

Drop Box File Manager

SLIIT Cyber Security

K A Ruvin Sawmith MS20924782

W G Therangika Manuhansini MS20924850

Index

1. How OAuth 2.0 Works in DropBox
 - 1.1. Requesting Token Code behalf of the User (Redirect user for DropBox Authentication)
 - 1.2. Convert Token Code to Token
 - 1.3. Token Scopes
 - 1.4. Configuring Scopes of the DropBox Application through App Console
2. Technologies
3. DropBox File Manager Application Manual
4. Source Code

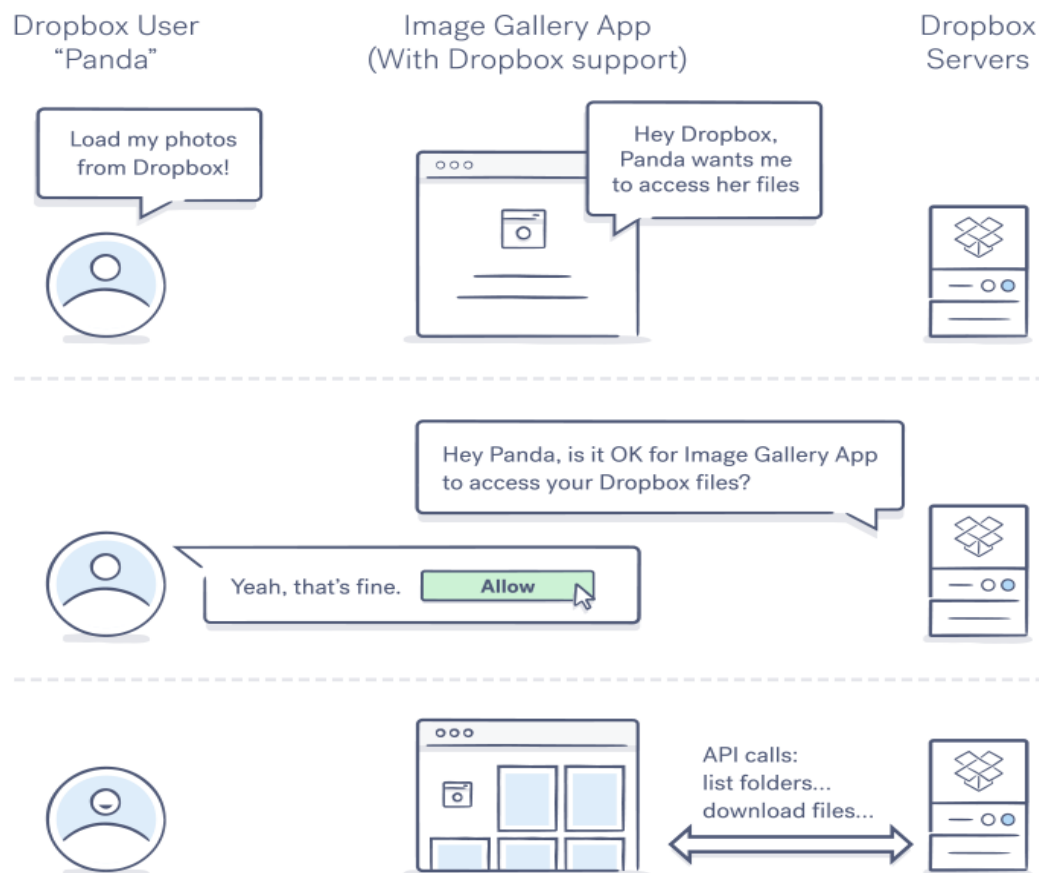
How OAuth 2.0 Works in DropBox

If Dropbox user or Application which user grants permission, wants to access resources under DropBox, Need to authenticate user,

For this process need, DropBox Application and its App Key.

Dropbox uses OAuth 2.0, when an application ask for Login Permission behalf of a legitimate DropBox user, DropBox Authentication Server redirects User for Login page it owns. After success login, DropBox returns a short time living token which should be passed into the header of each API requests.

Diagram 1.0 Depicts How Authentication Involves in the scenario.



1. Requesting Token Code behalf of the User (Redirect user for DropBox Authentication)

Method Type - Get

Url - https://www.dropbox.com/oauth2/authorize?client_id=<Your App Key>&redirect_uri=<Pre Configured Redirect URI>&response_type=code

Response – As Response If User Loggin Successfully, Authentication server reirects to prior given URI with Parameter “Code”.

