Introduction

So far we've covered the following topics in the Dropbox series:

- Part 1: Authentication
- Part 2: API Requests
- Part 3: Create, Delete and Move Folders

Once you are authenticated you can make API requests such as requesting your account information, creating, deleting folders...etc. One particular type of request is downloading a file from your Dropbox account. Once you've worked your way through the previous 3 parts this becomes trivially easy.

If you want to follow along go to the download page and download the code for the third part (article #65). Unzip it and open it up in Visual Studio.

Table Of Contents

A New Method

Let's add a new method to the DropboxApi type called DownloadFile(...).

```
public FileSystemInfo DownloadFile(string root, string path)
{
    // ...
}
```

It takes two parameters, namely:

- **root**: The root relative to which the path is specified. Valid values are sandbox and dropbox.
- **path**: The path to the file you want to download.

It returns an instance of the FileSystemInfo type, which we created earlier in part #3. The FileSystemInfo type contains the file's metadata.

Example usage of the new method:

```
var api = new DropboxApi(ConsumerKey, ConsumerSecret, accessToken);
var file = api.DownloadFile("dropbox", "Public/DropboxPart3.zip");
file.Save(@"C:\DropboxPart3.zip");
```

Top of page

Expanding FileSystemInfo

Before we can proceed we need to modify the existing FileSystemInfo type.

```
public class FileSystemInfo
{
    //...

    public byte[] Data { get; internal set; }

    public void Save(string path)
    {
        using (var fileStream = new FileStream(
            path, FileMode.Create, FileAccess.ReadWrite))
        {
            fileStream.Write(Data, 0, Data.Length);
        }
    }
}
```

The new property Data returns a byte array which contains the contents of the file we downloaded. The Save(...) method allows you to easily save the file. No world shocking news here.

Top of page

Downloading the File

Let's implement the DownloadFile(...) method. When downloading a file we need to contact the API content server. So let's add a URL to the DropboxRestApi type.

```
public class DropboxRestApi
{
    public const string ApiVersion = "1";

    public const string ApiContentServer =
        "https://api-content.dropbox.com/" + ApiVersion + "/";

    //...
}
```

Now we can start with implementing the DownloadFile(...) method. First we need to compose a URL for the file which we want to retrieve.

For example:

https://api-content.dropbox.com/1/files?root=dropbox&path=Public/DropboxPart3.zip

```
var uri = new Uri(new Uri(DropboxRestApi.ApiContentServer),
    String.Format("files?root={0}&path={1}",
    root, UpperCaseUrlEncode(path)));
```

Remark: We created the UpperCaseUrlEncode(...) method last time to circumvent a rather annoying problem when URL encoding the path.

Next we need to sign the request.

```
var oauth = new OAuth();
var requestUri = oauth.SignRequest(uri, _consumerKey, _consumerSecret,
    _accessToken);
```

Then you can finally execute a GET request to download the file.

```
var request = (HttpWebRequest) WebRequest.Create(requestUri);
request.Method = WebRequestMethods.Http.Get;
var response = request.GetResponse();
```

The HTTP response contains the file's metadata in JSON format within an x-dropbox-metadata header. Let's extract this metadata from the headers and deserialize it into a FileSystemInfo instance.

```
var metadata = response.Headers["x-dropbox-metadata"];
var file = ParseJson<FileSystemInfo>(metadata);
```

Now we can read the response and store the file as a byte array in the FileSystemInfo instance.

```
using (Stream responseStream = response.GetResponseStream())
using (MemoryStream memoryStream = new MemoryStream())
{
    byte[] buffer = new byte[1024];
    int bytesRead;
    do
    {
        bytesRead = responseStream.Read(buffer, 0, buffer.Length);
        memoryStream.Write(buffer, 0, bytesRead);
    } while (bytesRead > 0);

    file.Data = memoryStream.ToArray();
}
```

And the only thing that remains is to return our FileSystemInfo instance.

```
return file;
```

Voila, that's the whole implementation of the DownloadFile(...) method.

Thanks to the fact that we did most of the work during the previous parts, this turned out to be a rather short article. Check out the Dropbox REST API documentation if you want to explore it further and implement new features on your own.

https://www.dropbox.com/developers/reference/api

In part 5 I'll discuss how you can upload new files to your Dropbox account. You can download the source code accompanying this article from the download page. If you have any questions or suggestions please drop me an e-mail or submit a comment.

Top of page