharePoint Portal Server (SPS) and SharePoint Team Services (STS). SPS provided three functions i.e., Portal, Document Management & Search. STS on the other hand with FrontPage and provided team workspaces for collaboration.
* SharePoint 2003;
* SharePoint 2007;
* SharePoint 2010;
* SharePoint 2013;
* SharePoint 2016: SharePoint 2016 enhance the hybrid infrastructure (merging SharePoint On-Premises sites with SharePoint Online) and to take advantage of recent innovations in cloud technology.

**Features of SharePoint:**

* Enterprise Search Engine,
* Enterprise Content Management, Business Data Catalog,
* Excel Services, Faster Site Creation,
* List, Libraries,
* We can develop site,
* Customise site,
* Create Rest API,
* By Using Rest API, we can insert data in site.

**Prerequisite:**

* HTML
* CSS
* JavaScript
* Bootstrap
* C#
* jQuery
* React js
* CSOM
* JSOM
* SQL

**Key Feature:**

* Business connectivity Services,
* Record Management,
* Advance Searching,
* Web content Management,
* Enterprise management,
* Record Management,
* Site,
* Search,
* Branding,
* Compatibility,
* Interoperability,

Two SharePoint Premises:

SharePoint online by subscription;

SharePoint on premises by instilling on our PC;

---------------------------------------------------------------------------------------------

**-: Day 2 :-**

**Introduction of SharePoint Development platform**

SharePoint is a platform to support collaboration and content management system. It is a central web-based portal. Using SharePoint, we can manage our colleague’s and our own documents, social activities, data, and information.

* It allows groups to set up a centralized, password-protected space for document sharing.
* Documents can be stored, downloaded and edited, then uploaded for continued sharing.
* SharePoint offers such a wide array of features that it is very challenging for any one person to be an expert across all the workloads.

**Learnt with sort demo of SharePoint:**

* Default template on SharePoint and way to customise according to the user requirement.
* How to add List and libraries into the site.
* Shown how to insert different component of SharePoint into the site.

**SharePoint Master Page:**

Master Page: SharePoint master pages provide the interface and overall layout of the pages on a SharePoint site. The common elements of a page – its header, navigation links, Site Actions menu, and so forth they are placed in the same areas regardless of the page you’re viewing.

---------------------------------------------------------------------------------------------

**-: Day 3 :-**

**Difference Between List and Libraries:**

**List:** A SharePoint list is like a table in a database or a table in excel. The SharePoint list will have rows and columns that we will use to store information. For example, we may like to store Training Course details like: Training Course Name, Duration, Price etc;

* It stores homogeneous data;
* Version control is not possible.
* Version control is possible for list item
* Check in, Check-Out is not possible.

**Libraries:** SharePoint libraries are special types of lists that are created to store documents. Each file in a SharePoint document library is like one item. It also has columns or fields or properties.

SharePoint also provides various libraries for specific proposes like picture library, form library, etc.

* Different data can be stored like video, music, website etc,
* Version control is possible.
* Check in, Check-Out is possible.

**SharePoint Site Hierarchy**

**Three-layer architecture of SharePoint:**

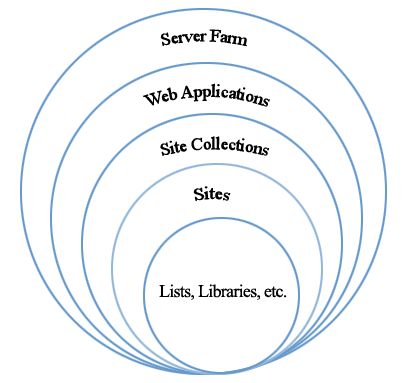


Figure : Diagram Shows the architectural hierarchy

**Server Farm:**  This is the top level of SharePoint; this level contains the Web Applications.

**Web Application:** This SharePoint instance contains the site collections.

**Site Collection:**  Site collections are at the third level within SharePoint hierarchy. The site collection has no parent site and as such is the first level of the hierarchy we can access. The site collection can create multiple sections underneath it which includes other sites/sub-sites and libraries.

**Site (Sub Site Collection):** The next level in the hierarchy are Sites, these are contained within the site collection, however, a site can have either a single parent or multiple parents within the site collection.

**List:** List and Libraries are the bottom level of the hierarchy. A SharePoint list is a collection of data that has some kind of structure to it

**Libraries:** SharePoint libraries are special types of lists that are created to store documents.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**-: Day 4 :-**

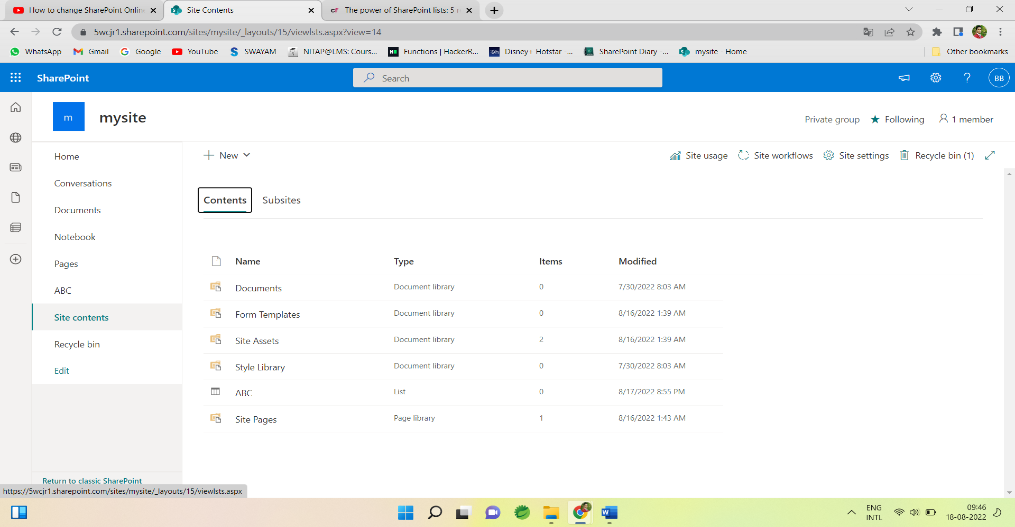
**SharePoint List**

A SharePoint list is a collection of data that has some kind of structure to it. It is like a table, a spreadsheet or a simple database. It can include many different types of information including numbers, text and even images.

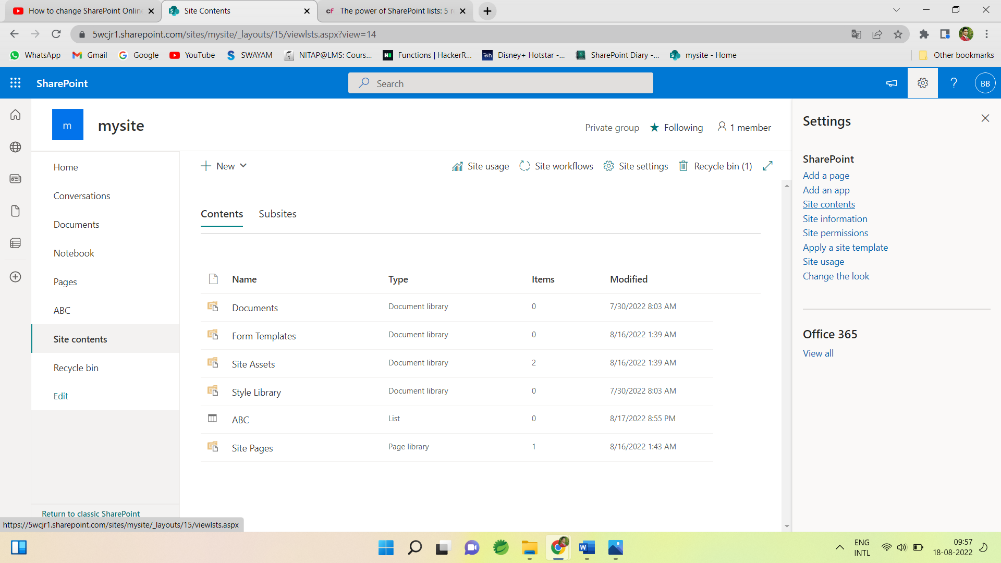
We can create and list that help to track issues, assets, routines, contacts, and more. Start from a template, Excel file, or from scratch.

**Create a list**

From SharePoint site home page

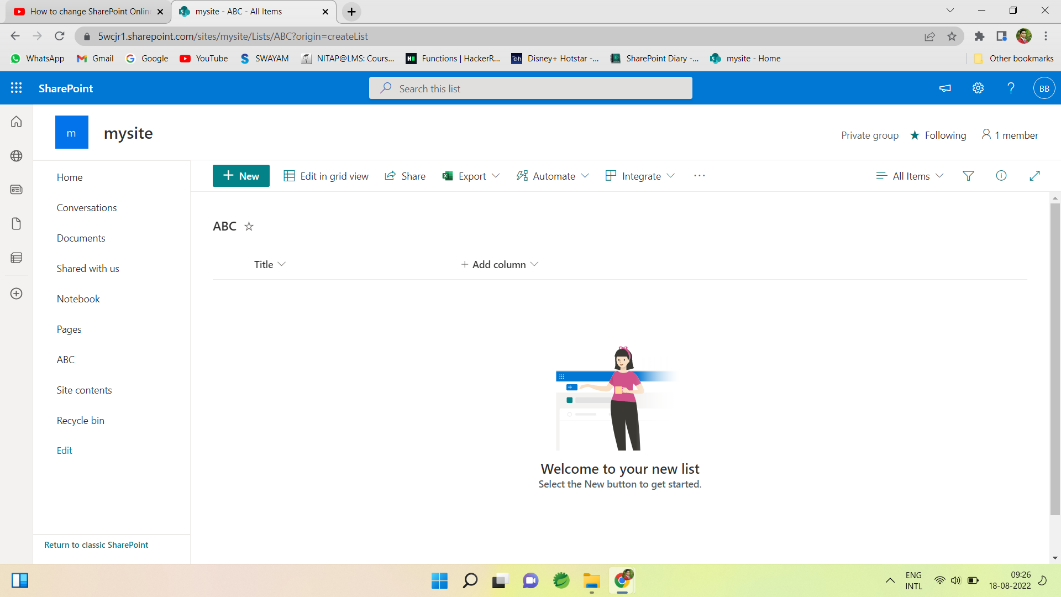


Or the Site contents page,

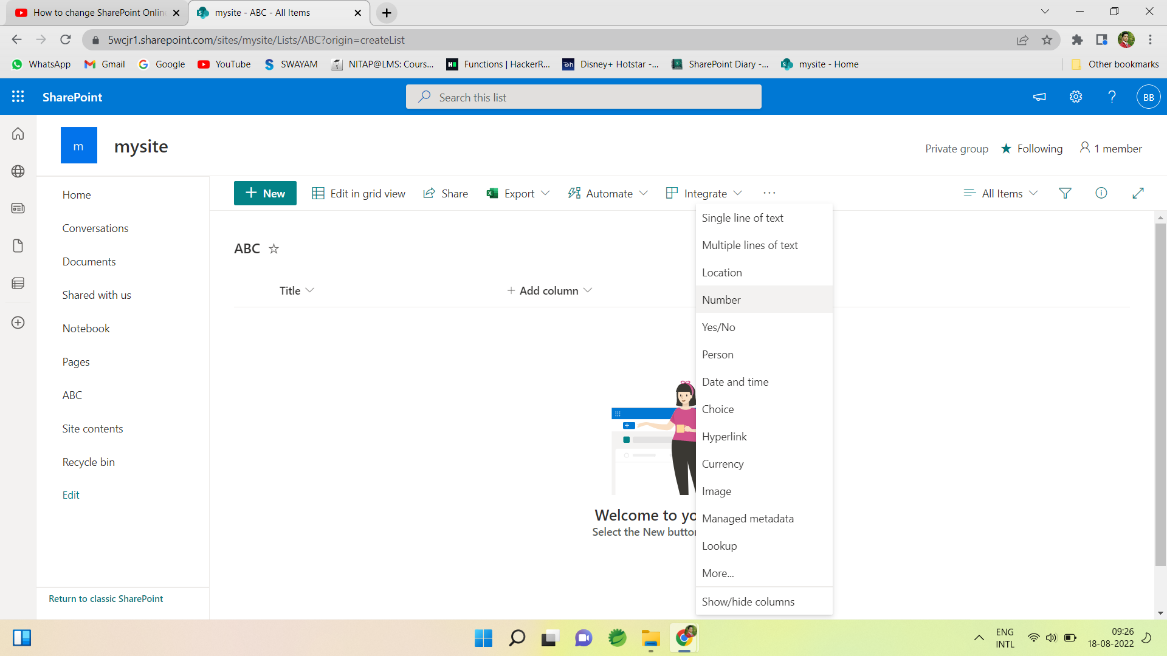


Then select -> + New -> List.

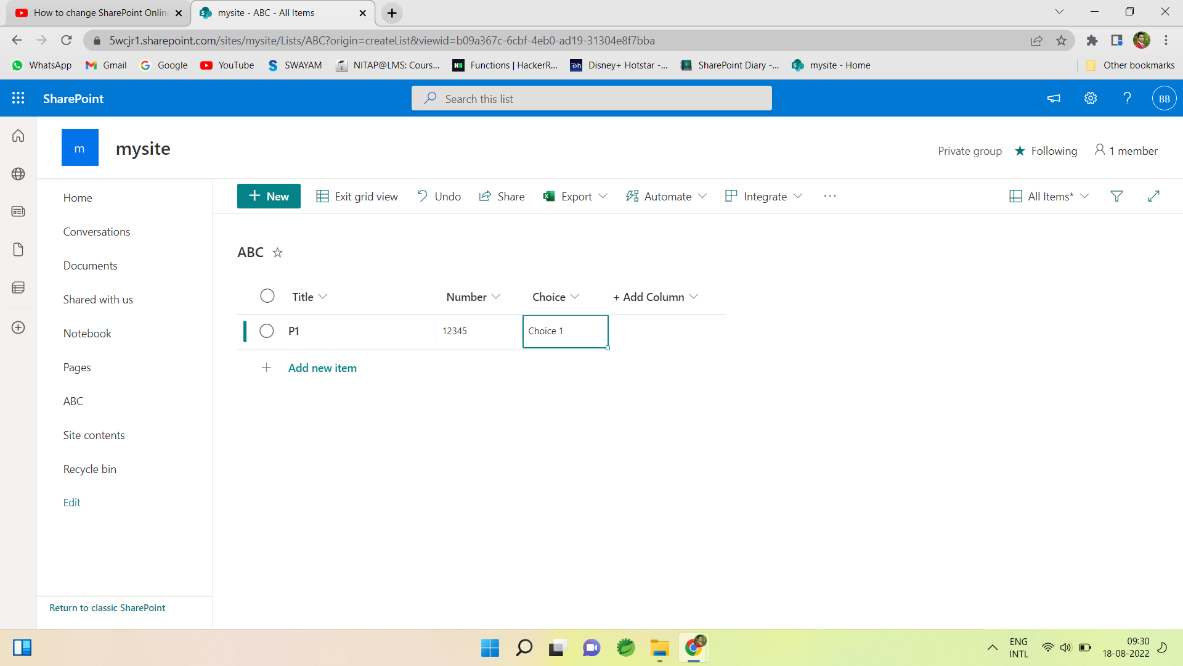
Now choose a Blank list and specify the name and description and List gets created



Now we can add column with specific type such as Number, Person, True/False, lookup, Image and many more by clicking on Add column option.



After Choosing the type of column we need to give name and description for that and column added to the list.



Similarly, we can add more column as requirement by its data type.

All created list is stored in site content which contain whole site content like list, libraries, documents etc.

After creating a list, we can edit list directly by clicking on Edit in grid view.

We can export as excel sheet by Export option.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Active Directory: Directory Which containing users, employee. Active Directory sync the users and after fetching we can extract all the necessary details.

Lookup: It is used to get the information from another table just like foreign key in SQL.

Calculated: SharePoint uses the calculated column to populate values based on some formula. The calculation can depend on other column values also, that can use other columns to calculate the values also.

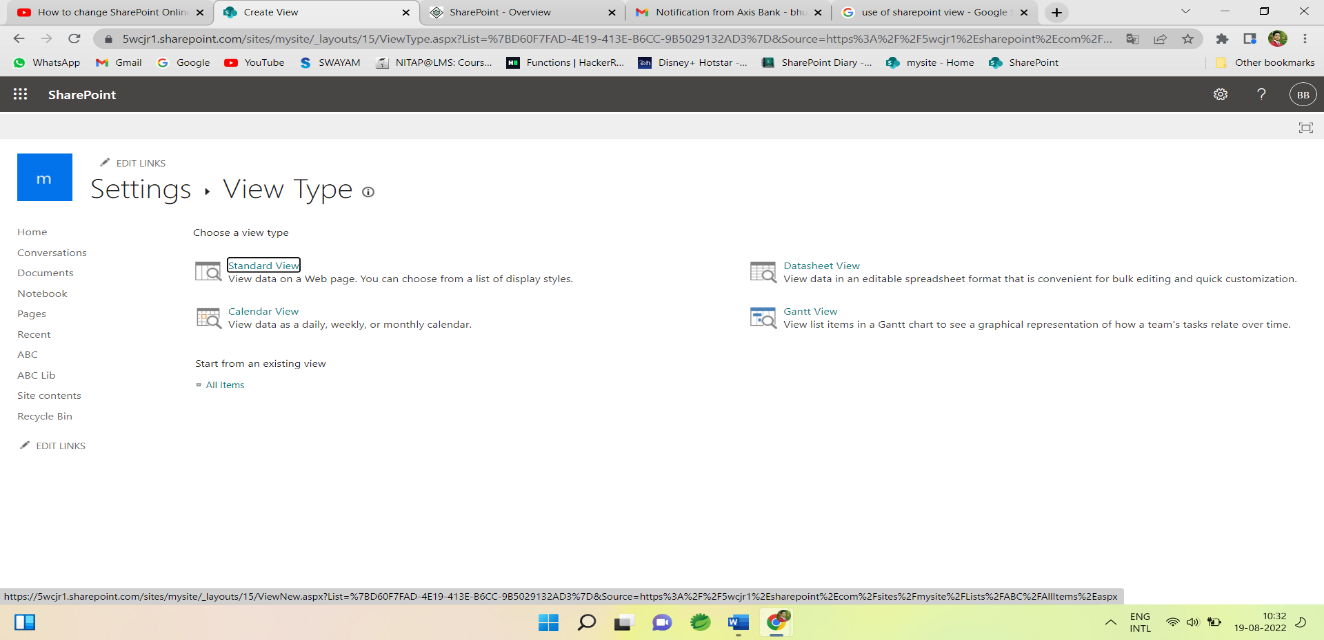
**-: Day 5 :-**

**SharePoint View**

We can create custom views of lists and libraries to organize and show items that are important (like certain columns), to add filtering or sorting, or to have a more engaging style. You can create a personal view (that only you can see) or, if you have permissions to do so, you can create a public view for everyone who uses the list to see.