This is the Token Code offered by the OAuth Server.

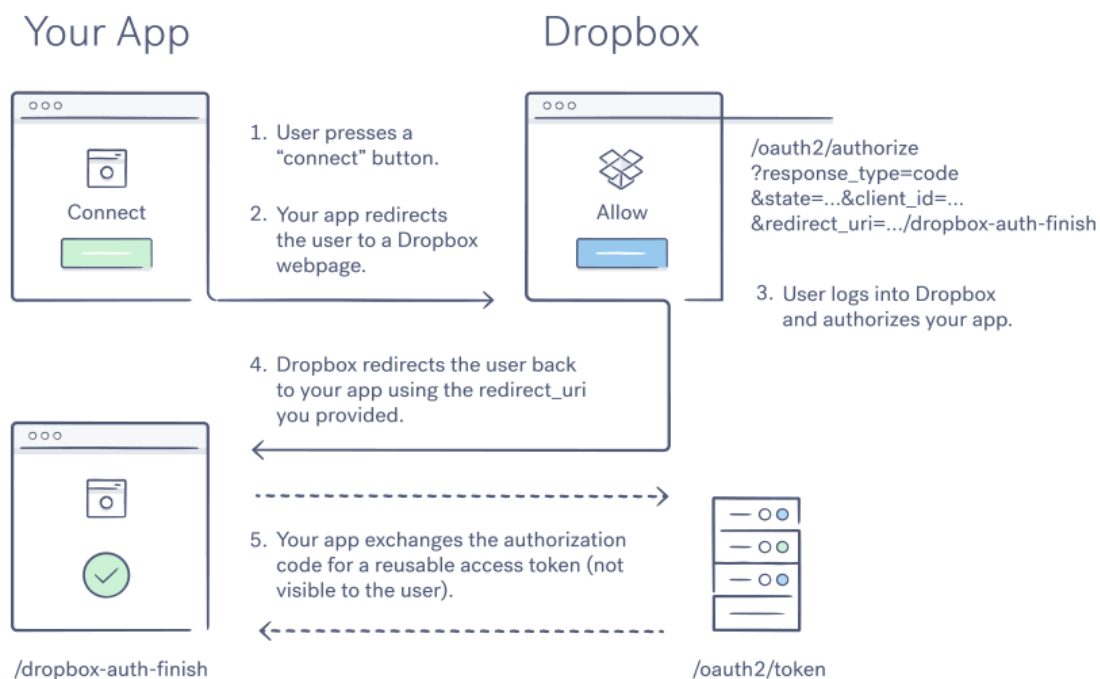
2. Convert Token Code to Token

Method Type – Post

Url - `https://api.dropbox.com/oauth2/token?code=<token_code received>&grant_type=authorization_code&redirect_uri=<configured redirect uri>`

Response – Receive Token for the User Resource Access

This mentioned process could be demonstrated by Diagram 2.0 (Auth2.0 Code Flow)



3. Token Scopes

Token scope plays huge role. When requesting token there should be enough privileges to complete the access of resource API end points.

4. Configuring Scopes of the DropBox Application through App Console

Diagram 3.0 Depicts Scopes under DropBox



scoped-app-demo-of-delight

Settings	Permissions	Branding	Analytics
Individual Scopes			
Individual scopes include the ability to view and manage a user's files and folders. View Documentation			
Account Info Permissions that allow your app to view and manage Dropbox account info			
<input checked="" type="checkbox"/> account_info.read Read Dropbox account information (Required)			
<input type="checkbox"/> account_info.write Create, modify, and delete Dropbox account information			
Files and folders Permissions that allow your app to view and manage files and folders			
<input type="checkbox"/> files.content.read Read file data			
<input type="checkbox"/> files.content.write Create, modify, and delete file data			

Technologies

ASP MVC 5 – render views, acting as the backend which performs application logical process.


DropBox API nuget package – access functionalities of DropBox resource server (Upload, Download and delete Files)

DropBox File Manager Application Manual

Functionalities

- A. Login to Dropbox Account
- B. View Available File list
- C. Upload New File
- D. Delete Available file
- E. Preview Files with DropBox End point
- F. Download a copy of the File
- G. Revoke DropBox Access to Applicatin(Using DropBox User Logout)

Login with Social Media

 Login with DropBox

© 2021 - My ASP.NET Application

- a) Supply the logging redirection button to DropBox when user click the sign in button user will redirect to DropBoX Authentication Page.



