# Lab: Asynchronous Programming

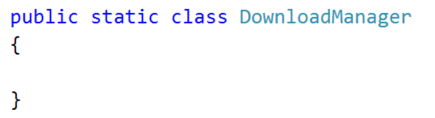
In our final piece of the **BashSoft** we are going to implement functionality for downloading a file from the internet using asynchronous and synchronous methods. First we are going to implement the synchronous one and after that reuse it, to make it asynchronous.

## Start Implementing the Download Method

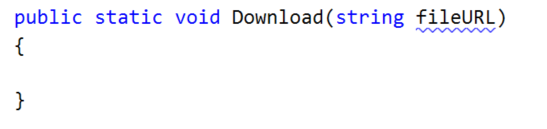
Let’s first start by making a new folder called **Network** in our project with a new public static class called **DowloadManager**:



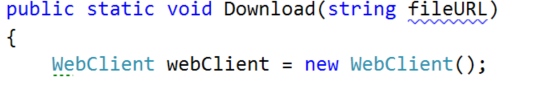
…



All we need as information in order to download a file is the **URL** to the specified file. That’s why we are going to make a public static void **Download** method with one input parameter which is the **fileURL**:



Since we are going to make a communication with the Internet, we are going to have an API that gives us such functionality. A class that gives us easy access to the **WWW** is the class **WebClient** class which can be found in the **System.Net**. It is not a static class and for that reason we are going to have to make a new **WebClient** in our method.

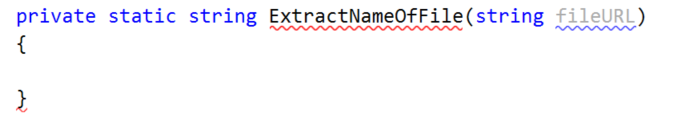


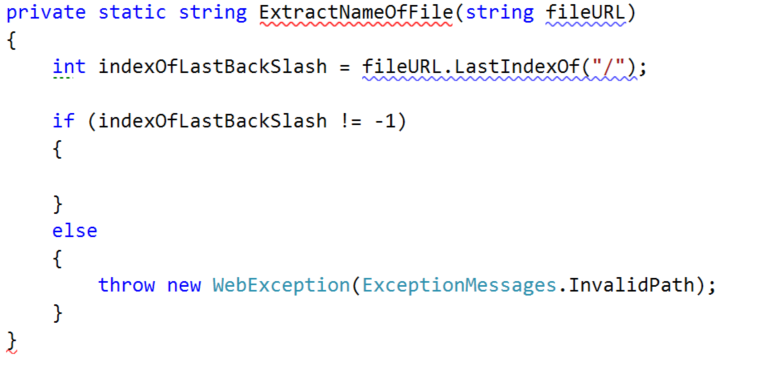
We can use the method **DownloadFile** from the **WebClient** that takes as a first parameter the **URL** to the image and as a second parameter, the path + file of the local machine. We can easily get the current path on the PC, but first we need to be able to extract the name of the file from the **URL**, so that we can use it to save the file with the same name on our PC.

For that reason we should make a new method called **ExtractNameOfFile** that is private static, returns a string (the actual file name) and take as a parameter the **fileURL**.

## Extract Name of File from URL

This is how the method should look before the declaration of its implementation:

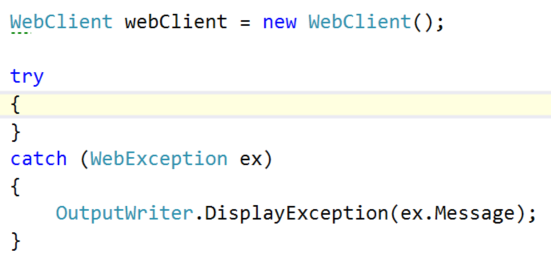


All we need to do it take the index of the last forward slash, because it separates the file name + extension from the folder it is in, on the server. Since the result can be either a non negative number if the **fileURL** contains such a character, or **-1** if it does not, we want to make sure that we do the job only when the index is correct and otherwise throw a new **WebException** with a the message for invalid path.  
  


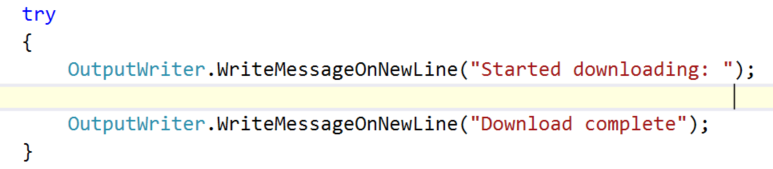
Our method is still underlined, because we haven’t returned a value for every case of our program, so let’s do this and go back to the implementation of the **Download** method. All we need to do is take everything from index found + 1 till the end of the **URL** and use it as a file name.