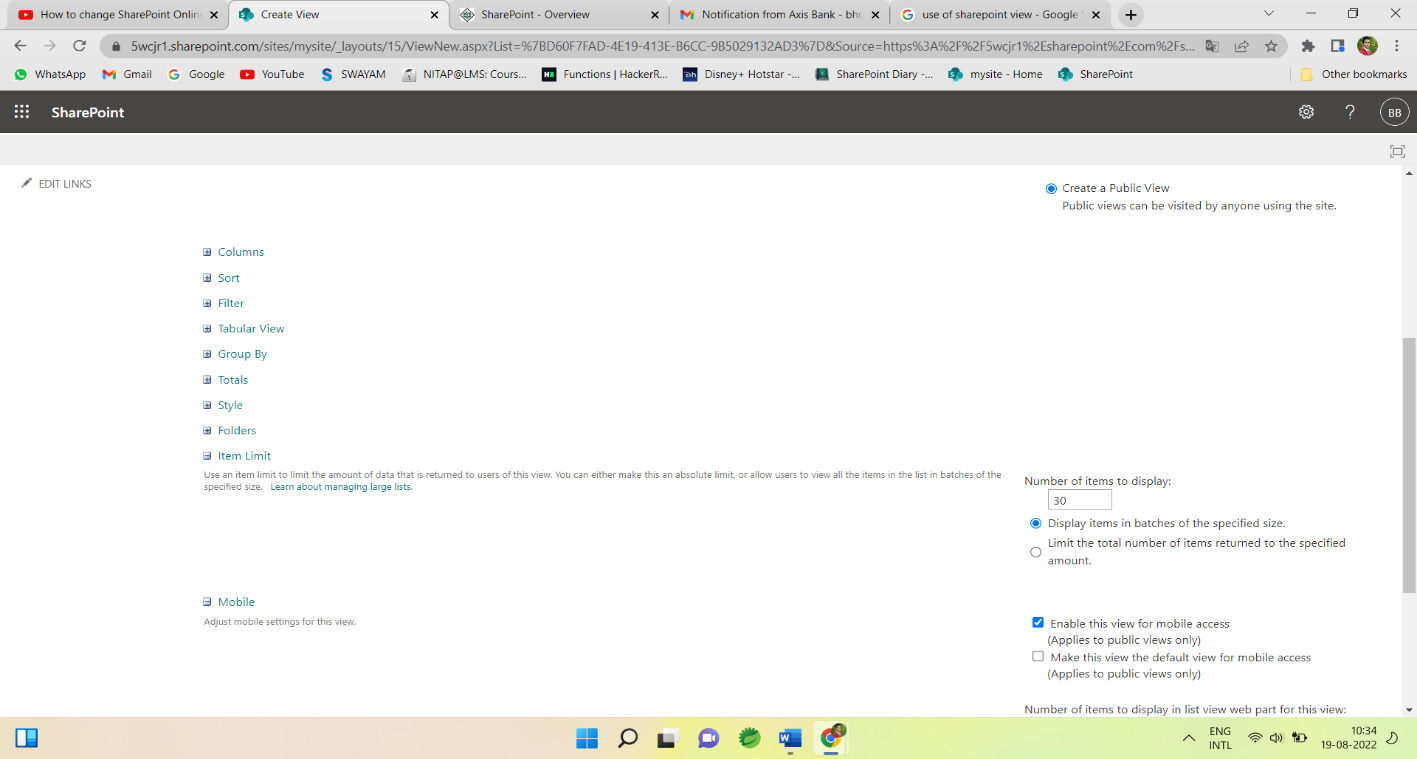
**Types of SharePoint views:**

* Standard view
* Datasheet view
* Calendar view
* Gantt view



**By using this view we can use some functionality like:**

* Columns
* Filter
* Sort
* Tabular view
* Group by
* Tables
* Style
* Folder
* Item Limit etc.



**Standard View:**  This view displays our list and library items one row following another. Standard view is the default for most types of lists and libraries. We can customize the view in many different ways, such as by adding or removing columns from the view.

**Calendar View:** This view displays our list and library in a format similar to a wall calendar. We can apply daily, weekly, or monthly views in this format. This view can be helpful if we want to see the items in the list or library chronologically. To use this view, list or library must contain columns with start dates and end dates for the calendar items.

**Datasheet View**: This view displays list and library items in a grid, similar to a spreadsheet. This view, also known as Quick Edit, can be helpful if we have to edit many items in a list or library at the same time. This view is also helpful if we want to export our data to a spreadsheet or database program. There are some limitations to Datasheet View - not all Excel functionality is available.

**Gantt View**: This view displays list and library items in bars that track progress, to see which tasks overlap each other and to visualize overall progress. To use this view, list or library must contain columns with start dates and end dates.

**SharePoint Library**

**Libraries:** SharePoint libraries are special types of lists that are created to store documents. Each file in a SharePoint document library is like one item. It also has columns or fields or properties.

SharePoint also provides various libraries for specific proposes like picture library, form library, etc.

* Different data can be stored like video, music, website, word file, pdf file etc,
* Version control is possible and important feature of SharePoint Library.

**Creation of SharePoint Library:**

Document library creation is same as List

Choose Site contents ->

Then select  + New -> Document Library.

Then Give name and Document library gets created.

In Document Library we can Store document like word file, excel file, music, video, website, link etc.

**-:Third Week :-**

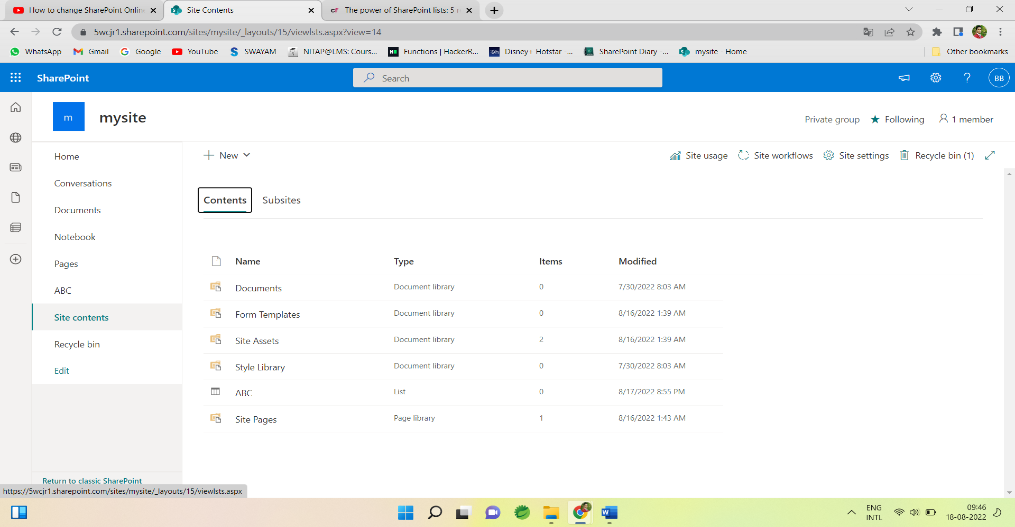
**-:Day 1st :-**

**SharePoint Predefined List Template:**

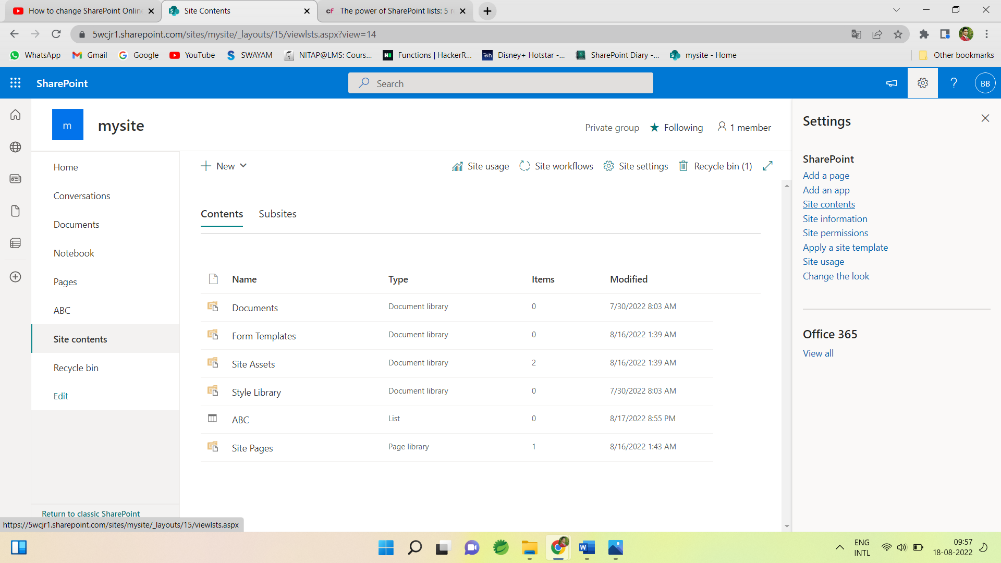
The SharePoint List Template is a great resource for developers who are looking to programmatically create new lists. Each template may represent a unique predefined set of settings, content types, columns, and views. Manually configuring a large quantity of these list settings may no longer needed when lists are created by using the appropriate SharePoint List Template.

**Create a SharePoint list using predefined templates:**

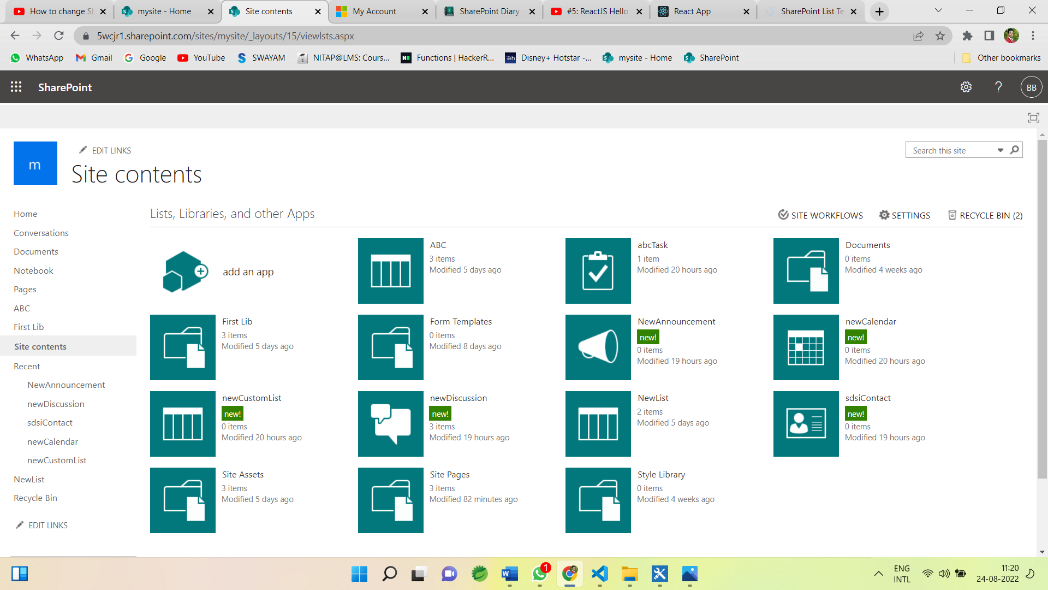
From SharePoint site home page



Or the Site contents page from gear icon,

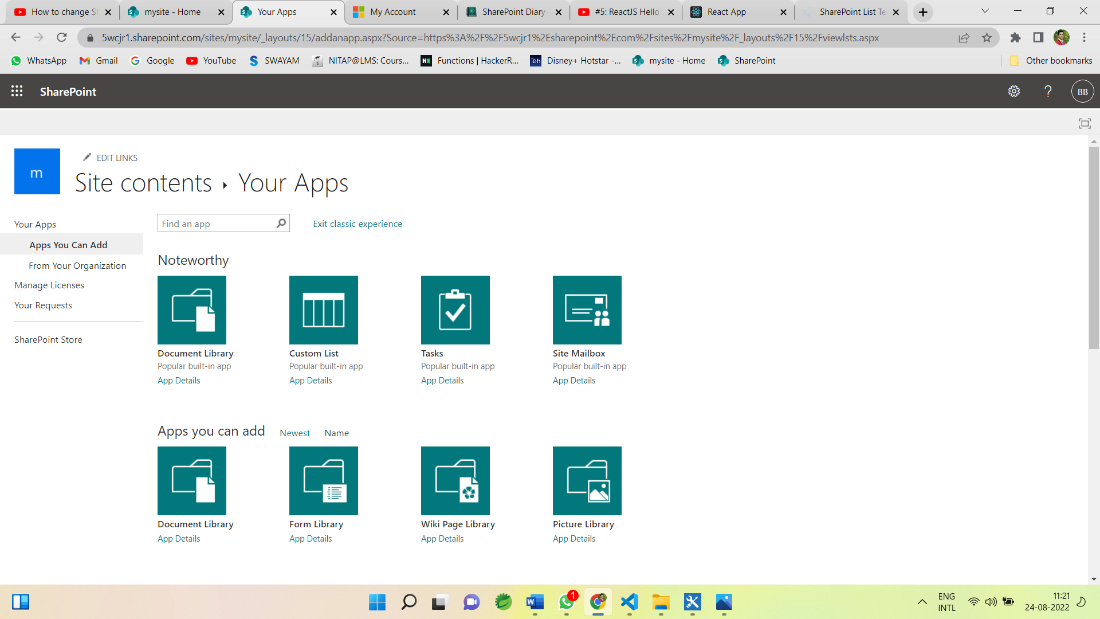


Then classic experience interface looks like



Then select ->  add an app

Now all the predefined list templates are visible



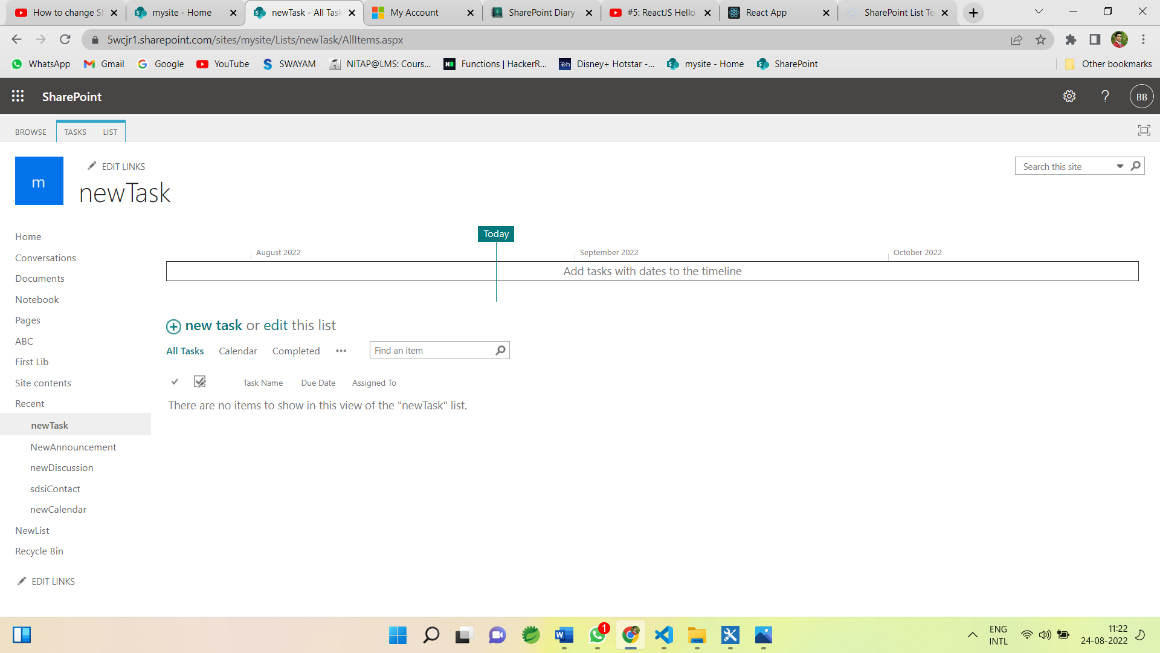


Now choose template according to the requirement.

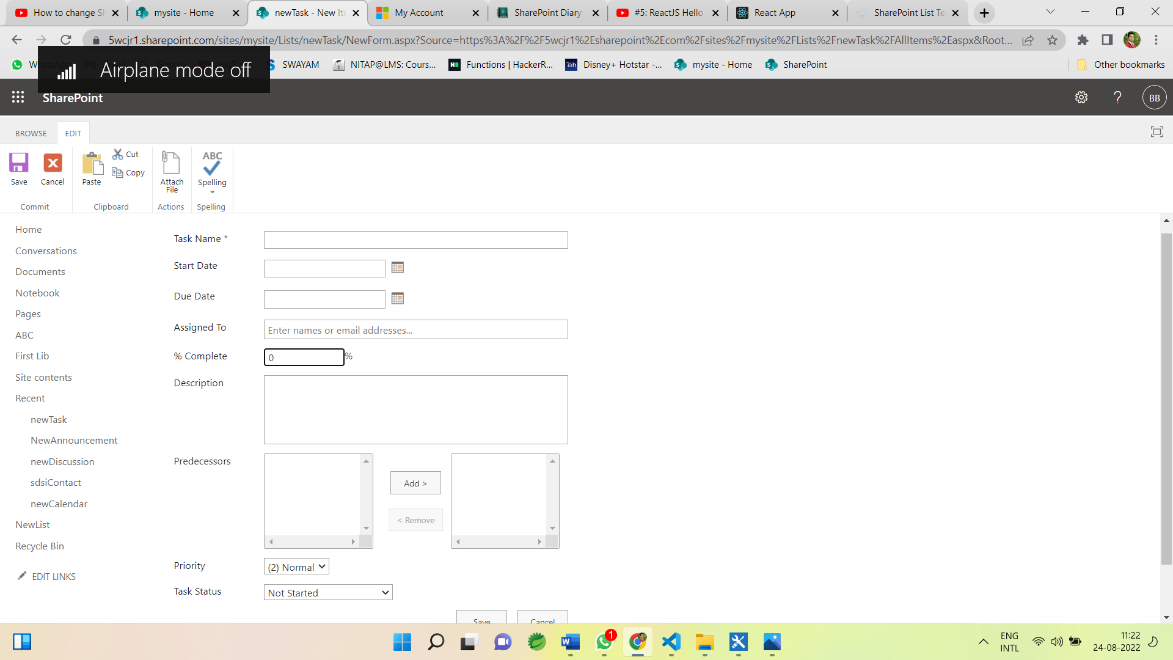
For example for Creating a task

Choose a task templete and give name

Then new task created with default field and views



Then click on + new task and its default fields open.