Sign in to Dropbox to link with File Uploader SS



Sign in with Google

or

Email

Password

This page is protected by reCAPTCHA, and subject to the Google [Privacy Policy](#) and [Terms of Service](#).

[Forgot your password?](#)

Sign in

- b) User will authenticate with her/his credentials and if authentication success redirect user to our app DropBox file manager.

DROP BOX FILE MANAGER

DropBox File Manager [About](#) [Contact](#) [Logout](#)

Details

[Upload New](#)

Name	ModifiedAt	Size	Path	
Get Started with Dropbox.pdf	3/11/2021 4:25:25 AM	1102331	/get started with dropbox.pdf	Download Preview Delete
test.txt	3/11/2021 10:44:40 AM	513	/public/test.txt	Download Preview Delete
download(1).jpg	3/12/2021 5:58:21 AM	11279	/public/download(1).jpg	Download Preview Delete

© 2021 - My ASP.NET Application

- c) List of preview the users files stored in DropBox. Supplying functionalities of uploading new download exist, delete exist and preview of exist files.

DropBox File Manager [About](#) [Contact](#) [Logout](#)

Upload File

Name

Path No file selected.

[Back to List](#)

© 2021 - My ASP.NET Application

- d) Upload process of new file, file and file name is required.

About.

DropBox File Manager

This Application has configured OAuth Login with DropBox Authentication server and then access logged user resources using DropBox resource server.

Functionalities

1. Login to Dropbox Account
2. View Available File list
3. Upload New File
4. Delete Available file
5. Preview Files with DropBox End point
6. Download a copy of the File
7. Revoke DropBox Access to Application(Using DropBox User Logout)

© 2021 - My ASP.NET Application

Delete

Are you sure you want to delete this?

DropBoxFile

Name	Get Started with Dropbox.pdf
ModifiedAt	3/11/2021 4:25:25 AM
Size	1102331
Path	/get started with dropbox.pdf

[Delete](#) | [Back to List](#)

© 2021 - My ASP.NET Application

Source Code

DropBoxController (MVC Cotroller)

```
using Dropbox.Api;
using PhotoApp.Models;
using PhotoApp.Service;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
```



```

using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PhotoApp.Controllers
{
    public class DropBoxController : Controller
    {
        public string token { get; set; }
        // GET: DropBox
        public ActionResult Index()
        {
            var apiAuth = ConfigurationManager.AppSettings["DropBoxAppKey"].ToString();
            return
Redirect("https://www.dropbox.com/oauth2/authorize?client_id="+apiAuth+"&redirect_uri=ht
tps://localhost:44304/DropBox/Details&response_type=code");
        }

        // GET: DropBox/Details?code=
        public async System.Threading.Tasks.Task<ActionResult> Details(string code)
        {
            ViewBag.Logout = true;
            if (ValueStore.token == null)
            {
                string tokenCode = code;
                var tokenObj = await new DropBoxService().RetreiveTokenAsync(tokenCode);
                ValueStore.token = tokenObj.access_token;
            }

            //var user = await new DropBoxService().GetUserBasicInfo(ValueStore.token);
            var result = await new DropBoxService().GetFilesAndFoldersInfo(ValueStore.token);
            return View(result);
        }

        // GET: DropBox/Create
        public ActionResult Create()
        {
            ViewBag.Logout = true;
            return View();
        }

        // Get: DropBox/Download
        [HttpGet]

```

```

        public async System.Threading.Tasks.Task<ActionResult> Download(string filename, string
folder)
        {
            try
            {
                var result =await new DropBoxService().DownloadAFile(ValueStore.token, filename,
folder);
                return File(result, System.Net.Mime.MediaTypeNames.Application.Octet, filename);
            }
            catch(Exception exr)
            {
                return View();
            }
        }

// GET: DropBox/Edit/5
public ActionResult Edit(int id)
{
    return View();
}

// POST: DropBox/Edit/5
[HttpPost]
public async System.Threading.Tasks.Task<ActionResult> Create(DropBoxFile dropBoxFile)
{
    try
    {
        string FileName = dropBoxFile.Name;
        string FileExtension = Path.GetExtension(dropBoxFile.ImageFile.FileName);

        FileName = FileName.Trim() + FileExtension;
        await new DropBoxService().UploadAFile(ValueStore.token, dropBoxFile.ImageFile,
FileName);
        return RedirectToAction("Details");
    }
    catch
    {
        return View();
    }
}

// GET: DropBox/Delete/5
public async System.Threading.Tasks.Task<ActionResult> Delete(string path)
{

