Subject: Re: RE: Job package: A scope-bound worker thread for non-blocking operations.
Posted by Oblivion on Mon, 18 Sep 2017 06:29:46 GMT
View Forum Message <> Reply to Message

Quote:That's intereseting, because I actually suggested adding future/promise to U++ facilities... Smile

Anyway, I might check it, but I do not see any archive / link posted.

Mirek

That can also be done, I can work on it separately, or re-introduce it as an option if you think it is worth it. (In it's current state Job package does not cover a shared_future alternative. Maybe I should consider adding a SharedJob class, but 1) there is already CoWork for paralellization, 2) I don't quite know a real life scenario where it would be more useful than CoWork.)

Nevertheless, recently I've found my "reference based" approach much more performant and convenient than future/promise.

Consider this (pseudo) code:

```
double sum_total = 0;
Job<Vector<double>> job;
job.Start(DoSomeHeavyCalculationAndFillTheVector);

//....

if(job.IsFinishe() && !job.IsError()) {
   for(auto& e : ~job)          // Reference access to Vector. Data is not copied or moved. Vector is filled via a direct -yet safe- access.
      sum_total += e;           // And retrieved in same fashion as well. Note also that there is no requirement for the data type to be contained.
                                // In fact, using One or Any, or Value, we can even differentiate between the job
                                // results for given operations in a single Job easily. :)
}
```

Package's link is on the bottom of the first post of this topic.

Best regards,
Oblivion