Now give the name and other fields.

Similarly, we can add other templates like contact, calendar, Discussion Board, Issue tracking, survey etc.

**-: Day 2nd :-**

**SharePoint Designer 2013:**

SharePoint Designer 2013 is the tool of choice for the rapid development of SharePoint applications. It is the tool of choice for the rapid development of SharePoint applications. Using SharePoint Designer, advanced users and developers alike can rapidly create SharePoint solutions in response to business needs.

Using this tool we can edit, customise the site we can write or edit code.

**SharePoint Workflow:**

* List Workflow
* Reusable Workflow
* Site Workflow

**Workflow Triger option:**

Trigger condition takes the format of an expression and must evaluate to either true or False. If the trigger condition is true then the flow will run, else it will ignore the trigger event.

Triger options are:

* Manually
* Automatic when condition satisfy.
* When an item is created and
* When item is changed.

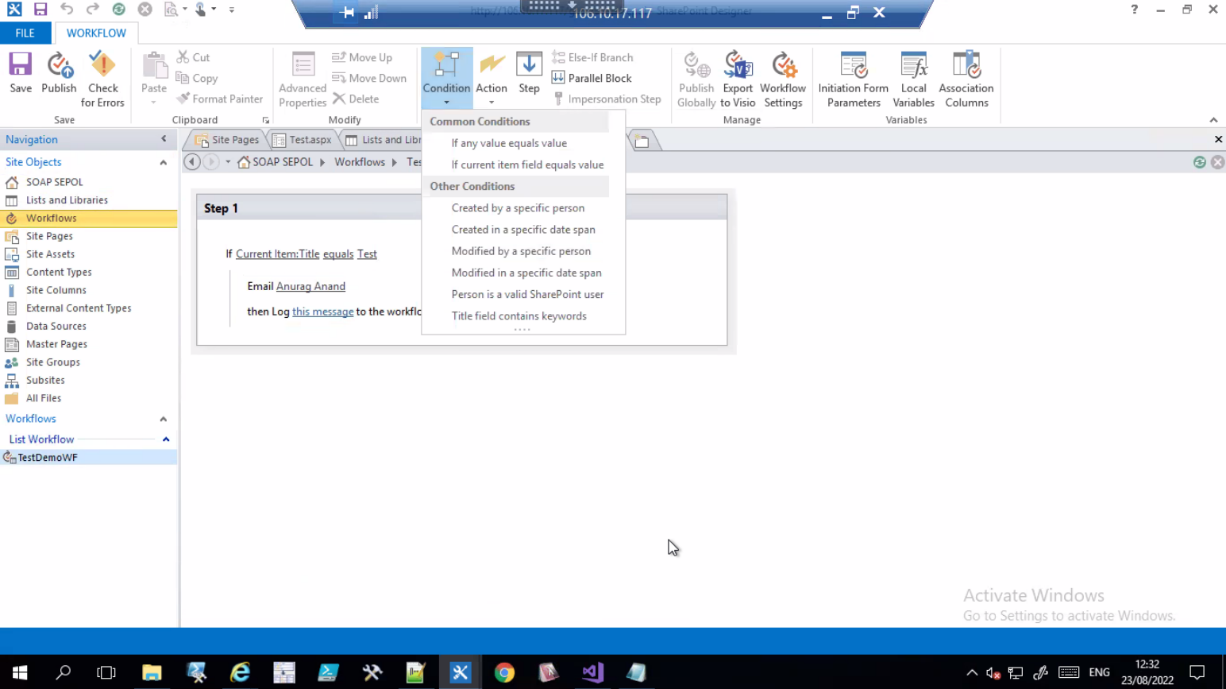
For example, if we use a trigger called ‘When an item is created or modified”, the flow will trigger with each update.

**Conditions of workflow:**

Workflow will be executed if the given condition satisfies then it executes the action inside the condition.

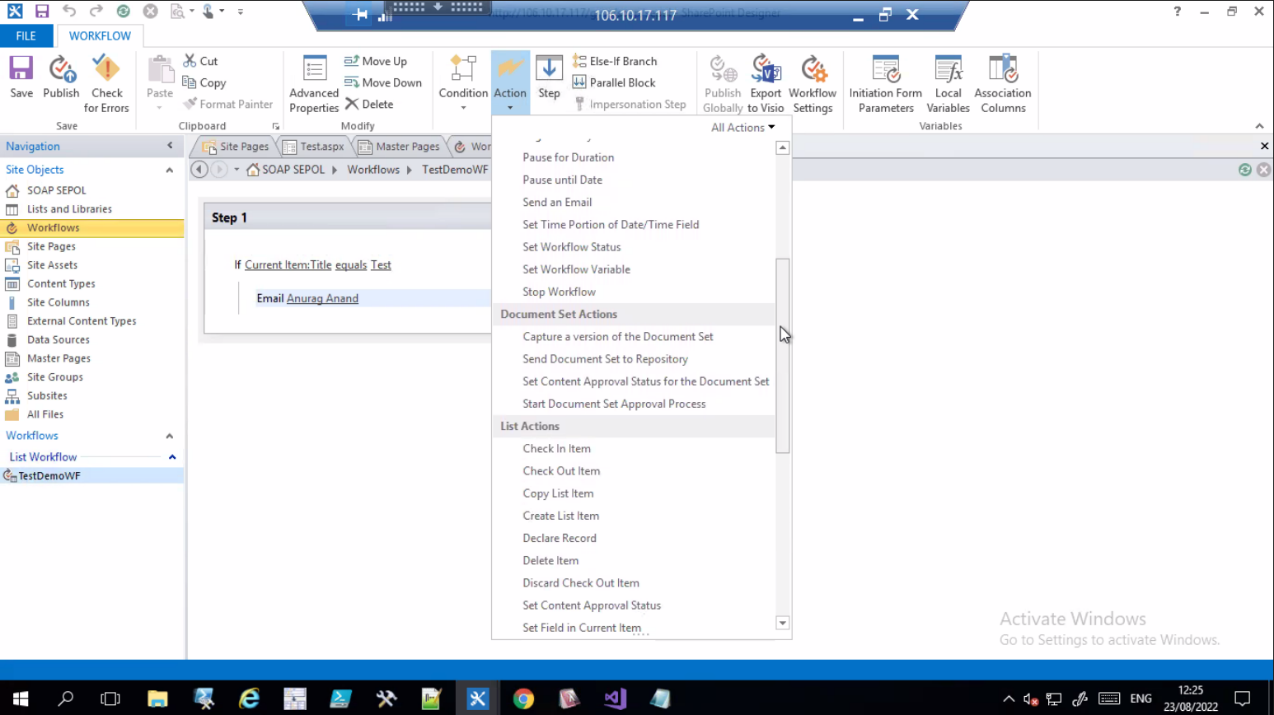
* Any value equals value
* Current item field equals value
* Created by specific person
* Created by specific date span etc.

We can apply workflow condition by choosing the suitable condition from condition tab:



Then Specify the task/action to be execute if the given condition satisfies from the action tab like:

* Pause for duration
* Pause until date
* Send an Email
* Set Workflow status
* Set Workflow variable
* Stop Workflow
* Check in Item
* Check Out Item etc.



**-: Day 3rd :-**

Today we had seen the Create operation of CRUD operation on SharePoint by coding.

**Create a HTML Form:**

Here we have a SharePoint list as “ComapnyInfoList” which has below columns:

* Username
* Password
* EmailID and
* gender

**HTML Code for form creation**

-----------------------------------------------------------------------------------------

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://onlysharepoint2013.sharepoint.com/sites/Raju/SiteAssets/Preetihtml/UserInfoJs.js"></script>

<meta charset="utf-8" />

<title>My Html</title>

</head>

<body>

<h2>Information Form</h2>

Username:<br>

<input type="text" name="Username" id="usrname" />

<br><br>

Password:<br>

<input type="password" name="password" id="usrpwd" />

<br><br>

EmailID:<br>

<input type="email" name="EmailID" id="usrid" />

<br><br>

Gender:<br>

<div id="gend">

<input type="radio" name="gender" value="Male" id="gen0" /> Male

<input type="radio" name="gender" value="Female" id="gen1" /> Female

</div>

<br><br>

<input type="submit" value="Submit" id="btnSubmit" />

<br>

<p id="pTitle"></p>

</body>

</html>

**JSOM Code for inserting data**

---------------------------------------------------------------------------------------------------

$(document).ready(function () {

$("#btnSubmit").click(function () {

insertitemtolist();

});

});

function insertitemtolist() {

var username = $("#usrname").val();

var password = $("#usrpwd").val();

var emailId = $("#usrid").val();

var gender = $("input[name='gender']:checked").val();

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CompanyInfoList');

var item = new SP.ListItemCreationInformation();

var oListItem = oList.addItem(item);

oListItem.set\_item('Title', username);

oListItem.set\_item('Password', password);

oListItem.set\_item('EmailID', emailId);

oListItem.set\_item('Gender', gender);

oListItem.update();

clientContext.load(oListItem);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded), Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded(sender, args) {

$("#pTitle").html("successfully executed");

}

function onQueryFailed(sender, args) {

alert('request failed ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

----------------------------------------------------------------------------------------------------

**-: Day 4th :-**

Today we had seen the Get/Read, Update and Delete operation of CRUD operation on SharePoint by coding.

**JSOM Code for updating data in CompanyInfoList**

---------------------------------------------------------------------------------------------------

$(document).ready(function () {

ExecuteOrDelayUntilScriptLoaded(retrieveListItems, "sp.js");

$("#btnUpdate").click(function () {

updateListItemByID();

});

});

var masterListItem;

function retrieveListItems() {

var id = GetParameterValues('MyID');

getitemsbyID(id);

}

function getitemsbyID(itemID) {

var clientContext = new SP.ClientContext.get\_current();

var masterlist = clientContext.get\_web().get\_lists().getByTitle('CmpanyInfoList');

masterListItem = masterlist.getItemById(itemID);

clientContext.load(masterListItem);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded),

Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded() {

$("#usrname").val(masterListItem.get\_item('Title'));

$("#usrpwd").val(masterListItem.get\_item('Password'));

$("#usrid").val(masterListItem.get\_item('EmailID'));

if (masterListItem.get\_item('Gender') == "Male") {

$("#gen0").attr('checked', 'checked');

}

else if (masterListItem.get\_item('Gender') == "Female") {

$("#gen1").attr('checked', 'checked');

}

}

function onQueryFailed(sender, args) {

alert('Request failed. \nError: ' + args.get\_message() + '\nStackTrace: ' + args.get\_stackTrace());

}

function GetParameterValues(param) {

var url = window.location.href.slice(window.location.href.indexOf('?') + 1).split('&');

for (var i = 0; i < url.length; i++) {

var urlparam = url[i].split('=');

if (urlparam[0] == param) {

return urlparam[1];

}

}

}

function updateListItemByID() {

var id = GetParameterValues('MyID');

var username = $("#usrname").val();

var password = $("#usrpwd").val();

var emailId = $("#usrid").val();

var gender = $("input[name='gender']:checked").val();

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CmpanyInfoList');

var oListItem = oList.getItemById(id);

oListItem.set\_item('Title', username);

oListItem.set\_item('Password', password);

oListItem.set\_item('EmailID', emailId);

oListItem.set\_item('Gender', gender);

oListItem.update();

clientContext.executeQueryAsync(Function.createDelegate(this, this.success), Function.createDelegate(this, this.failue));

}

function success() {

alert('Item updated!');

}

function failue(sender, args) {

alert('Request failed. ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

**Create HTML File for displaying the data:**

**-----------------------------------------------------------------------------------------**

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script> ----> // $ in Jsom

<script src="https://onlysharepoint2013.sharepoint.com/sites/Raju/SiteAssets/Preetihtml/UserInfoJs.js"></script> ---> SPServices in Jsom

<meta charset="utf-8" />

<title>ALL LIST ITEM</title>

</head>

<body>

<p id="disp"></p>

</body>

</html>

**JSOM Code for Reading/Getting the data From list “CompanyInfoList”**

**----------------------------------------------------------------------------------------**

$(document).ready(function () {

getdatafromlist();

});

function getdatafromlist(){

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CompanyInfoList');

var oListItems=oList.getItems(SP.CamlQuery.createAllItemsQuery()); ------>to etrieve all list items we use CamlQuery

clientContext.load(oListItem);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded),

Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded(sender, args) {

var newtitle='';

var listItemEnum=oListItems.getEnumerator(); -----> data is in the list enumerator will fetch one by one

while(listItemEnum.moveNext()){ ------>As long as the Enumerator will find next item this is will return true

var oListItem=listItemEnum.get\_current().get\_item('title');

newtitle+=oListItem;

}

$("#disp").append(newtitle);

}

function onQueryFailed(sender, args) {

alert('request failed ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

**JSOM Code for Deleting data From List “CompanyInfoList”**

**----------------------------------------------------------------------------------------**

$(document).ready(function () {

deleteitem();

});

function deleteitem() {

var dltItem=$("#dlttext").val();

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CompanyInfoList');

oListItem=oList.getItemById(dltItem); -------->Get the element by given ID

oListItem.deleteObject();

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded),

Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded(sender, args) {

$("#dltmsg").html("successfully deleted");

}

function onQueryFailed(sender, args) {

alert('request failed ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

**SharePoint Branding:**

SharePoint Branding is the process of changing the look and feel of the SharePoint user interface using custom master pages, style sheets, images, JavaScript, jQuery and so on.  
  
  
  
A few of the branded sites are shown below.  
  
  
  
**Advantages of Branding**  
  
A good branding leads to:

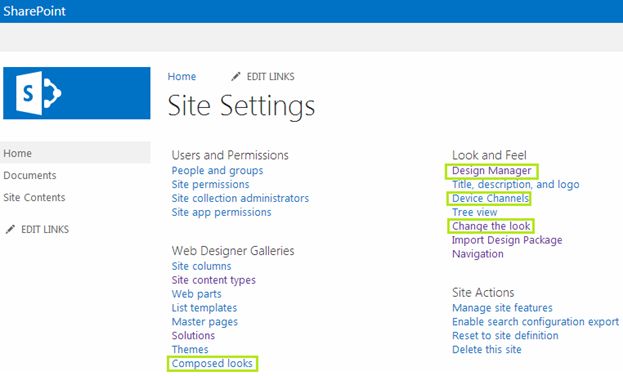
1. Better content presentation
2. Better navigation
3. Better look and feel
4. Enable responsive web design
5. Hide SharePoint default look

**Skills needed for Branding**

* HTML,
* CSS
* Basic Scripting skills and
* SharePoint Branding knowledge is an essential skill to integrate the custom branding artifacts.

**SharePoint Branding knowledge includes:**

* Master Page location
* Master Page elements like Site Settings, Quick Launch and so on.
* Composed looks

We can access these pages from Site Actions > Site Settings menu.  
  
  
  
**Note:**The Publishing feature needs to be activated for getting the Design Manager feature.  
  
**Branding Elements in SharePoint 2013**  
  
The following are the branding elements in SharePoint 2013:

1. Master Pages
2. Preview Files
3. CSS Files
4. Color Palettes
5. Font Palettes
6. Images

**-:Fourth Week :-**

**-:Day 1st :-**

**Security – User Group and Permissions:**

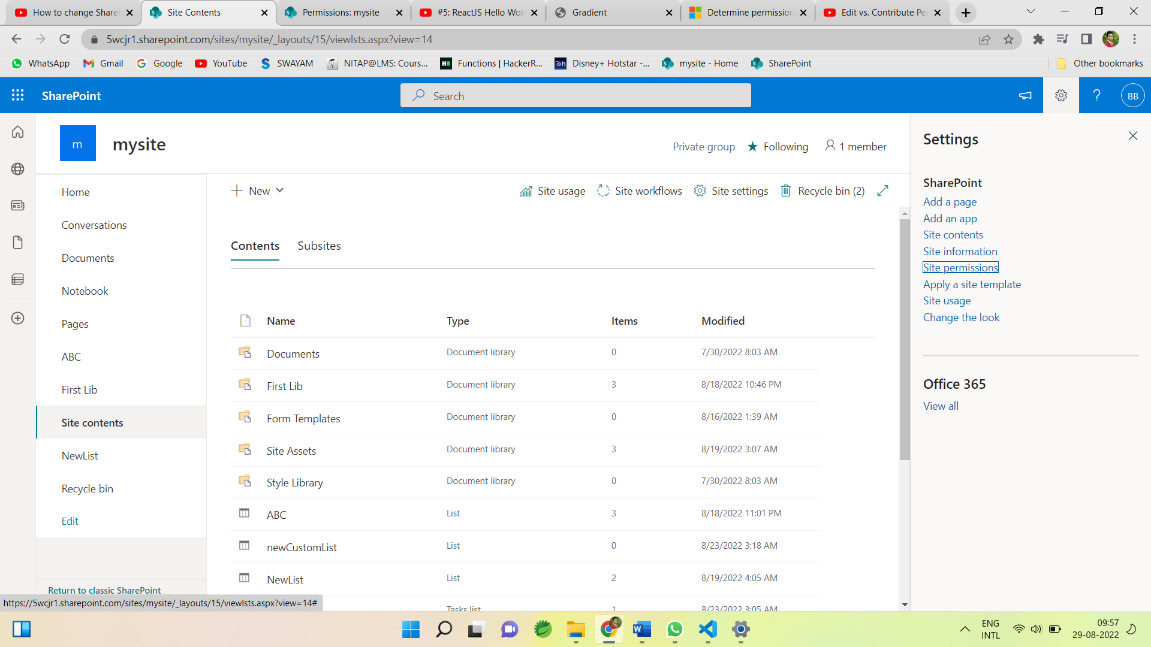
Some sites in an enterprise contain content that should not be available to all users. For example, proprietary technical information should be accessible only on a need-to-know basis. An intranet portal of company should be available to employees only, whereas the home page of an Internet Web site is accessible by anonymous clients.

By permissions control access to sites and site content we can manage permissions by using SharePoint groups, which control membership. Permissions also help to secure content at the item and document level.

