Subject: Re: RE: Job package: A scope-bound worker thread for non-blocking operations.
Posted by Oblivion on Tue, 10 Oct 2017 09:51:00 GMT
View Forum Message <> Reply to Message

A further refinement would be to explicity relate CoWork to promise/future pattern:

My suggested namings:

-
- WorkResult (former WorkEntry): Unlike std::promise, this is called privately. I don't really see any use in calling it explicitly
- Work (former WorkResult): This can be the U++ std::future counterpart (see the first prototype packageI provided above) This can be a helper class to CoWork, representing a single, isolated (semantically) thread.

- CoWork::Work(): This is the thread-starter method which will return, well, Work :)

I propose adding CoWork::Work() as a (non-static) method to CoWork, because from my experience with Job,Thread, future/promise and CoWork, and as I noted on my previous message, keeping workers contained in CoWork instances, using a CoWork::Do() call, have some advantages over using a static method such as CoWork::Schedule(): Such as the ability to use CoWork::Finish() on demand, and waiting the workers to be finished automatically on class instance destruction.

As for the error management mechanism using a specific exception type such as Work::Error, along with IsError(), GetError() and GetErrorDesc() methods, IMO they would be a very useful addition, but they are not necessary, can be removed.

What do you think?

Best regards,
Oblivion