Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Sun, 16 Apr 2017 16:28:37 GMT
View Forum Message <> Reply to Message

Quote:Hello,

In my opinion handling multi-threading API using #ifdef in header is not good for the end user. Because, he/seh is obligated to use #ifdef in theire code. Personally, i would hide it inside the interface and use One container (Movable will not work for this class).

```
class Result // Interface - all methods are abstract
{
public:
  // The same as previous

  virtual bool IsAsync() = 0;

  // Mt methods are presented without #ifdef guard
};

class RegularResult : public Result // The name could be different
{
public:
   bool IsAsync() override { return false; }

   // Mt methods returning falses, -1 etc.
};

class AsyncResult : public Result
{
public:
   bool IsAsync() override { return true; }

   // Return correct values for MT
};

One<Result> result(new AsyncResult());
```

What do you think?

Sincerely,
Klugier

Hello Klugier,

Thank you very much for your constructive criticism and suggestions.

I believe we might have a simpler solution. :)

FtpAsyncGet() and FtpAsyncPut() functions call FtpAsyncIO().It is possible to return en error code if anything goes wrong.
So we can simply define a function, say, IsMultitihreaded(), to check the U++ multi-threading infrastructure automatically, in Ftp.cpp:

```
static inline bool IsMultithreaded()
{
#ifdef flagMT
 return true;
#else
 return false;
#endif
}
```

Then we can call it in FtpAsyncIO:

```
Ftp::Result FtpAsyncIO(Ftp::Request request, Event<Ftp::Result> progress, bool put)
{
  // Check if U++ MT is enabled.
 Ftp::Result r;
 try {
  if(!IsMultithreaded())
   throw Exc("Multithreading is not enabled.");

// ...

 }
 catch(Exc& e) {
  // Couldn'r create or run the worker thread.
  r.info("state") = "failed";
  r.info("rc")    = -1;
  r.info("msg")   = e;
  LLOG("-- FTP worker failed. Reason: " << e);
 }
 return r;
```

In this way we can remove other ifdefs, and user won't be confused.
(Multithreading as a requirement is already mentioned in API docs for each method and function.)

Note that I also changed the return value from int to Ftp::Result. (Now they are similar to single-threaded variants)

Calling an async function such as Result::InProgress() now won't do harm, it wil silently return false.

AS for your other suggestions:

ParseFtpDirEntry() actually modifies the ValueMap. It parses the string into key-value pairs. However, now I put the actual parser code into DirEntry::Parser() method, and got rid of the friend declaration.

Same thing goes for the Request class. So, the ugly forward declarations are removed.

I updated the package to reflect the changes...

Regards,

Oblivion