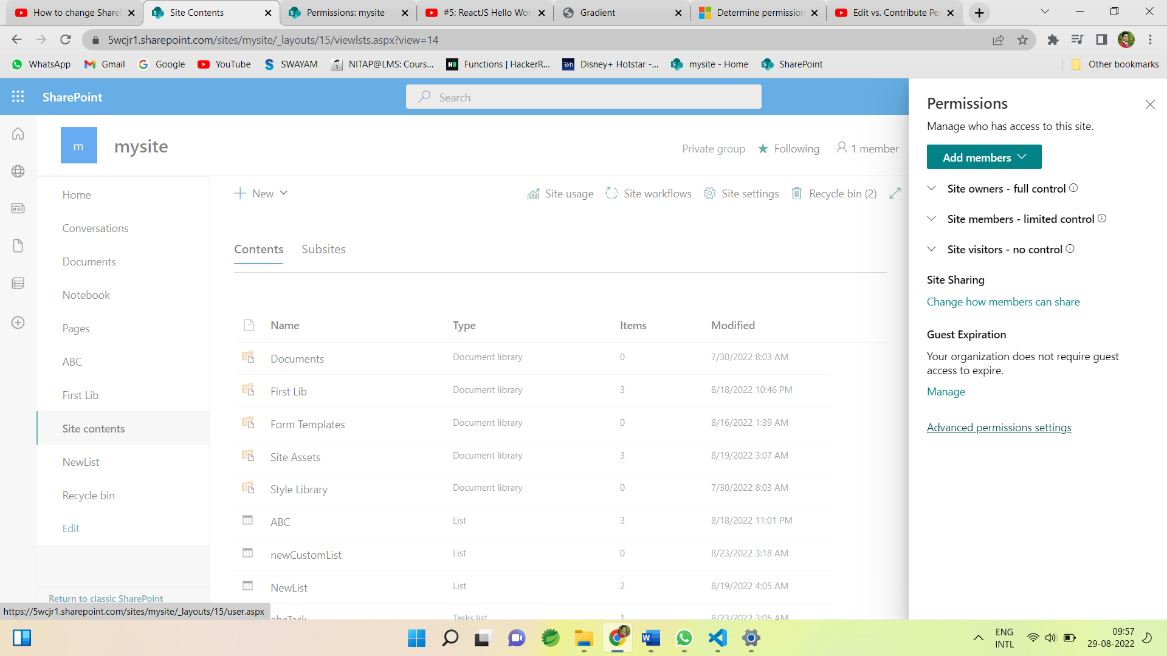
In SharePoint we can give permission by default permission levels of SharePoint or we can create our own Permission and specify the accessibility.

**SharePoint default permissions:**

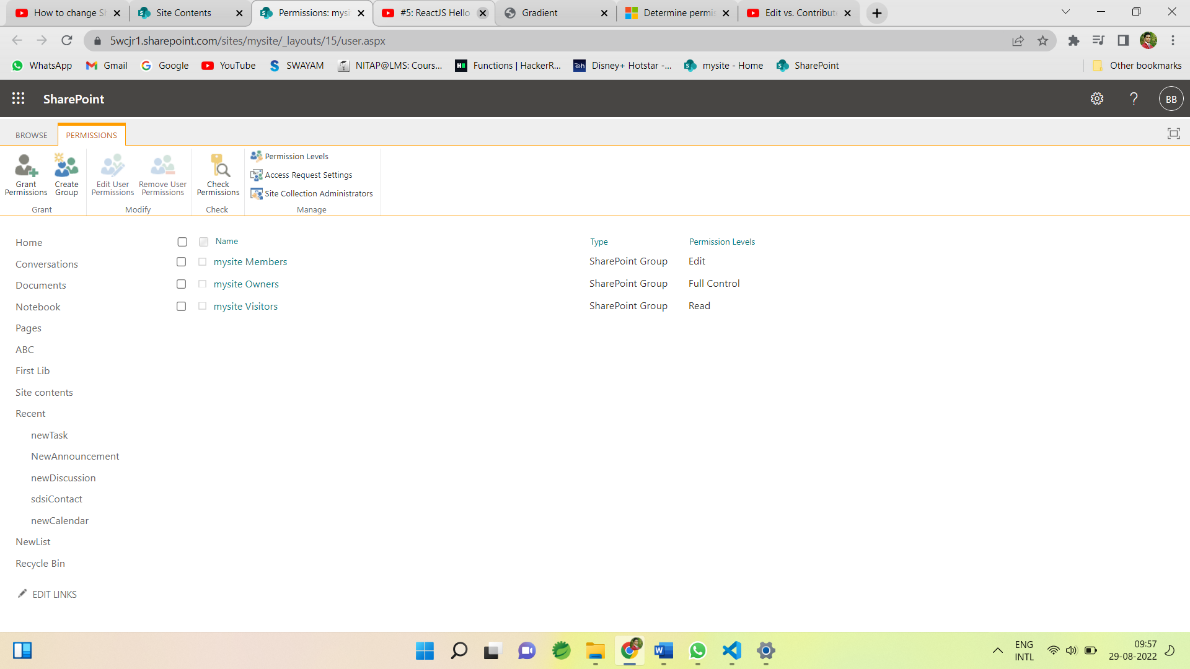
From SharePoint site home page clicking on the gear icon, we get an interface having 🡪 site permission



Clicking on Site permissions we get another tab of Permissions having Advance Permission setting



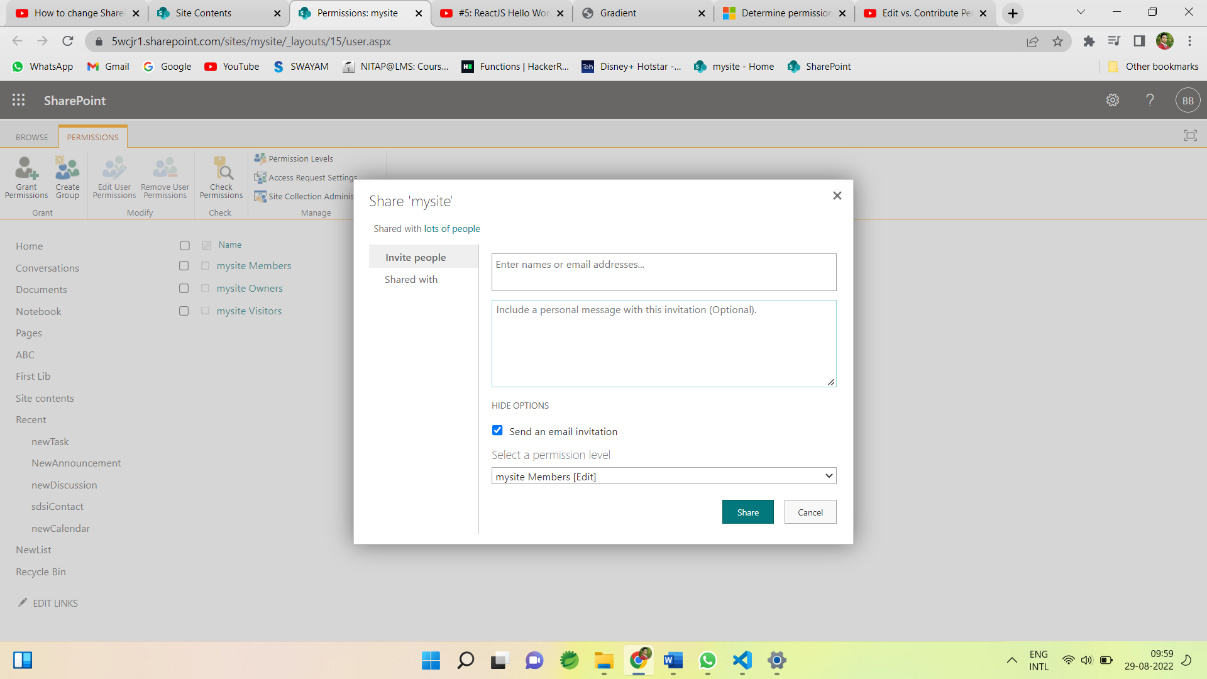
Click on Advance Permission setting we get all the Permissions Settings.



Here We can Create groups, Grant permissions, Edit user permissions, Remove user permissions, Check permissions, we can see permissions levels etc.

**Grant permissions :**

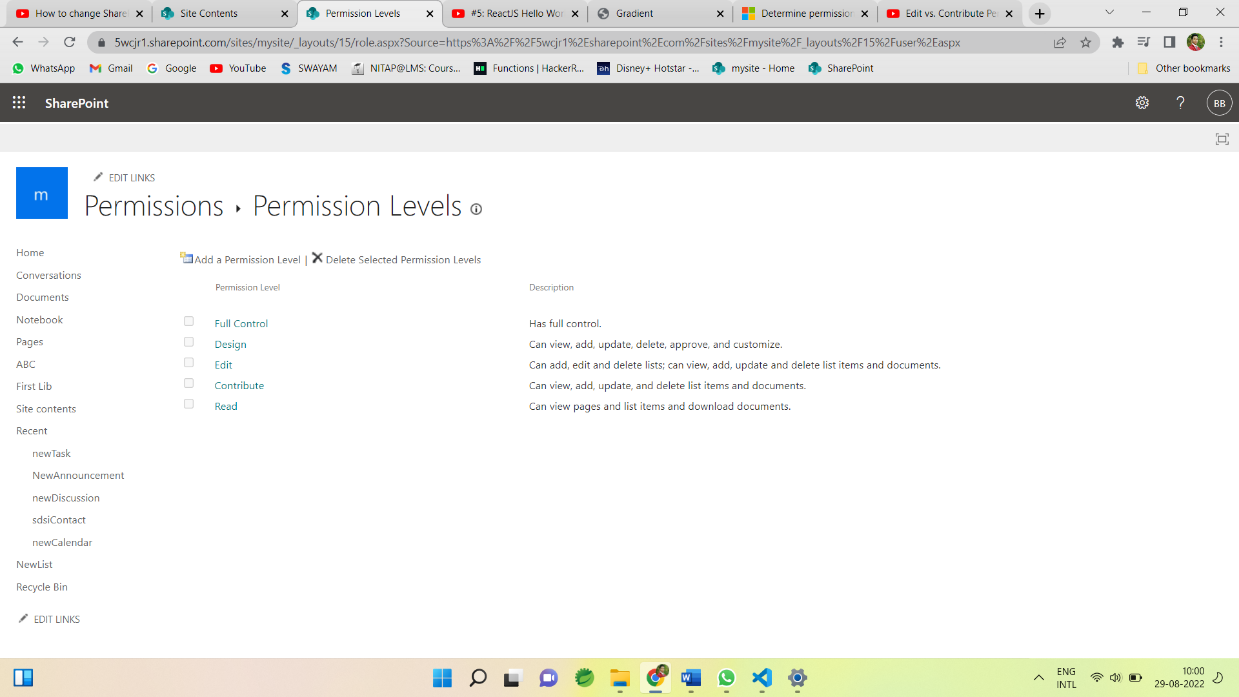
To grant permission click on Grant Permission tab and invite the person or group to give permission



Then this user is visible in the PERMISSIONS tab with specified permission level.

**Permissions levels :**

We can see the permissions levels on SharePoint through permission levels



Different permissions levels are:

* Full control: Grant full control to the users or groups.
* Design: Groups having permissions of design can view, add, update, delete, approve and customise.
* Edit: Groups having permissions of Edit can add, edit and delete lists; can view, add, update and delete list items and documents.
* Contribute: Groups having permissions of Contribute can view, add, update, and delete list items and documents, But can’t add, edit and delete lists;
* Read : Groups having permissions of Read can view pages and list items and download documents.

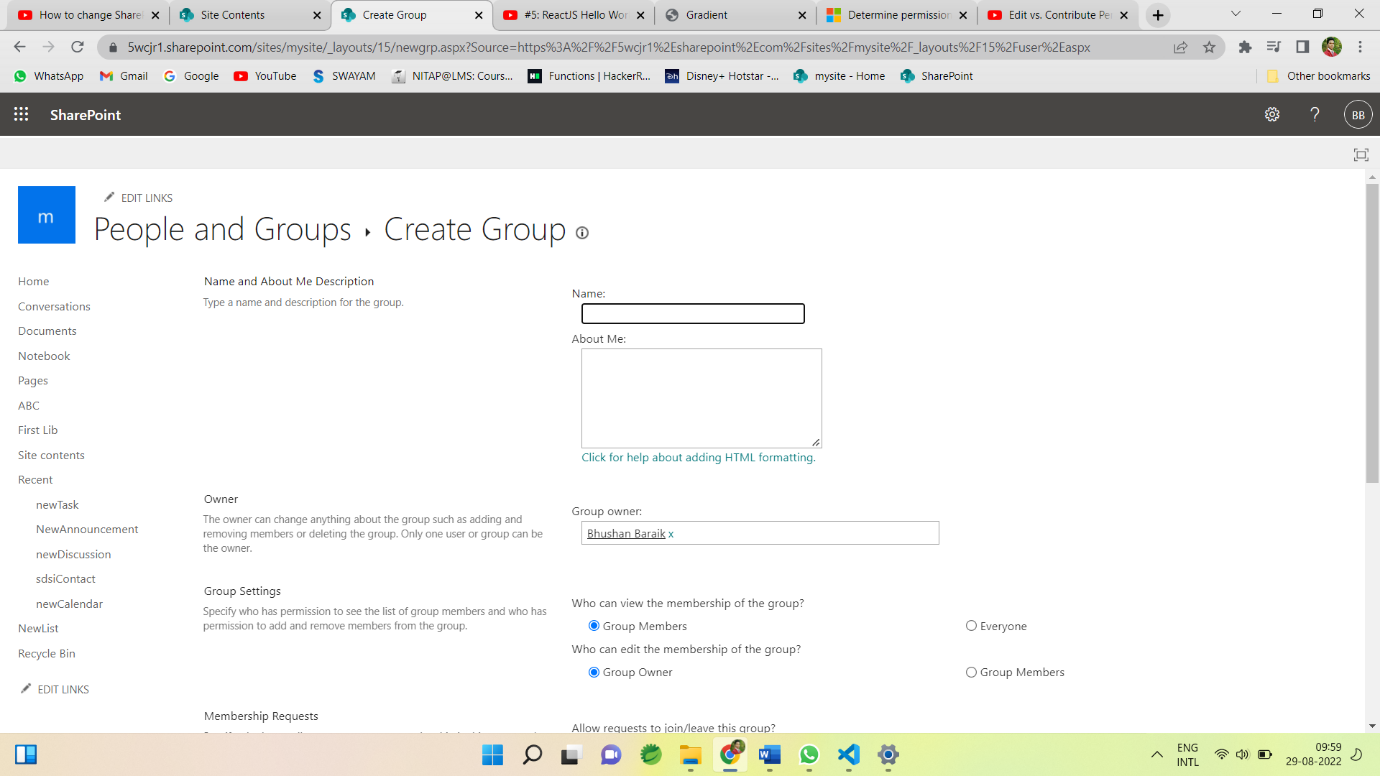
**Default Group with permissions :**

* Visitors 🡪Read
* Member 🡪Edit
* Owner 🡪 Full control
* Viewer 🡪view

**Create Group:**

We can create group to specify the permission to that group.

Provide the name to the groups 🡪



And specify the permission level for group 🡪



Security – User Group and Permissions is the most important part of SharePoint as SharePoint is a content sharing and management tool so its important to give the correct permissions to the different groups and users for the security of data/content.

**-:Day 2nd :-**

**Claim Based Authentications**

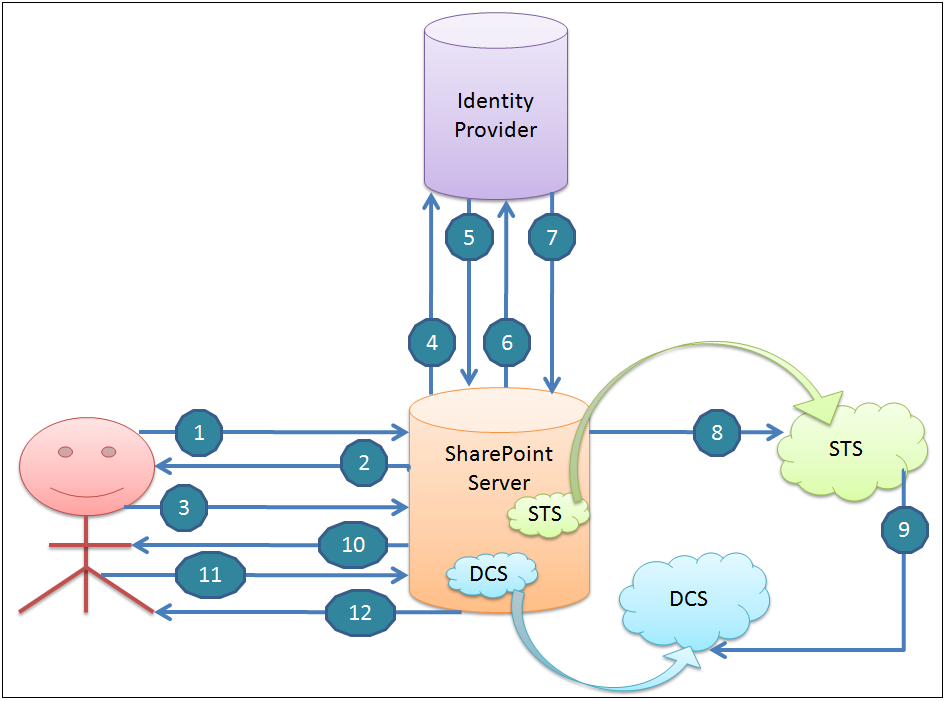
There are a few Authentication methods and we can use any of them.

1. Windows authentication
2. Forms-based authentication (FBA)
3. Security Assertion Markup Language (SAML) token-based authentication Windows Authentication:

In Windows authentication, SharePoint takes advantage of the default authentication provider (ADDS) to authenticate the logged-in user. In this type of authentication, SharePoint site takes the same credentials, which was used by the user to log in to his machine.

NTLM, Kerberos, Digest, Basic

**Windows claims Authentication process in SharePoint**



1. Assume, on clients computer meres no any clams-Based security token, user requests SharePoint resource (Site, list, page, etc....).
2. SharePoint asks the user for the credential.
3. The user provides the credentials.
4. SharePoint takes the credentials and goes to the identity provider to verify the credentials
5. The identity provider will verify the credentials and will send back the Windows Security Token
6. SharePoint asks Domain Controller for the list of Security Groups that the user belongs to.
7. Domain Controller sends the list of Security Groups t? SharePoint
8. Using this information (Windows Security Token and Group Membership of the user account), Security Token Service in SharePoint server creates claims-based Security Token.
9. Security Token Service (STS) stores claims in Distributed Cache Service (DCS) in ShareP0int farm.
10. IIS server on the SharePoint server sends authorization code to the client computer.
11. The end-user accesses the page if the user is authorized, through analysis of the claims in the security token and configured permission.
12. After authorization is successful, the SharePoint server sends the content of the page.