```

```

        ViewBag.Logout = true;
        var Dropobject = await new DropBoxService().GetParticularFileInfo(ValueStore.token,
path);
        return View(Dropobject);
    }

    // POST: DropBox/Delete/5
    [HttpPost]
    public async System.Threading.Tasks.Task<ActionResult> Delete(string path,
FormCollection collection)
    {
        try
        {
            await new DropBoxService().DeleteAFile(ValueStore.token, path);
            return RedirectToAction("Details");
        }
        catch
        {
            return View();
        }
    }

    public async System.Threading.Tasks.Task<ActionResult> GetPreview(string path)
    {
        var result = "https://www.dropbox.com/preview"+path+"?role=personal";
        return Redirect(result);
    }

    public async System.Threading.Tasks.Task<ActionResult> Logout()
    {
        await new DropBoxService().RevokeTokenAsync();
        return RedirectToAction("Index", "Home");
    }
}
}

```

1. DropBoxService.cs

```

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

```

```

using System.Configuration;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Web;
using System.Web.Script.Serialization;
using Dropbox.Api;
using Dropbox.Api.Files;
using PhotoApp.Models;

namespace PhotoApp.Service
{
    public class DropBoxService
    {
        public async System.Threading.Tasks.Task<DropBoxTokenResponse>
RetreiveTokenAsync(string tokenCode)
        {
            HttpClient client = new HttpClient();
            var appKey = ConfigurationManager.AppSettings["AuthApiSecret"].ToString();
            client.DefaultRequestHeaders.Add("Authorization", "Basic "+ appKey);
            string url = "https://api.dropbox.com/oauth2/token?code=" + tokenCode +
"&grant_type=authorization_code&redirect_uri=https://localhost:44304/DropBox/Details";
            var response = await client.PostAsync(url,null);
            var contents = await response.Content.ReadAsStringAsync();
            JavaScriptSerializer serializer = new JavaScriptSerializer();
            var tokenObject = serializer.Deserialize<DropBoxTokenResponse>(contents);
            return tokenObject;
        }

        public async System.Threading.Tasks.Task RevokeTokenAsync()
        {
            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Authorization", "Bearer "+ ValueStore.token);
            string url = "https://api.dropboxapi.com/2/auth/token/revoke";
            await client.PostAsync(url, null);
            ValueStore.token = null;
        }

        public async System.Threading.Tasks.Task<DropBoxFile> GetParticularFileInfo(string token,
string path)
        {
            //List<DropBoxFile> objs = new List<DropBoxFile>();
            using (var dbx = new DropboxClient(token))

```

```

    {
        var list = await dbx.Files.ListFolderAsync(string.Empty, recursive: true);

        foreach (var item in list.Entries.Where(i => i.IsFolder))
        {
            Console.WriteLine("D {0}/", item.Name);
        }

        foreach (var item in list.Entries.Where(i => i.IsFile))
        {
            Console.WriteLine("F{0,8} {1}", item.AsFile.Size, item.Name);
            var obj = new DropBoxFile() { Name = item.Name, ModifiedAt =
item.AsFile.ClientModified.ToString(), Size = item.AsFile.Size.ToString(), Path =
item.AsFile.PathLower };
            if(obj.Path == path)
            {
                return obj;
            }
        }
        return null;
    }
}

public async System.Threading.Tasks.Task<List<DropBoxFile>>
GetFilesAndFoldersInfo(string token)
{
    List<DropBoxFile> objs = new List<DropBoxFile>();
    using (var dbx = new DropboxClient(token))
    {
        var list = await dbx.Files.ListFolderAsync(string.Empty, recursive: true);

        foreach (var item in list.Entries.Where(i => i.IsFolder))
        {
            Console.WriteLine("D {0}/", item.Name);
        }

        foreach (var item in list.Entries.Where(i => i.IsFile))
        {
            Console.WriteLine("F{0,8} {1}", item.AsFile.Size, item.Name);

