Subject: Re: FTP class and reference example for U++
Posted by Klugier on Wed, 12 Apr 2017 21:05:33 GMT
View Forum Message <> Reply to Message

Hello,

I am not tested this package, but i would like to notice small issue with the API design. The problem is that you need to pass too many arguments to the single function, if you want to change single argument like ascii. Let's take a look at below function:

```
int FtpAsyncGet(const String& remote_file, const String& local_file, const String& host, int port = 21, const String& user = Null,
	const String& pass = Null, Event<int, int, String, int64, int64> progress = CNULL, bool active = false, bool ssl = false,
	bool ascii = false);
```

This is the poor design, because the call will look like this:

```
FtpAsyncGet("remote.txt", "local.txt", "locaclhost, 21, Null, Null, CNull, false, false, true);
```

Looks terrible. Isn't it? In my opinion the good API has got maximum three parameters - otherwise it becomes hard to use by the client.

We can replace following code with the object approach (This is the example and I highly recommended to over thing this design):

```
int FtpAsyncGet(const FtpAssyncRequestConfig& config);

FtpAsyncGet(FtpAssyncRequestConfig("remote.txt", "local.txt", "locaclhost.txt").EnableAscii());

class FtpAssyncRequestConfig final
{
public:
   FtpAssyncRequestConfig(const String& remote_file, const String& local_file, const String& host)
	: // initialization ...
   {}

   // Get, Set, Enable interface...

private:
   // private members
};
```

Much more easier to set the n-th parameter, but require additional work for the library provider.

_____

You could replace NAMESPACE_UPP with namespace Upp { and END_NAMESPACE_UPP with }. The NAMESPACE_UPP approach was deprecated since last release.

_____

The next problem I see in the code is the ordering problem - you should put public class interface at the top then private things (Ftp and DirEntry classes). This is the general rule mentioned in following documentation topic.

_____

This code review aims to improve the code quality.

Sincerely,
Klugier