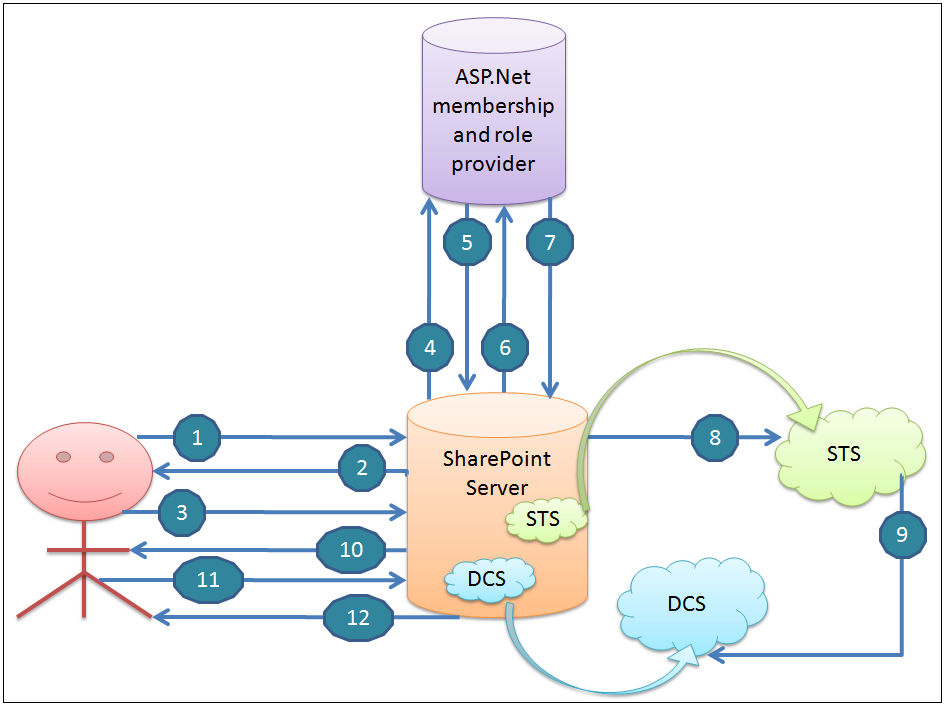
**Note:** Authorization code is used for authentication of any subsequent requests.

**Forms-Based Authentication (FBA)**

Form-based authentication is a process of checking the user's claim based identity with the help of ASP.Net membership and eole provider; You can use Forms-based authentication if the user credentials are stored in one of the below authentication providers

1. ADDS
2. SQL Server or equivalent database
3. Lightweight Directory Access Protocol (LDAP) Datastore

**Forms-based claims Authentication process in SharePoint**



1. Assume, on client's computer there's no any claims-based security token, User requests SharePoint resource (Site, list, page, etc....).
2. SharePoint sends Form-Based login page to the user and asks the user to enter credential.
3. The user provides the credentials.
4. SharePoint sends the credentials to Membership Provider.
5. Membership Provider verifies the credentials
6. SharePoint queries Role Provider for the credentials (user)
7. Role Provider sends the list of Roles that are associated with the user.
8. Based on Information from Membership and Role Provider, Security Token Service in SharePoint server creates claims-based Security Token.
9. Security Token Service stores claims in Distributed Cache Service (DCS) in SharePoint farm. Claims in Security Token include user identity and the roles of the user account.
10. IIS server on the SharePoint server sends Federated Authentication or Fed Cookie to the client machine. The cookie contains encrypted key or index to the security token (which is created in SharePoint server in Distributed Cache Service - DCS).
11. The end-user accesses the page, If the user is authorized, through analysis of the claims in the security token and configured permission.
12. After authorization is successful, the SharePoint server sends the content of the page. Fed Auth Cookie is used for authentication of any subsequent requests.

**Security Assertion Markup Language (SAML) token-based authentication**

SAML based authentication process includes:

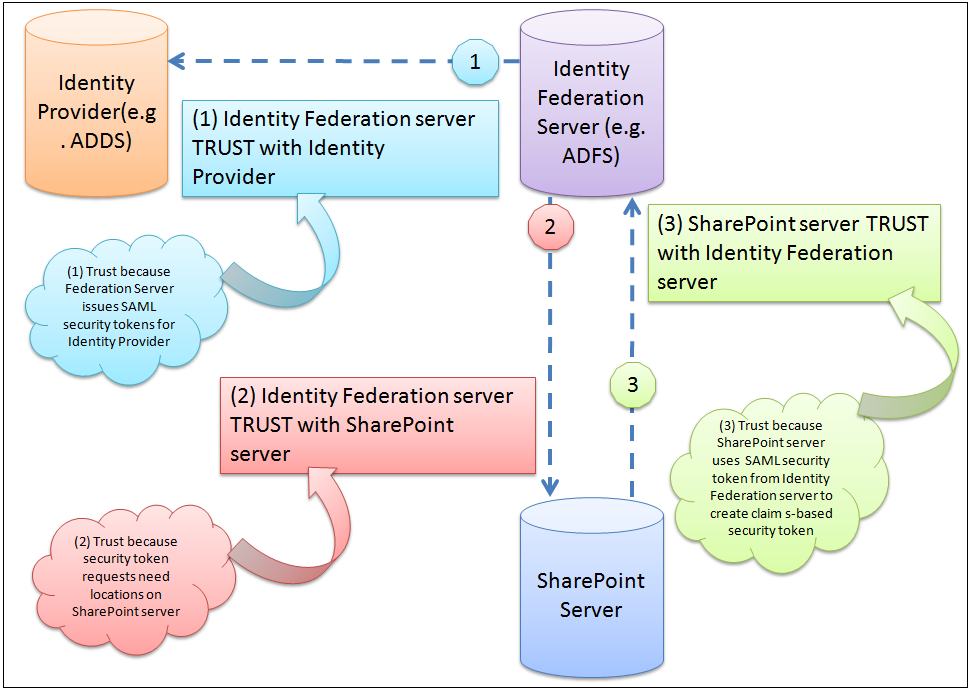
(1) User: The end-user who requests the content/data.

(2) SharePoint Server: The SharePoint server where the actual data is available.

(3) Identity Federation server such as ADFS (Active Directory Federation Services): ADFS is a Microsoft component developed for Single Sign-On Solution (SSO) especially when the users are not capable of accessing the application using Integrated Windows Identity (IWA) through Active Directory (AD).

(4) Identity provider - ADDS (Active Directory Domain Services): which contains id, password and other details of the objects/users.

**Trust Relationships - Security Assertion Markup Language (SAML) token-based authentication**

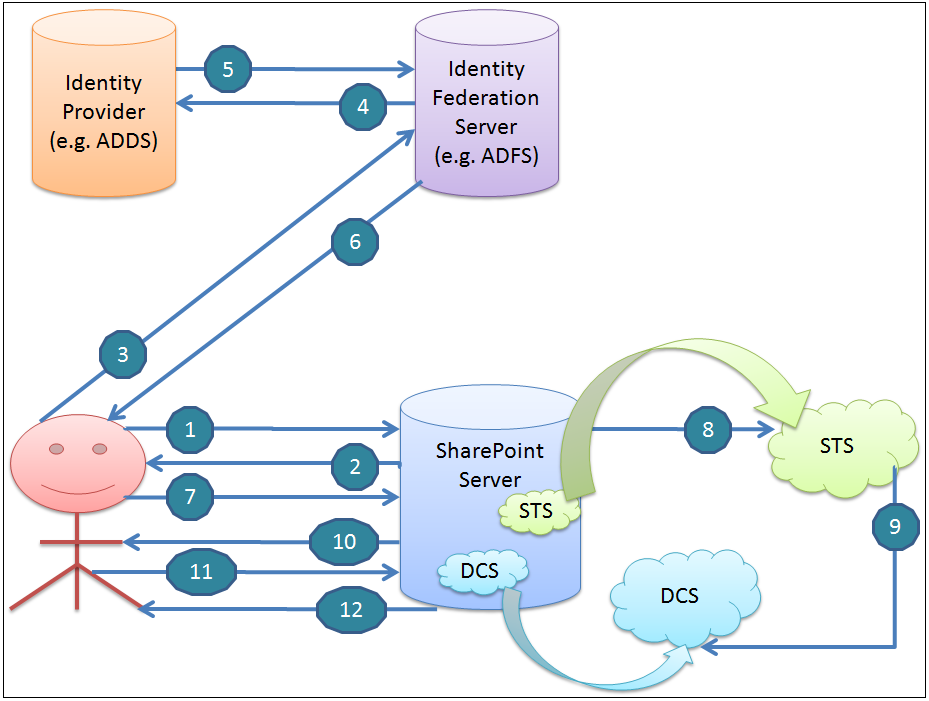


1. Identity Federation server (ADFS here) should establish TRUST with Identity Provider (ADDS here) for which ADFS is issuing SAML security tokens.
2. Identity Federation server (ADFS here) also needs to TRUST security token requests for locations on the SharePoint server.

For this, you need to configure ADFS with SharePoint Web Application URLs as Relying Parties. The idea behind this is to trust pages within those Web Application URLs for SAML security token requests.

1. SharePoint Server needs to TRUST the Identity Federation server (ADFS here).
   1. ADFS uses the certificate to sign the security token that it issues.
   2. SharePoint server needs a public key of that certificate to validate the digital signature on the security tokens issued by ADFS server.

**Security Assertion Markup Language (SAML) token-based authentication process**



1. Assume, on client's computer there's no any claims-based security token, User requests SharePoint resource (Site, list, page, etc....) and SAML Authentication process starts with this request.
2. SharePoint server redirects the user (client computer) to the Identity Federation server (ADFS here) server to obtain SAML-based login page
3. The user enters credentials in the login page and sends back to Identity Federation server (ADFS here) server and requests SAML security token
4. Identity Federation server (ADFS here) sends credentials to Identity Provider (ADDS here)
5. Identity Provider (ADDS here) confirms credentials and send a reply to Identity Federation server (ADFS here)
6. Identity Federation server (ADFS here) then creates a SAML token for the user, signs it and send it to the user.
7. The client computer sends a fresh request to SharePoint resource (Site, list, page, etc....). This time SAML security token is also included in the request that it got from the Identity Federation server (ADFS here).
8. Security Token Service in SharePoint server creates claims-based Security Token from SAML security token received from the Identity Federation server (ADFS here).
9. Security Token Service stores claims in Distributed Cache Service in SharePoint farm
10. IIS server on the SharePoint server sends Federated Authentication or ned Auth Cookie to the client machine. The cookie contains encrypted key or index to the security token (which is created in SharePoint server in Distributed Cache Service).
11. The end-user accesses the page if the user is authorized, through analysis of the claims in the security token and configured permission.
12. After authorization is successful, the SharePoint server sends the content of the page. Fed Auth Cookie is used for authentication of any subsequent requests.

**-:Day 3rd :-**

**Service Applications in SharePoint**

Services that are running in background for the user activities and operation.

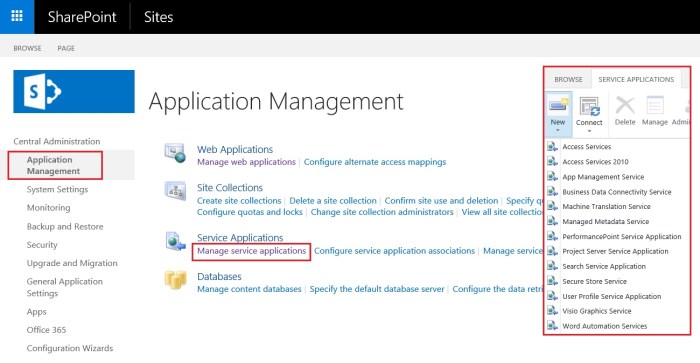
Service Applications in SharePoint are associated with web applications, and specific services are typically deployed as needed in a particular farm. Only deployed services are referred to as service applications. This is a huge advantage in terms of conserving resources and optimizing performance. For instance, a specific web application can be configured to use only the services it needs. The number of service applications that exist is vast, and, as third-party vendors can create their own services for SharePoint Server. A list of services includes the following:

* Access Services
* [APP Management Service](https://social.technet.microsoft.com/wiki/contents/articles/12516.sharepoint-2013-app-management-service.aspx)
* [Business Data Connectivity Service](https://docs.microsoft.com/en-us/sharepoint/administration/business-connectivity-services-overview)
* [Machine Translation Service](https://docs.microsoft.com/en-us/sharepoint/dev/general-development/machine-translation-services-in-sharepoint)
* [Managed Metadata Service](https://docs.microsoft.com/en-us/sharepoint/governance/managed-metadata-planning)
* [PerformancePoint Service Application](https://docs.microsoft.com/en-us/sharepoint/administration/performancepoint-services-overview)
* [Project Server Service Application](https://docs.microsoft.com/en-us/project/install-and-configure-project-server-2016)
* [Search Service Application](https://docs.microsoft.com/en-us/sharepoint/search/search)
* [Secure Store Service](https://docs.microsoft.com/en-us/sharepoint/administration/configure-the-secure-store-service)
* [User Profile Service Application](https://docs.microsoft.com/en-us/sharepoint/administration/user-profiles-and-identities)
* Visio Graphics Service
* Word Automation Services

We can set up a single service application to be shared among multiple web applications or deploy multiple instances of the same service across a farm and, in some cases, across multiple farms. Also, there is no limitation regarding the number of services that can be deployed in any single farm.

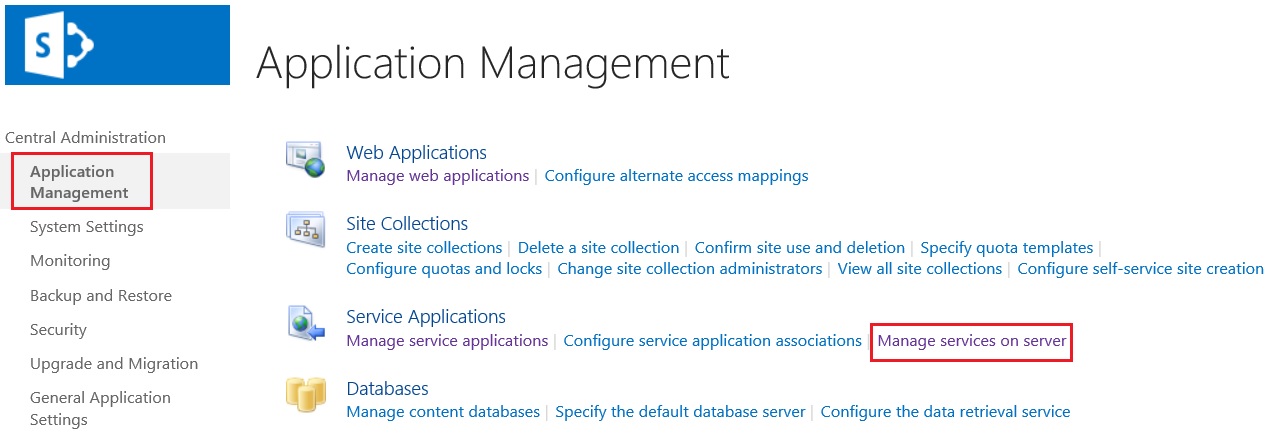
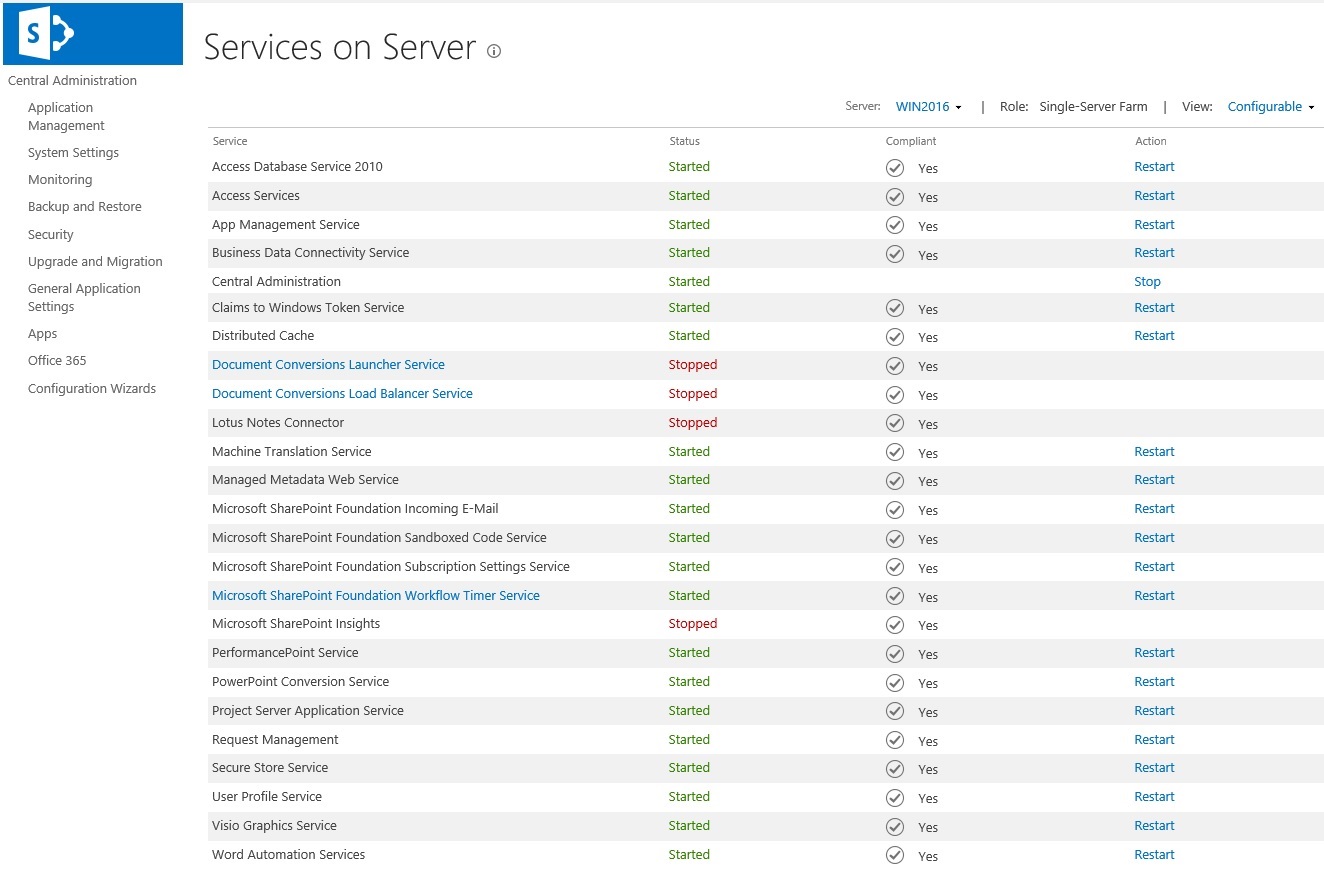
* Business Data Connectivity
* Machine Translation
* Managed Metadata
* User Profile
* Search
* Secure Store

Navigate to “Central Admin -> Application Management ->Manage Service Applications” and click on “New” from ribbon to see the list of service applications available.

[](https://i0.wp.com/sharepointtechnicalsupport.com/wp-content/uploads/2018/10/Service-Applications-in-SharePoint-1300x674.jpg?ssl=1)

## SharePoint services 2016

For every service application there is a service associated with it which we can from central admin by navigating to “Central Admin -> Application Management -> Manage Services on Server”.

