Subject: Re: Job package: A lightweight multithreading tool, using promise/future mechanism
Posted by Oblivion on Sun, 10 Sep 2017 19:16:13 GMT
View Forum Message <> Reply to Message

Hello Mirek, and thank you for your patience.

Quote:mirek
OK, so the difference is

- "return" semantics (implemented using promise/future)
- error states

Right?

If true, what about using future / promise directly? IMO the only problem is to have them interfaced with Thread and/or CoWork (for different kinds of usage).

Well, Job is one way of having them interfaced with U++ Thread. Using f/p pair directly is somewhat cumbersome.
However, TBH, what I really see lacking in multithreading tools in general is a simple error handling mechanism, and a per-thread shutdown mechanism.
This is the one of the reasons why I've come up with the Job class. It is also an attempt to solve these problems.

E.g.
Thread.IsError()
Thread.GetError()
Thread.GetErrorDesc()
Thread.ShutDown()
and
Thread::Error (exception)

As you can see in the Job.hpp these are not very hard to implement. (I don't see why it should be difficult for Thread at least (performance?). Cowork, being a job queue, is naturally a more complex beast.)

Best regards,
Oblivion