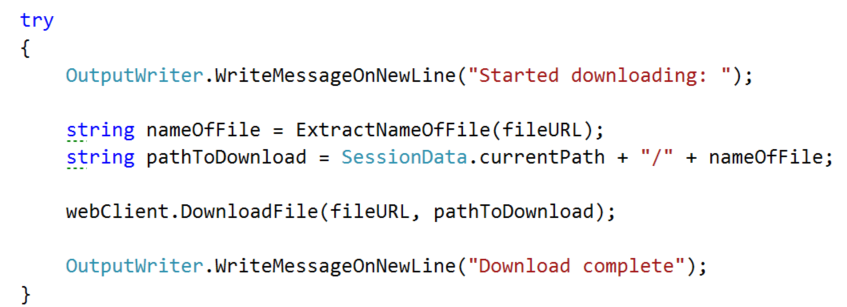
## Finish Implementing the Download Method

Since we are going to use the **ExtractNameOfFile** and it can raise an exception, we would like to put all the functionality that we are going to write in a try block and in the catch block we are going to catch a web exception and display its message using the **DisplayException**.  
  


On the first line **in** the **try** **block** we should **indicate** the **user** that we’re **starting** **to** **download** the file and **in** the **last** **line** in the try we should **indicate** the **user** that the **download** is **complete**.



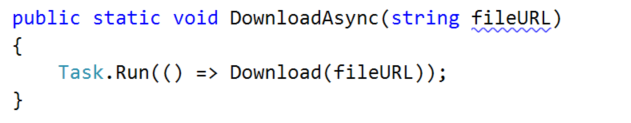
Between the two messages we should save the extracted name of the file, after that generate the local path + the name of the file and finally pass the **fileURL** and the **local** **path** **+** the **name** **of** the **file** to the **DownloadFile** method of the **webClient**:



There are other possible exceptions that a could be raised, but most of them are **WebExceptions** so we can leave it to be the only **exception** **cached**.

## Implement Asynchronous Downloading

Now that we have implemented the **Download** of a file in **C#** we can easily extend it to be asynchronous, by running it on another thread as a separate task. Since the declaration of the method is pretty close to the previous, here is how it should look:



We are not going to go into depth of how the **Task** should be used properly, because that is not the most appropriate way to be used, but that needs using **OOP** to some extent.

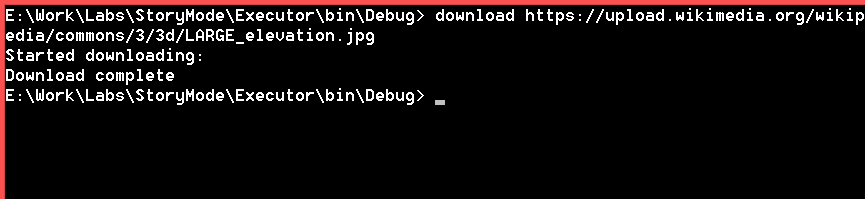
## Implement Calls from the Command Interpreter

The implementation of the functions that call the **DownloadManager** is pretty common to all others in the current class. We check **if** the **number** **of** **parameters** is **2** and if so, **pass** the **second** **parameter** to the corresponding method.



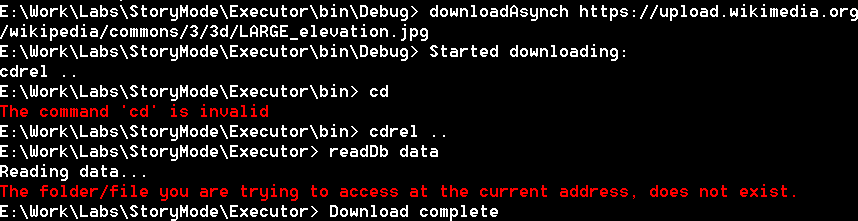
After that all we should do is **call** **them** **in** the **appropriate** **place** **in** the **InterpredCommand** method.

Finally let’s test the functionality:

**Synchronously:**  


We cannot change our position in the file system while downloading.

Asynchronously:

  
We can do much more stuff while downloading the file.

This is everything from the functionality that we are going to implement in the current course. Since many of the things that we did are quite smelly as a code, it will be our duty to fix those pieces of code and extend the functionality even more.