[](https://i0.wp.com/sharepointtechnicalsupport.com/wp-content/uploads/2018/10/manage-services-on-server-1273x441.jpg?fit=1273%2C441&ssl=1)[](https://i1.wp.com/sharepointtechnicalsupport.com/wp-content/uploads/2018/10/services-on-server-1324x870.jpg?fit=1324%2C870&ssl=1)

SharePoint Server includes a set of service applications that we can use to share services across web applications. In some cases, we can also share service applications across farms. We can manage service applications by using the SharePoint Central Administration website or by using Windows PowerShell 3.0.

**-:Day 4th :-**

**Introduction to PowerShell SharePoint**

PowerShell is a command line scripting language and we can use PowerShell in SharePoint which will provide administrators to give full access to SharePoint objects.

By using PowerShell in SharePoint, Administrators can work directly with SharePoint objects like web application, site collection, sites/subsites, list, libraries, field/columns, etc.

In a SharePoint server, all PowerShell SharePoint cmdlets are available in Microsoft.SharePoint.PowerShell.dll

**Some Useful PowerShell SharePoint Commands:-**

**1. Create site collection using PowerShell in SharePoint:-**

Add-PSSnapin Microsoft.SharePoint.PowerShell -ErrorAction SilentlyContinue

**#Configuration Variables**

$SiteCollURL = "https://www.enjoysharepoint.com/sites/RajDemo"

$SiteName = "Welcome To Enjo SharePoint"

$SiteOwner = "HQAdmin\rswain"

$SiteTemplate = "STS#0" #Team Site Template

**#Create new Site Collection**

New-SPSite -URL $SiteCollURL -OwnerAlias $SiteOwner -Template $SiteTemplate -Name $SiteName

**2. Delete site collection PowerShell:-**

Remove-SPSite -Identity "https://www.enjoysharepoint.com/sites/RajDemo"

**3.Get all lists in a SharePoint Site using PowerShell:-**

Add-PSSnapin Microsoft.SharePoint.PowerShell -ErrorAction SilentlyContinue

$Site = Get-SPSite "https://www.enjoysharepoint.com/sites/RajDemo"

# get the all sub sites of site

$SubSites = $Site.AllWebs

$SubSites | ForEach-Object {

$Site = $\_

# get all lists from site

$lists = $Site.Lists | Where-Object { $\_.BaseType -eq 'GenericList' }

$lists | ForEach-Object {

New-Object -TypeName PSObject -Property @{

ListName = $\_.Title

SiteName = $Site.Title

SiteUrl = $Site.Url

}}}

**4.Get all libraries in a SharePoint Site using PowerShell:-**

Add-PSSnapin Microsoft.SharePoint.PowerShell -ErrorAction SilentlyContinue

$Site = Get-SPSite "https://www.enjoysharepoint.com/sites/RajDemo"

# get the all sub sites of site

$SubSites = $Site.AllWebs

$SubSites | ForEach-Object {

$Site = $\_

# get all document Libraries from site

$lists = $Site.Lists | Where-Object { $\_.BaseType -eq 'DocumentLibrary' }

$lists | ForEach-Object {

New-Object -TypeName PSObject -Property @{

LibraryName = $\_.Title

SiteName = $Site.Title

SiteUrl = $Site.Url

}}}

**5.Create a List using PowerShell in SharePoint:-**

Add-PSSnapin 'Microsoft.SharePoint.PowerShell' -ErrorAction SilentlyContinue

$SPweb = Get-SPWeb -Identity 'https://www.enjoysharepoint.com/sites/RajDemo'

$ListTemplate = $SPweb.ListTemplates['Custom List']

$SPweb.Lists.Add('SharePoint', 'List creation demo using PowerShell' , $ListTemplate)

**6.Delete List in SharePoint using PowerShell-**

Add-PSSnapin 'Microsoft.SharePoint.PowerShell' -ErrorAction SilentlyContinue

$SPweb = Get-SPWeb -Identity 'https://www.enjoysharepoint.com/sites/RajDemo'

$list = $SPweb.Lists['SharePoint']

$item1 = $list.Items

$item1[0].Delete()

**7.PowerShell add item to SharePoint list:-**

Add-PSSnapin 'Microsoft.SharePoint.PowerShell' -ErrorAction SilentlyContinue

$SPweb = Get-SPWeb -Identity 'https://www.enjoysharepoint.com/sites/RajDemo'

$list = $SPweb.Lists['SharePoint']

$item1 = $list.Items.Add()

$item1['Title'] = 'Chendrayan'

$item1.update()

**8.Delete Item in a specific list in SharePoint using PowerShell:-**

System.Reflection.Assembly]::LoadWithPartialName(“Microsoft.SharePoint”)

$site = new-object Microsoft.SharePoint.SPSite("https://www.enjoysharepoint.com/sites/RajDemo")

$relweburl = ”/rajtest”

$web = $site.openweb($relweburl)

$list = $web.Lists[“RajList”]

$listItems = $list.Items

$listItemsTotal = $listItems.Count

for ($item=$listItemsTotal-1;$item -ge 0; $item–)

{

Write-Host(“DELETED: ” )

$listItems[$item].Delete()

}}

**-:Day 5th :-**

**SharePoint Server Object Model**

There are two types of object model in SharePoint.

* Server object model
* Client object model

SharePoint server object model code will run in the SharePoint server (where SharePoint is installed).

Apart from visual web parts, custom features, etc, we can also develop client applications like a console application, windows applications, or Asp.Net web-based applications which will run on a SharePoint server.

To work with the SharePoint server object model, we need to add Microsoft.SharePoint.dll.

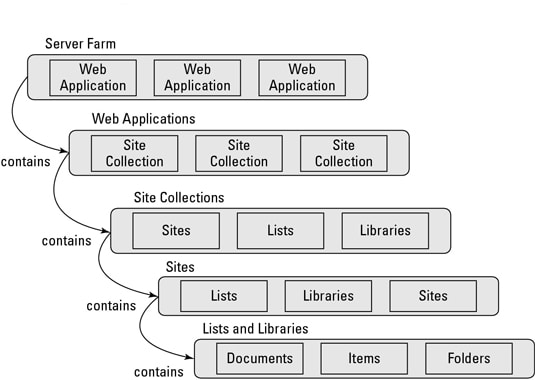
Microsoft provides Microsoft.SharePoint namespace to work with the top-level site, site, subsite or list, etc. The dll is located in the below directory.

Below are various SharePoint components, classes, and namespaces.

|  |  |  |
| --- | --- | --- |
| SharePoint Components | Server Object Model Classes | Namespace |
| Farm | SPFarm | Microsoft.SharePoint.Administration |
| Server | SPServer | Microsoft.SharePoint.Administration |
| Web Application | SPWebAplication | Microsoft.SharePoint.Administration |
| Content Database | SPContentDatabase | Microsoft.SharePoint.Administration |
| Site Collection | SPSite | Microsoft.SharePoint |
| Site | SPWeb | Microsoft.SharePoint |
| List/Library | SPList | Microsoft.SharePoint |
| ListItem | SPListItem | Microsoft.SharePoint |

## SharePoint server object model classes

Let us first discuss the various hierarchy of SharePoint server object model classes. All the server object classes are derived from Microsoft.SharePoint namespace and the administration classes inside the Microsoft.SharePoint.Administration namespace.



SharePoint server object model examples

**SPFarm:**  
This reference the entire SharePoint Server Farm.

[SharePoint server farm](https://www.spguides.com/sharepoint-2013-server-farm-architecture/) can be small, medium or large based on the various factors like organizations infrastructure, number of web applications, number of users accessing it, etc. The smallest server farm consists of a database server running on Microsoft SQL Server, one or more servers running IIS and office SharePoint server.

**SPServer:**  
By using this class we can browse through the collections of servers belongs to the Farm.

**SPWebApplication:**

[SharePoint web application](https://www.spguides.com/create-sharepoint-web-application/) is similar to that of the IIS web site. When a web application is created using the “Create or Extend Web Application” link in “Application Management” it creates a content database for the site collection, creates an IIS website, and the virtual directories.

**SPSite:**  
Represent a Site collection.

**SPWeb:**  
Represent a web [site](https://www.spguides.com/sharepoint-create-subsite/).

**SPUserToken:**  
The SPUserToken class represents a token for a valid SharePoint user.

**SPList:**  
SPList corresponds to a single list instance, whether that is a list of items or a document library.

**SPListItem:**  
This defines a reference to a specific item of a list.

**SPDocumentLibrary:**  
This type represents a document library.

**SPFile:**  
This class is used to enumerate the files contained in a document library.

**SPPrincipal:**  
This class is the parent class for SPGroup and SPUser.

**SPControl:**  
This class we need while developing web controls or Web Parts.

**SPContext:**  
This is a very useful class and it has some direct methods to access useful information about current requests.

**RunWithElevatedPrivileges in SharePoint**

SPSecurity.RunWithElevatedPrivileges method executes the specified method with Full Control rights even if the user does not otherwise have Full Control.

**Syntax:**

SPSecurity.RunWithElevatedPrivileges(

delegate()

{

// Code will go where

}

);

Below are the controls we will use to develop the form.

* **TextBox:** This control helps to create a text box in the input design form.
* **Button:** This control helps to create a button. So that when a user click on button, the document library will create in the SharePoint 2016 Site.
* **Label:** If some error will come after deploying the code, then this control helps to show the error message.

Server Object Model code to create SharePoint 2016 Document Library programmatically.

using Microsoft.SharePoint;

using System;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

namespace Document\_Library\_Creation.Document\_Library\_Creation

{

public partial class Document\_Library\_CreationUserControl : UserControl

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

}

}

protected void BtnCreateDocumentLibrary\_Click(object sender, EventArgs e)

{

CreateDocumentLibrary();

}

public void CreateDocumentLibrary()

{

try

{

SPSite siteCollection = SPContext.Current.Site;

SPWeb web = SPContext.Current.Web;

web.AllowUnsafeUpdates = true;

SPList spList = web.Lists.TryGetList(txtDocumentLibraryName.Text.Trim());

if (spList == null)

{

web.Lists.Add(txtDocumentLibraryName.Text.Trim(), txtDescription.Text.Trim(), SPListTemplateType.DocumentLibrary);

//SPList newList = web.Lists["TestList"];

web.AllowUnsafeUpdates = false;

CreateColumn(web);

lblMessage.Text = "Your document library created successfully";

}

else

{

lblMessage.Text = "This document library already exists";

}

}

catch (Exception ex)

{

lblMessage.Text = ex.Message;

}

}

}

}

Below SharePoint 2016 server object model Code, we can use for creating column (Single line of text and Choice column) in SharePoint 2016 Document Library programmatically.

string libraryDocumentLocationColumnName = "DocumentLocation";

string libraryLanguageColumnName = "DocumentLanguage";

string languageChoiceColumn1 = "English";

string languageChoiceColumn2 = "Hindi";

string languageChoiceColumn3 = "Kannad";

string libraryDepartmentColumnName = "DocumentDepartment";

string departmentChoiceColumn1 = "IT";

string departmentChoiceColumn2 = "HR";

string departmentChoiceColumn3 = "Finance";

public void CreateColumn(SPWeb web)

{

try

{

web.AllowUnsafeUpdates = true;

SPList spList = web.Lists.TryGetList

(txtDocumentLibraryName.Text.Trim());

spList.Fields.Add

(libraryDocumentLocationColumnName, SPFieldType.Text, false);

spList.Fields.Add(libraryLanguageColumnName,

SPFieldType.Choice, false);

spList.Fields.Add(libraryDepartmentColumnName, SPFieldType.Choice, false);

/\* get the newly added choice field instance \*/

SPFieldChoice chFldDocLanguage =

(SPFieldChoice)spList.Fields[libraryLanguageColumnName];

/\* set field format type i.e. radio / dropdown \*/

chFldDocLanguage.EditFormat =

SPChoiceFormatType.Dropdown;

/\* set the choice strings and update the field \*/

chFldDocLanguage.Choices.Add

(languageChoiceColumn1);

chFldDocLanguage.Choices.Add

(languageChoiceColumn2);

chFldDocLanguage.Choices.Add

(languageChoiceColumn3);

chFldDocLanguage.Update();

/\* get the newly added choice field instance \*/

SPFieldChoice chFldDocDepartment =

(SPFieldChoice)spList.Fields[libraryDepartmentColumnName];

/\* set field format type i.e. radio / dropdown \*/

chFldDocDepartment.EditFormat =

SPChoiceFormatType.Dropdown;

/\* set the choice strings and update the field \*/

chFldDocDepartment.Choices.Add

(departmentChoiceColumn1);

chFldDocDepartment.Choices.Add

(departmentChoiceColumn2);

chFldDocDepartment.Choices.Add

(departmentChoiceColumn3);

chFldDocDepartment.Update();

spList.Update();

web.AllowUnsafeUpdates = false;

}

catch (Exception exc)

{

lblMessage.Text = exc.Message;

}

}

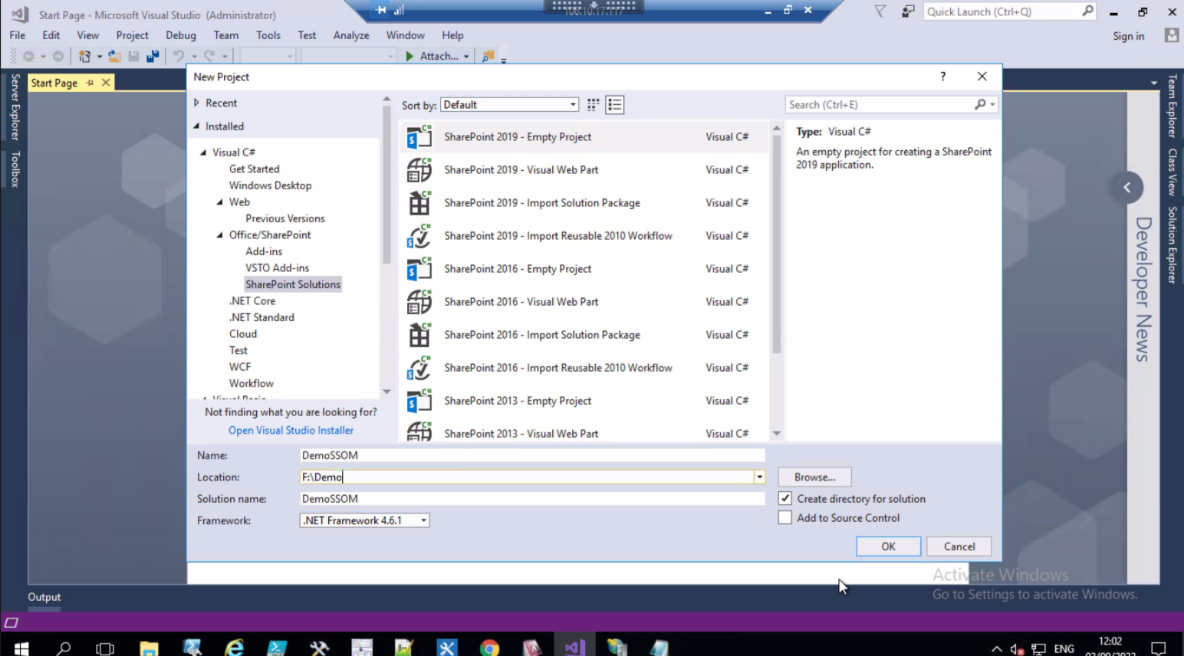
**-:Fifth Week :-**

**-:Day 1st :-**

**Project development in Visual Studio**

We had seen the development of SharePoint web application development in visual studio:

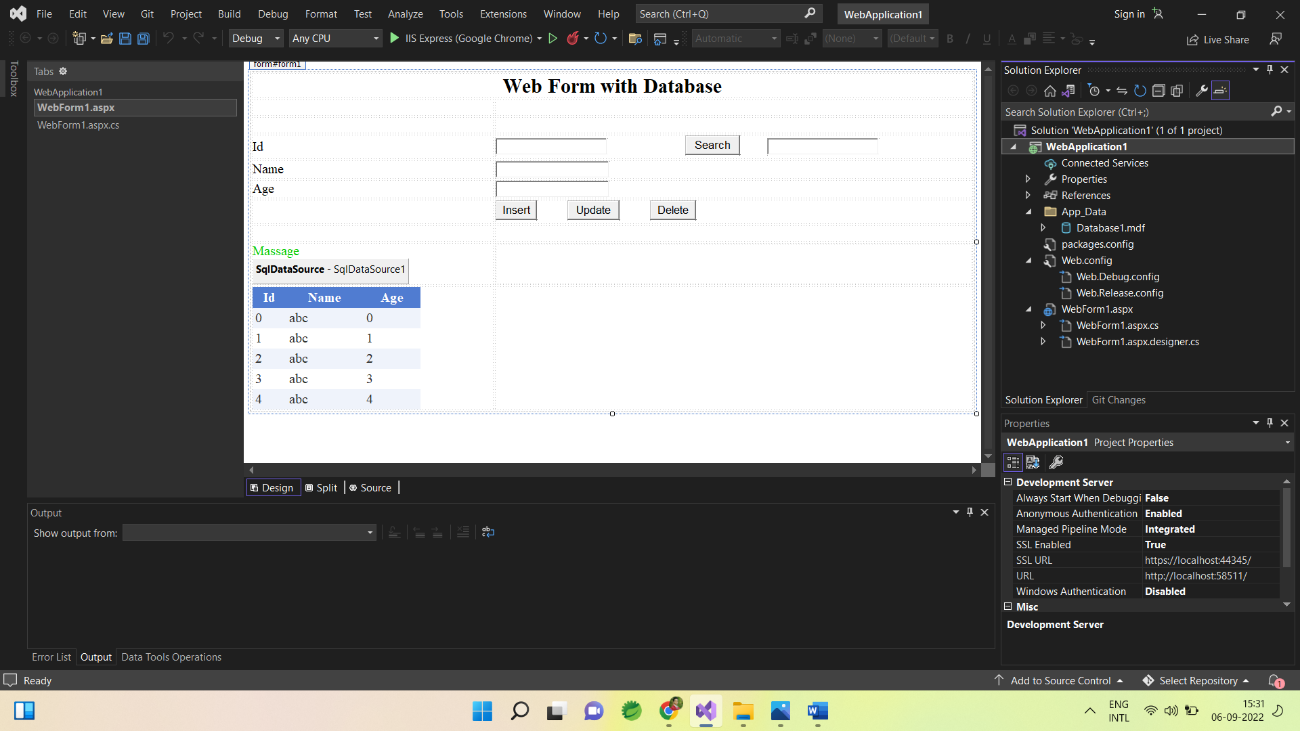
1. Visual studio interface 🡪 We had seen the UI of Visual Studio, Steps to create Project. Add features and many more.
2. Creating SharePoint application 🡪 Creating SharePoint application through SharePoint solution explorer



Like SharePoint Empty project, Visual Web part, web part, Event Receiver, Import Solution Package etc.