```

```

        var obj = new DropBoxFile() { Name = item.Name, ModifiedAt =
item.AsFile.ClientModified.ToString(), Size = item.AsFile.Size.ToString(), Path =
item.AsFile.PathLower };
        objs.Add(obj);
    }
    return objs;
}

public async System.Threading.Tasks.Task<bool> UploadAFile(string token,
HttpPostedFileBase fileobject, string filenamewithExt)
{
    try
    {
        using (var dbx = new DropboxClient(token))
        {
            //string file = filepath;
            string folder = "/Public";
            string filename = filenamewithExt;
            string url = "";
            using (var mem = new MemoryStream())
            {

                fileobject.InputStream.CopyTo(mem);
                mem.Position = 0;
                var updated = dbx.Files.UploadAsync(folder + "/" + filename,
WriteMode.Overwrite.Instance, body: mem);
                updated.Wait();
                var tx = dbx.Sharing.CreateSharedLinkWithSettingsAsync(folder + "/" + filename);
                tx.Wait();
                url = tx.Result.Url;
            }
            Console.Write(url);
            return true;
        }
    }
    catch (Exception exr)
    {
        return false;
    }
}

```

```

public async System.Threading.Tasks.Task<byte[]> DownloadAFile(string token, string
fileName, string pathToFile)
{
    try
    {
        using (var dbx = new DropboxClient(token))
        {
            string folder = pathToFile;
            string file = fileName;

            var list = await dbx.Files.ListFolderAsync(string.Empty, recursive: true);
            foreach (var item in list.Entries.Where(i => i.IsFile))
            {
                string urlFile = item.AsFile.PathLower;
                if (String.Concat(pathToFile) == urlFile)
                {
                    using (var response = await dbx.Files.DownloadAsync(urlFile))
                    {
                        var s = response.GetContentAsByteArrayAsync();
                        s.Wait();
                        var data = s.Result;
                        return data;
                    }
                }
            }

            //return true;
            return null;
        }
    }
    catch (Exception exr)
    {
        return null;
    }
}

```

```

public System.Threading.Tasks.Task<Dropbox.Api.Users.FullAccount>
GetUserBasicInfo(string token)
{
    using (var dbx = new DropboxClient(token))
    {
        var user = dbx.Users.GetCurrentAccountAsync();
    }
}

```

```

        return user;
    }
}

public async System.Threading.Tasks.Task<bool> DeleteAFile(string token, string filepath)
{
    try
    {
        using (var dbx = new DropboxClient(token))
        {
            var result=await dbx.Files.DeleteV2Async(filepath);
            return true;
        }
    }
    catch (Exception exr)
    {
        return false;
    }
}
}
}

```

2. DropBoxFile.cs

```

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

namespace PhotoApp.Models
{
    public class DropBoxFile
    {
        public string Name { get; set; }
        public string ModifiedAt { get; set; }
        public string Size { get; set; }
        public string Path { get; set; }
        public HttpPostedFileBase ImageFile { get; set; }
    }
}

4.Create.cshtml
@model PhotoApp.Models.DropBoxFile

@{

```



```

ViewBag.Title = "Create";
Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Upload File</h2>

@using (Html.BeginForm("Create", "DropBox", FormMethod.Post, new
{
    enctype = "multipart/form-data"
}))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4></h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger"
            })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Path, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                <input type="file" name="ImageFile" required />
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Upload" class="btn btn-default" />
            </div>
        </div>
    </div>
}

```

```
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

3. Detail.cshtml

```
@model IEnumerable<PhotoApp.Models.DropBoxFile>
```

```
@{
    ViewBag.Title = "File Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<h2>Details</h2>
```

```
<p>
    @Html.ActionLink("Upload New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ModifiedAt)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Size)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Path)
        </th>
        <th></th>
    </tr>
```

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ModifiedAt)
        </td>
        <td>
```

```
        @Html.DisplayFor(modelItem => item.Size)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Path)
    </td>
    <td>
        @Ajax.ActionLink("Download", "Download", new { /* id=item.PrimaryKey
*/filename=item.Name ,folder=item.Path }, new AjaxOptions { HttpMethod = "GET" }) |
        @Html.ActionLink("Preview", "GetPreview", new { /* id=item.PrimaryKey */ path=
item.Path }) |
        @Ajax.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ path= item.Path},
new AjaxOptions { HttpMethod = "GET" })
    </td>
</tr>
}
</table>
```