Making a HTTP request and listening its completion in Windows Phone

This article demonstrates how to make HTTP requests using both WebClient and HttpWebRequest and waiting for their completion.

Introduction

One of the most important aspects of smartphones is connectivity and data management. Due to the constraints of processing power and available storage, lots of applications for mobile devices depend on connecting to a remote web service and download the necessary data, be it the result of a search query or a simple image. In this article, we are going to demonstrate how to use both `WebClient` and `HttpWebRequest` to contact a remote server and download string and binary data.

Using WebClient

First of all, `WebClient` is more limited in Windows Phone than in other .NET versions. We can only use the `DownloadStringAsync` call to download a text-based resource, like the source code of a web page or a XML or JSON based result of a server operation. The good news is that this call is asynchronous and we can subscribe to two events to get the current progress of the operation and the full downloaded data, `DownloadProgressChanged` and `DownloadStringCompleted`, respectively:

```
webClient = new WebClient();
webClient.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(webClient_DownloadStringCompleted);
webClient.DownloadProgressChanged += new
DownloadProgressChangedEventHandler(webClient_DownloadProgressChanged);
webClient.DownloadStringAsync(new Uri(downloadUrl));
```

The callback for `DownloadProgressChanged` receives a parameter of type `DownloadProgressChangedEventArgs`, which has the following properties:

- **long BytesReceived**: number of bytes currently downloaded.
- **int ProgressPercentage**: a value between 0 and 100 indicating the progress percentage of the download.
- **long TotalBytesToReceive**: total size in bytes of the string to download.
- **object UserState**: a custom user object to store additional info.

Warning: **TotalBytesToReceive** and **ProgressPercentage** properties will only work correctly if the server specifies the **Content-Length** header in the response.

And the callback for `DownloadStringCompleted` has a parameter of type `DownloadStringCompletedEventArgs` with the following properties:

- **bool Cancelled**: specifies if the operation was cancelled by calling `WebClient.CancelAsync`.
- **Exception Error**: an exception which includes any errors that could have happened.
- **string Result**: contains the string downloaded from the remote server.
- **object UserState**: again, a custom user object to store additional info.

Common problems when using WebClient

- Sometimes you will get a **NotFoundException** thrown. This can happen by a wide list of causes, from a non-standard header in the response to trying to connect via secure HTTP and providing bad credentials.
- If you have an HTTP sniffing tool like Fiddler2 running and you test your application in the emulator, the above explained **NotFoundException** problem can happen too.
- You don't have to `Dispose` or control its deletion in any way; `WebClient` cleans itself after performing the requested actions.

Using HttpWebRequest

This is a more complex, but at the same time, more customized way of downloading both string and binary data. For example, you can call `BeginGetRequestStream` to obtain a writeable stream if you want to perform a **POST** operation, so you have a place where

you can write the data you are sending in the HTTP request. If you want a standard **GET** request, you call `BeginGetResponse` and pass as its parameter a new `AsyncCallback` containing the callback function that will handle the logic of the operation:

```
webRequest = (HttpWebRequest)HttpWebRequest.CreateHttp(imageUrl);
webRequest.BeginGetResponse(new AsyncCallback(ResponseCallback), webRequest);

...

private void ResponseCallback(IAsyncResult asyncResult)
{
    HttpWebRequest webRequest = (HttpWebRequest)asyncResult.AsyncState;
    HttpWebResponse webResponse =
(HttpWebResponse)webRequest.EndGetResponse(asyncResult);

    MemoryStream tempStream = new MemoryStream();
    webResponse.GetResponseStream().CopyTo(tempStream);
}
```

Common problems when using HttpWebRequest

- Windows Phone has an internal mechanism that caches web request, and this affects both `WebClient` and `HttpWebRequest`. This can lead to receiving the same result when querying the same URL twice, even if the response should be different. To solve this, add a random parameter to the end of the URL that you know won't affect the response of the web server in any way.
- If you are using it in **POST** mode, remember to call `Close` of the stream that returns `EndGetRequestStream` inside the callback for `BeginGetRequestStream`; otherwise, a race condition exception will be thrown.
- If the server supports it, you can receive a compressed response stream to save extra bandwidth. Just make sure you call `{{{1}}}` before performing the request and, and the response stream will be automatically GZip-compressed (but remember to decompress it before using the data!).

Source code

You can download an example project showing how to use both `WebClient` and `HttpWebRequest` here: Media:HTTPRequests.zip.