Use of toolbox 🡪 Toolbox can help us on creating various ASP.NET controls like button, textbox etc.

How to design UI 🡪Learnt designing user Interface using various tools and feature of Visual studio.

Folder structure of project 🡪 When project created it create some default folder containing project files like Web Config, WebForm1.aspx which contain WebForm1.aspx.cs and WebForm1.aspx.designer.cs

Design file (User Interface file 🡪.aspx)🡪 File with .aspx contain designer code of the project. It may have business logic in case of web site but not in case of web application.

Business logic file (CodeBehind File 🡪.aspx.cs) 🡪 CodeBehind file is the business logic file or backend code file which contain function and methods of the task and action performed in UI parts.

Publish and upload 🡪 Learnt the difference between publish and upload.

**-:Day 2nd :-**

## CAML Query in SharePoint

CAML stands for Collaborative Application Markup Language and we use CAML to define queries against SharePoint list data or library data. And then we can use the SharePoint CAML query with CSOM or JavaScript or Server Object model. CAML is an XML-based query language.

## SharePoint Online caml query builder

We can use the [U2U CAML Query Builder for SharePoint 2013](https://www.u2u.be/software) to write the CAML query for SharePoint 2013, 2016, [SharePoint 2019](https://www.spguides.com/sharepoint-2019/), or [SharePoint Online](https://www.spguides.com/sharepoint-online/). This tool will be helpful to create and test SharePoint CAML Queries.

Here is a SharePoint caml query example.

<Query>

<Where>

<Eq>

<FieldRef Name='Title' />

<Value Type='Text'>dummy@gmail.com</Value>

</Eq>

</Where>

<OrderBy>

<FieldRef Name='Title' Ascending='True' />

</OrderBy>

</Query>

<ViewFields>

<FieldRef Name='Title' />

</ViewFields>

<QueryOptions />

Operator in CAML Query

Here we can use various other operators like below:

|  |  |
| --- | --- |
| Comparison Operators | General Meaning |
| Eq | = |
| Gt | > |
| Lt | < |
| Geq | >= |
| Leq | <= |
| Neq | <> |
| Contains | Like |
| IsNull | Null |
| IsNotNull | NotNull |
| BeginsWith | Beginning with word |
| DateRangesOverlap | compare the dates in a recurring event with a specified DateTime value, to determine whether they overlap |

**FieldRef Name:** This attribute provides the internal name of the field that is referenced. Specifies the type of reference for a field in an events list.

**ViewFields:** ViewFields basically gives us option to reduce data in response. It allows us to specify which fields we want in return for our query. It is always a best practice to use ViewFields as and when possible.

**-:Day 3rd :-**

## Features in SharePoint

Microsoft SharePoint helps us on enable and disable site collection features that determine everything from the use of workflows to the online help collection that should be available to users. Most site collection features are turned off by default, but we can easily change that.

**To enable or disable a site collection feature**

* On modern communication sites, select Site contents in the top menu bar and then click Site settings.
* On a modern team site, select Site contents in the left pane, and then click Site settings in the top navigation bar.
* On some sites, click Settings and then click Site Settings. If you don't see Site settings, click Site information and then click View all site settings.

On the Site settings page, click Site collection features under the Site Collection Administration heading.

Now we need to Activate or Deactivate on each site collection feature we want to enable or disable:

Click Activate to enable the site collection feature.

Click Deactivate to disable the site collection feature.

**-:Day 4th :-**

## Types of pages in SharePoint

## There are two primary types of pages in Microsoft SharePoint Foundation. They are site pages and application pages. The following topic will discuss the differences between the two types of pages. Application and site pages both inherit their layout from the same master page.

## Site pages: Site pages are pages that are created, edited, and customized by end users. They are primarily used for the content in a site. Site pages come in two types🡪1.) a wiki page and

## 2.) a Web Parts page.

## A wiki page contains text, images, Web Parts, and other elements.

## A Web Parts page contains Web Parts in Web Part zones. Web Part have a predefined layout that uses Web Part zones. Both types of site pages are edited using a Web browser or Microsoft SharePoint Designer.

## When a user customizes a site page, the template for the page is then stored in the content database. The page is retrieved from the content database every time it is requested by a user. A customized page can, however, be reset to the original template page through the Web browser or a tool such as SharePoint Designer.

## Customized site pages cannot contain in-line server code. Since they are rendered not compiled hence it is not easy to add any inline code, code behind or code beside. Best way of adding code to these pages is through web-parts, server controls in master pages, user controls stored in "Control Templates" folder or through smart parts. If we want to add any inline code to master page, first you need to add configuration within web.config to add code behind to SharePoint master pages or page layouts.

## These Pages are stored in the Content Database and they are parsed when requested by user. A wiki page and web part page is an example of Site Pages. They can be edited, modified by the Power Users and customized according to their needs.

## SharePoint specific features like Information Management Policies, Workflows, auditing, security roles can only be defined against site pages not against application pages.

## Application pages: Application pages are used to support application implementations in SharePoint Foundation.

## Application pages are stored on the file system of the front-end Web server in the ProgramFiles/Common Files/Microsoft Shared/web server extensions/14/TEMPLATE/LAYOUTS directory and exist for every site in a Web application. This folder is mapped to an Internet Information Services (IIS) virtual directory called \_layouts.

## Every site and subsite will have access to the application pages by using the \_layouts virtual directory. For example, http://myserver/\_layouts/settings.aspx and http://myserver/subsite/\_layouts/settings.aspx access the same application page on the front-end Web server unlike site pages, which are an instance for the specified site.

## Application pages are not subject to the same restrictions as site pages. They allow in-line code without restriction. They cannot, however, use dynamic Web Parts or Web Part zones or be modified using SharePoint Designer. Modifying the default application pages is not supported in SharePoint Foundation. Custom application pages can be added to a subdirectory inside the \_layouts folder.

## Application pages are same as ASP.net Pages and stored on the layouts folder of SharePoint front end web server. When user requests , application pages are compiled and they are much faster than Site Pages. Admin Pages like settings.aspx, accessdenied.aspx are an example of Application Pages. Thus we can say that Application pages are common to all SharePoint Users.

## Master Pages: Master pages provide the look and feel and standard behavior that we want for all of the pages in our site. Together with content pages, they produce output that combines the layout of the master page with content from the content page.

**-:Day 5th :-**

## Web Parts page in SharePoint

## A Web Parts page contains Web Parts in Web Part zones. Web Part have a predefined layout that uses Web Part zones. Both types of site pages are edited using a Web browser or Microsoft SharePoint Designer.

## 

## When a user customizes a site page, the template for the page is then stored in the content database. The page is retrieved from the content database every time it is requested by a user. A customized page can, however, be reset to the original template page through the Web browser or a tool such as SharePoint Designer.

## Customized site pages cannot contain in-line server code. Since they are rendered not compiled hence it is not easy to add any inline code, code behind or code beside. Best way of adding code to these pages is through web-parts, server controls in master pages, user controls stored in "Control Templates" folder or through smart parts. If we want to add any inline code to master page, first you need to add configuration within web.config to add code behind to SharePoint master pages or page layouts.

## These Pages are stored in the Content Database and they are parsed when requested by user.

## Web part pages having Media and Content part which contains Content Editor and Script Editor

## 

## Difference between Content Editor and Script Editor:

## Webpart usage:

## In Script Editor Web Part we can add HTML , CSS and JavaScript only.

## In Content Editor webpart we can add HTML, CSS, JavaScript, formatted text, tables, hyperlinks, and images also.

## Content Link availability:

## In Content Editor webpart, we can give a link to the file uploaded in a document library.

## In Script editor webpart we do not have such option.

## Advantage of having "Content Link" property in CEWP is that, we do not need to always edit the page for code modification. We can just edit the file linked to the CEWP. This helps in preventing accidental loss of code as we can restore the version of that file.

## Code reusability:

## In Script editor webpart, we need to manually update the code on all pages by editing the page.

## As content editor webpart has a Content Link Property, we can reuse the same JavaScript, html, CSS, etc. files on multiple pages. If we make the code change, it will be visible on all the pages where the content editor is added with that file.

## -:Sixth Week :-

## Three Programmoming models used for the Client Side Object Model:

## .NET Client Side Object model

## JavaScript Object model

## Silverlight Object model

## Limit in view

## Default 🡪 30 item,

## Maximum limit 🡪100 item

**SPContext in SharePoint**

SPContext is often used with custom web parts and pages to retrieve SharePoint site objects from the context of the SharePoint site accessed. Its a good practice to use SPContext instead of creating new objects. SPContext object will be available for access in our coding as long as the current HttpContext is not null. Meaning, if a user is logged in to a SharePoint site, the SPContext object will always be available.

## 

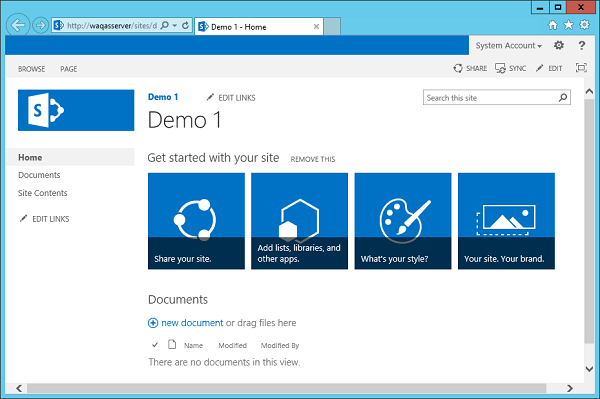
## SharePoint - Sandbox Solutions

It is similar to uploading a document to a document library. When we finish our development, we are going to take the solution package and instead of giving it to our SharePoint administrator, we will give it to an end user, someone with site-collection owner privilege. Then they will take the package and upload it to the site-collection solution gallery.

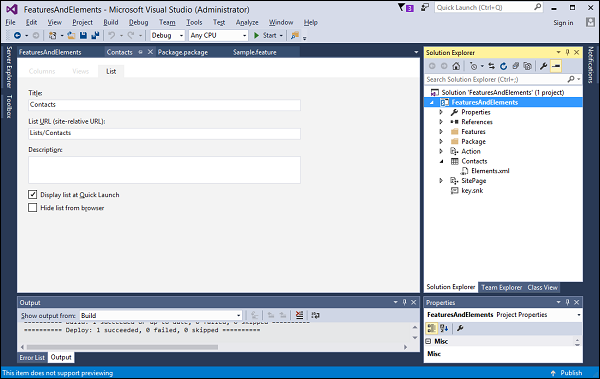
Just like with Farm solutions, the tools in Visual Studio automate this deployment process, during development.

Let us have a look into a simple example of Sandbox Solution Deployment. It is quite simpler than Farm solution deployment.

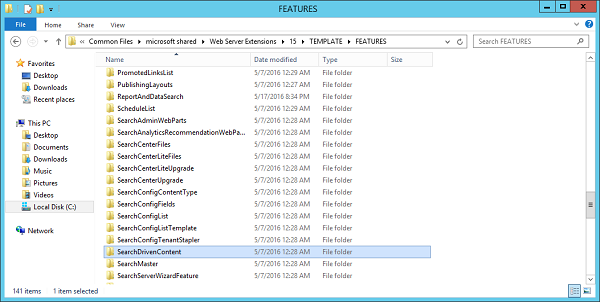
**Step 1** − Here we need to create a new site collection and call it Demo 1.



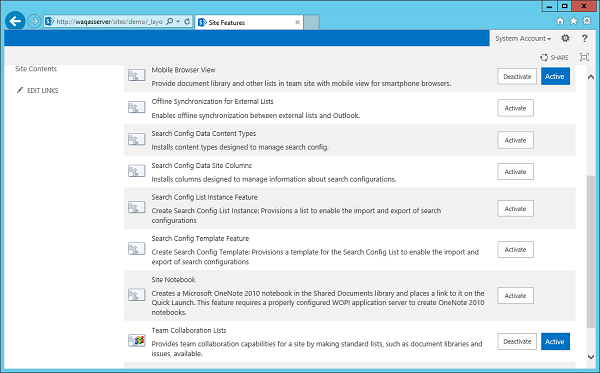
**Step 2** − Change Contacts list name back to just Contacts in FeaturesAndElements project.



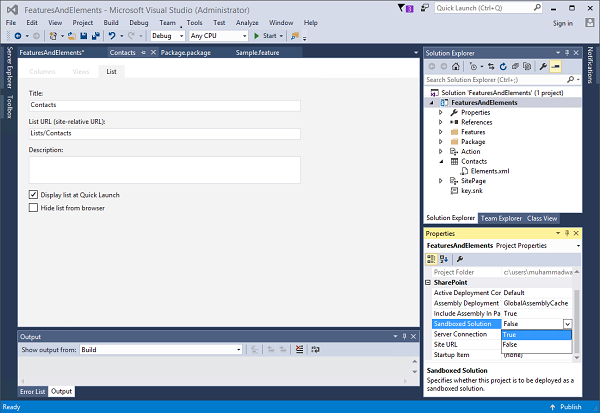
**Step 3** − Retract the solution by right-clicking on the project and choosing Retract. If we come back to the SharePoint system folders, you will notice that our Feature folder is absent.



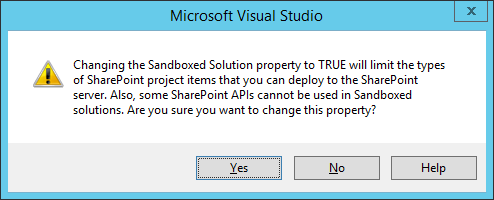
Next, if we go to Manage site features, we should not see Sample Feature.



Step 4 − Go back to Visual Studio project, click the project in the Solution Explorer and then go to the properties window. Change Sandbox Solution from False to True.



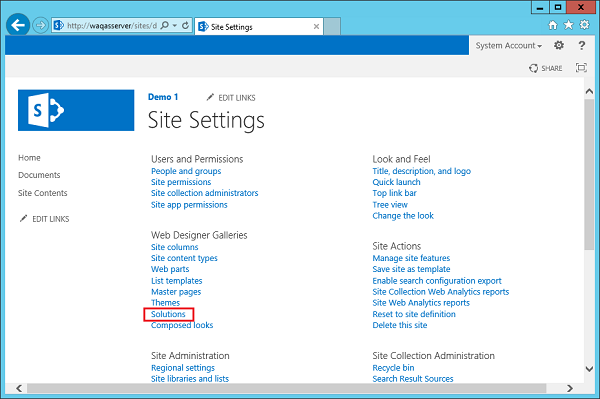
A warning dialogue is displayed.



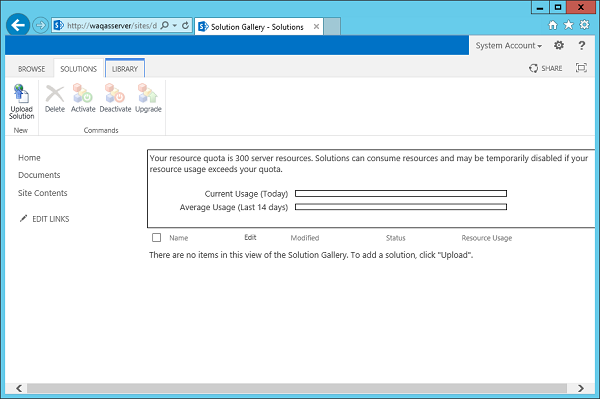
Click Yes to make the change. In this case, building a sandbox solution is the same as building a farm solution, but the deployment process is completely different.

With the sandbox solution, instead of deploying files up into the SharePoint system folders, we deploy into the SharePoint content database.

Step 5 − Go to the Site settings. Under the Web Designer Galleries, there is Solutions gallery.

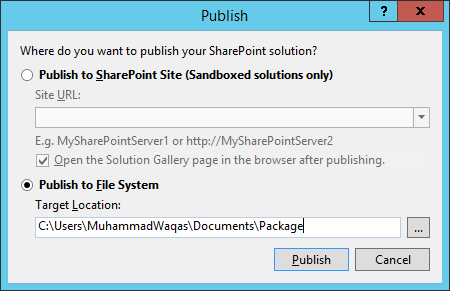


Step 6 − Click the Solutions link and you will see the following page where we deploy our sandbox solutions.



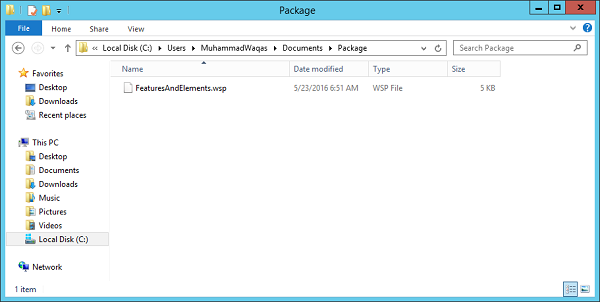
Now all done with the development. Instead of giving the solution package to the SharePoint administrator and then having them use PowerShell or Central Admin to deploy the Farm solution,we can give our package to an end user, someone with site-collection owner privilege and then they can upload the solution into the Solution gallery.

Step 7 − Go back to Visual Studio, right-click and select Publish to File System.



Click the Publish button to publish the New Solution Package to the package folder.

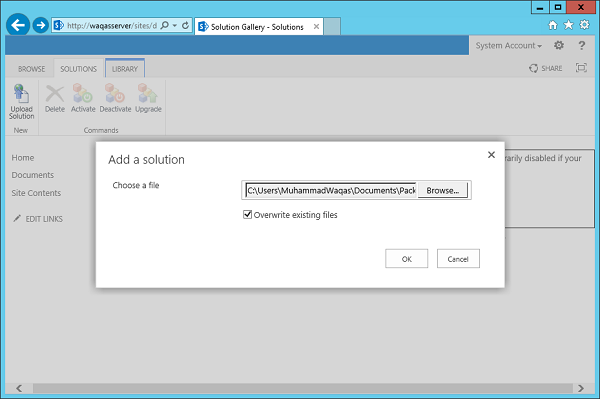
we will see the package in the Package folder.



Step 8 − Now go to the SharePoint site. Click the Upload Solution button option on the Ribbon.

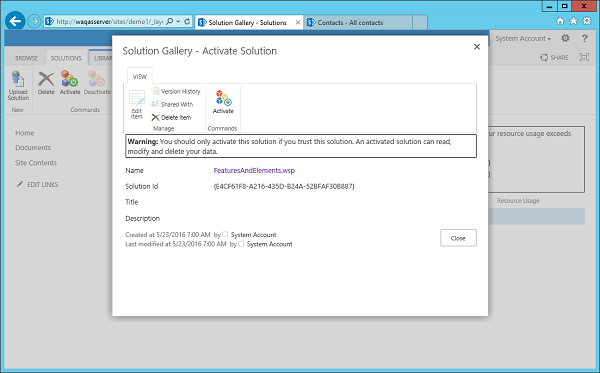


Step 9 − Browse to your FeaturesAndElements solution. Click OK.



You will see the following dialogue.

Step 10 − You just need to click the Activate button to activate the sandbox solution



Step 11 − Go to the Manage site features.



We can now see our Sample Feature and when we click Activate, we should get the same behavior as we had seen before.

## Activate

## Differences between Farm Solution and Sandbox Solution

## Farm solution

* Farm solutions are hosted in the IIS worker process (W3WP.exe).
* Running code of farm solution can affect the whole farm.
* When we debug a SharePoint project whose Sandboxed Solution property is set to "farm solution," the system's IIS application pool recycles before SharePoint retracts or deploys the feature so as to release any files locked by the IIS worker process. Only the IIS application pool serving the SharePoint project's site URL is recycled.
* IIS server restart.

## Sandbox solution

* Sandboxed solutions, which are hosted in the SharePoint user code solution worker process (SPUCWorkerProcess.exe)..
* Running code of sandbox solution can only affect the site collection of the solution. Because sandboxed solutions do not run in the IIS worker process, neither the IIS application pool nor the IIS server restart.
* Visual Studio attaches the debugger to the SPUCWorkerProcess process that the SPUserCodeV4 service in SharePoint automatically triggers and controls.
* It is not necessary for the SPUCWorkerProcess process to recycle to load the latest version of the solution.

## Types of solutions

## Farm Solution

## Sandbox solution

## Sharepoint add-in (sharepoint hosted app) Jvascript, query.

## Provider hosted app

## Auto hosted app

**SharePoint Workflow:**

**Types of Workflow**

* List Workflow
* Reusable Workflow
* Site Workflow

**Workflow Triger option:**

Trigger condition takes the format of an expression and must evaluate to either true or False. If the trigger condition is true then the flow will run, else it will ignore the trigger event.

Triger options are:

* Manually
* Automatic when condition satisfy.
* When an item is created and
* When item is changed.

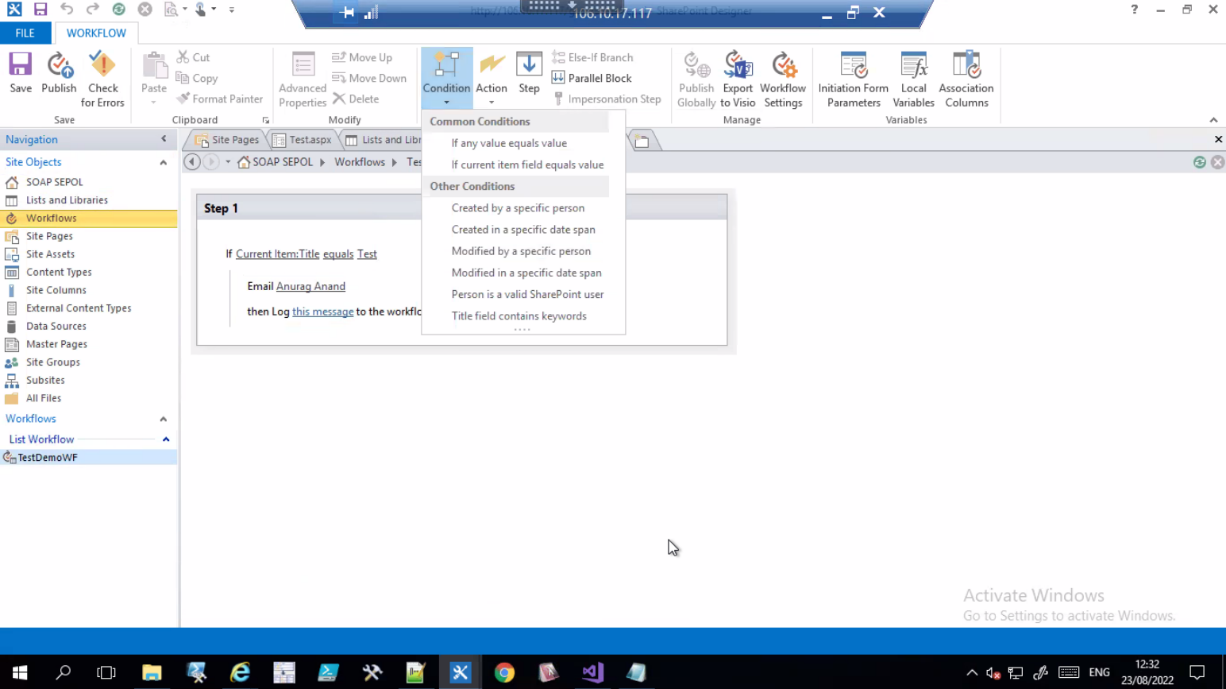
For example, if we use a trigger called ‘When an item is created or modified”, the flow will trigger with each update.

**Conditions of workflow:**

Workflow will be executed if the given condition satisfies then it executes the action inside the condition.

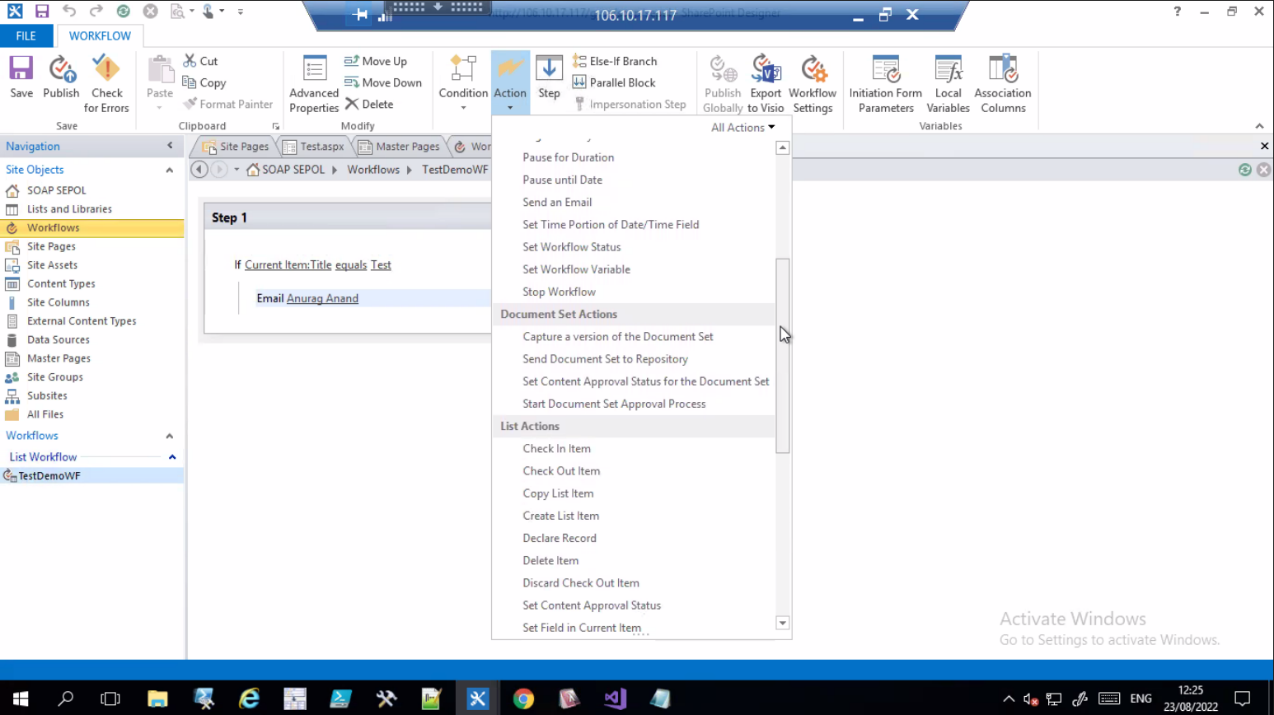
* Any value equals value
* Current item field equals value
* Created by specific person
* Created by specific date span etc.

We can apply workflow condition by choosing the suitable condition from condition tab:



Then Specify the task/action to be execute if the given condition satisfies from the action tab like:

* Pause for duration
* Pause until date
* Send an Email
* Set Workflow status
* Set Workflow variable
* Stop Workflow
* Check in Item
* Check Out Item etc.



## -:Sevethth Week :-

**CSOM**

Classes in SharePoint

SPContext 🡪 Server Side

ClientContext 🡪 Client Side

References For custom side Page 🡪 SP.js and SPServices-min.js

CSOM Managed Code (C#) References:

Microsoft.SharePoint.Client.dll

Microsoft.SharePoint.Client.Runtime.dll

SSOM References:🡪 Microsoft.SharePoint.dll

**SharePoint Rest API**

**Rest API stands for Representational State Transfer and is based on standard Open Data Protocol (OData).**

By using Rest API, we can interact with SharePoint remotely by using any technology that supports Rest protocol. By using SharePoint rest API, we can do various operations like create, update, delete and display operations, like the SharePoint REST API CRUD operations.

By using Rest API, we can do CRUD operations from [SharePoint Add-ins](https://www.enjoysharepoint.com/sharepoint-hosted-app-development-tutorial/), solutions, and from client object model applications also.

SharePoint Rest API is another form of client object model similar to [JSOM](https://www.spguides.com/javascript-object-model-sharepoint/), or [CSOM](https://www.spguides.com/sharepoint-csom-tutorial/).

## Advantages of using Rest API in SharePoint

* It allows you to work remotely with SharePoint sites using any technologies that support Rest protocols.
* we do not need to add any or refer to any SharePoint libraries or dlls to work with Rest API, we just need to make an HTTP rest endpoint (URL) for the SharePoint operations like insert, update or delete.

## SharePoint Rest API HTTP commands :

To work with Rest API in SharePoint Online or SharePoint 2013/2016/2019, the first things we need to construct a **Restful HTTP request** by using the OData protocol standard.

If we want to do create, read, update and delete operations using **SharePoint Rest API**, then we need to use the GET, POST, PUT, MERGE, DELETE and PATCH **HTTP methods**.

### **GET**

**HTTP GET** method is used to read information from the SharePoint server or to retrieve information from SharePoint list if we want to read items from the SharePoint list. So mostly for the Read operations we use the GET method.

### **POST**

If we want to create or update operations like creating a list or creating an item in a SharePoint Online list, then we use the **Rest API POST operations**. In Rest API POST operations, if any properties are not required then these are set to their default values.

### **PUT/MERGE**

If we want to update an existing object like an update item in a SharePoint list or update the SharePoint list title, then we can use the Rest API **PUT** and **MERGE** operations.

**Difference between HTTP PUT and MERGE in SharePoint rest API.**

* **In the**REST API MERGE HTTP method**, setting properties is optional, and if any properties that you don’t explicitly set keep their current property.**
* For the PUT operations, it is mandatory to set up all the required properties while updating SharePoint objects. And for the optional properties, if you do not specify the values, then it will set to the default properties.

### **DELETE**

**HTTP DELETE** method is used when we want to delete any SharePoint objects like deleting SharePoint list, list items, documents, etc.

### **Various useful SharePoint REST endpoint examples**

For doing any kind of operation in SharePoint using Rest API, the first thing we need to do is to create the rest endpoint.

All the rest endpoint URL starts with:

<https://SiteURL/Sites/SiteName/_api/>

**Example:**

Below is an example for SharePoint Online.

<https://tsinfotechnologies.sharepoint.com/sites/SPGuides/_api/>

For on-premises versions like SharePoint 2013, [SharePoint 2016](https://www.enjoysharepoint.com/sharepoint-2016-installation-step-by-step/), or [SharePoint 2019](https://www.spguides.com/sharepoint-2019/), the Rest api endpoint will be like below:

<http://bsahoo3:8787/sites/Training/_api/>

Here are a various **SharePoint rest api endpoint examples**, I have taken a SharePoint Online site as reference, but the Rest API endpoints will be same for SharePoint on-premises.

But while working with Rest API code, we can get the site URL by using the below JavaScript variable:

\_spPageContextInfo.webAbsoluteUrl

## SharePoint rest api crud operations:

### **SharePoint rest api get all list items**

**To get all list items using SharePoint rest api.**

Here I have added a button and on button click, we can retrieve SharePoint list items using rest api and display them.

I have taken a button and on click of that button, calling the GetListItems() method. And displaying all the SharePoint list items in a dialog box in the onSuccess method.

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<button onclick="GetListItems();" type="button">Get All List Items​</button>

<script>

function GetListItems()

{

var url = \_spPageContextInfo.webAbsoluteUrl + "/\_api/web/lists/getByTitle('Employees')/items";

$.ajax({

url: url,

type: "GET",

headers:

{

"accept":"application/json; odata=verbose"

},

success: onSuccess,error: onError

});

}

function onSuccess(data) {

var items = data.d.results;

var allItems='';

for (var i = 0; i < items.length; i++) {

allItems+="Item ID- "+ items[i].Id+ " : Title- " + items[i].Title + '\r\n';

}

alert(allItems);

}

function onError(error) {alert(JSON.stringify(error));

}

</script>

**Code to insert item to SharePoint using rest api.**

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<button onclick="InsertItem();" type="button">Insert Item to SharePoint List​</button>

<script>

function InsertItem()

{

var restendpoint = \_spPageContextInfo.webAbsoluteUrl + "/\_api/web/lists/getByTitle('Employees')/items";

$.ajax

({

url: restendpoint,

type: "POST",

data: JSON.stringify

(

{

\_\_metadata:

{

type: "SP.Data.EmployeesListItem"

},

Title : "Bijay Kumar"

}),

headers:

{

"Accept": "application/json;odata=verbose",

"Content-Type": "application/json;odata=verbose",

"X-RequestDigest": $("#\_\_REQUESTDIGEST").val(),

"X-HTTP-Method": "POST"

},

success: function()

{

alert("Item added to the SharePoint list");

},

error: function()

{

alert("Error Occurred!");

}

});

}

</script>

**Code For Update**

**We need to get the item by id and then we can update.**

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<button onclick="UpdateItem();" type="button">Update Item (ID=4)​</button>

<script>

function UpdateItem()

{

var restendpoint = \_spPageContextInfo.webAbsoluteUrl + "/\_api/web/lists/getByTitle('Employees')/items/getbyid(4)";

$.ajax({

url: restendpoint,

type: "POST",

headers: {

"accept": "application/json;odata=verbose",

"X-RequestDigest": $("#\_\_REQUESTDIGEST").val(),

"content-Type": "application/json;odata=verbose",

"IF-MATCH": "\*",

"X-HTTP-Method": "MERGE"

},

data: "{\_\_metadata:{'type':'SP.Data.EmployeesListItem'},Title:'Bhawana Rathore'}",

success: function(data) {

alert('Item updated successfully!');

},

error: function(error) {

alert(JSON.stringify(error));

}

});

}

</script>

**Code SharePoint rest api delete list item.**

This is also a POST operation but we have to pass additional parameter as “X-Http-Method”: “DELETE”.

The rest endpoint is the same as updating list item, because we will first get the item by id which item you want to delete.

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<button onclick="DeleteItem();" type="button">Delete Item (ID=3)​</button>

<script>

function DeleteItem()

{

var restendpoint = \_spPageContextInfo.webAbsoluteUrl + "/\_api/web/lists/getByTitle('Employees')/items/getbyid(3)";

$.ajax({

url: restendpoint,

type: "POST",

headers: {

"Accept": "application/json;odata=verbose",

"X-Http-Method": "DELETE",

"X-RequestDigest": $("#\_\_REQUESTDIGEST").val(),

"If-Match": "\*"

},

success: function(data) {

alert('SharePoint list item deleted');

},

error: function(data) {

alert('Error Occurred!');

}

});

}

</script>

**Rest API Headers**

The Rest headers are an important part of the API request and response as they represent the meta-data associated with the API request and response.

We use REST API headers to represent the meta-data associated with an API request and response.

If we encounter issues with an API, the first place we should look is the headers, since they can help us to track down any potential issues.

**Examples of API Headers**

Here are some of the most common API Headers we will encounter when testing any API.

**Authorization:** Contains the authentication credentials for HTTP authentication.

**WWW-Authenticate:** The server may send this as an initial response if it needs some form of authentication before responding with the actual resource being requested.

**Accept-Charset:** This header is set with the request and tells the server which character sets (e.g., UTF-8, ISO-8859-1, Windows-1251, etc.)

are acceptable by the client.

**Content-Type:** Tells the client what media type (e.g., application/json, application/javascript, etc.) a response is sent in.

**Cache-Control:** The cache policy defined by the server for this response,

a cached response can be stored by the client and re-used till the time defined by the Cache-Control header.

**X-Http-Method :** Specify the http request method.

**X-RequestDigest :** Request digest is a client side “token” to validate posts back to SharePoint to prevent attacks where the user might be tricked into posting data back to the server. The token is unique to a user and a site and is only valid for a (configurable) limited time.

**If-Match :** This is used for concurrency control. To avoid concurrency. Like if we click in the update button at the same time it wont allow others to update for same time. It basically matches \* with the .net frameworks

**Master Page and Site Page**

**SharePoint Master Page:**

## Master Pages: Master pages provide the look and feel and standard behaviour that we want for all of the pages in our site. Together with content pages, they produce output that combines the layout of the master page with content from the content page.

## SharePoint master pages provide the interface and overall layout of the pages on a SharePoint site. The common elements of a page – its header, navigation links, Site Actions menu, footer and so forth they are placed in the same areas regardless of the page you’re viewing.

## Site pages: Site pages are pages that are created, edited, and customized by end users. They are primarily used for the content in a site. Site pages come in two types🡪1.) a wiki page and

## 2.) a Web Parts page.

## A wiki page contains text, images, Web Parts, and other elements.

## A Web Parts page contains Web Parts in Web Part zones. Web Part have a predefined layout that uses Web Part zones. Both types of site pages are edited using a Web browser or Microsoft SharePoint Designer.

## Template for the page is stored in the content database. The page is retrieved from the content database every time it is requested by a user.

## A customized page can, however, be reset to the original template page through the Web browser or a tool such as SharePoint Designer.

## Customized site pages cannot contain in-line server code.

## These Pages are stored in the Content Database and they are parsed when requested by user.

## They can be edited, modified by the Power Users and customized according to their needs.

**Pages Extension**

* Master page 🡪 .master
* Site page 🡪.ascx
* Solution🡪 Frontend 🡪.ascx

Backend .ascx.cs

**Important Links :**

**SharePoint SPContext**

* https://www.sharepointdiary.com/2012/03/what-is-spcontext-in-sharepoint.html

**SharePoint CRUD Operation (JSOM)**

* <https://www.spguides.com/crud-operations-using-jsom-in-sharepoint/>

**SharePoint REST API**

* <https://www.spguides.com/sharepoint-rest-api/>
* <https://www.codeproject.com/Articles/990131/CRUD-Operation-to-List-Using-SharePoint-Rest-API>

**SharePoint Hosted App**

* <https://www.enjoysharepoint.com/sharepoint-hosted-app-development-tutorial/>