Subject: Re: CoWork buggy!?
Posted by Werner on Sun, 23 Mar 2008 10:38:38 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 March 2008 08:18Well, not sure if I fully undestand the complaint, but if the issue is that Finish does not terminate threads, it is exactly what we wanted. Once created, threads are terminated only at app exit.

The reason is obvious - creating / terminating threads is somewhat expensive. So we have a pool of worker threads ready when next CoWork hits. And they are shared across all CoWorks as well, even nested.

Mirek
Ok, maybe I misunderstood the idea of "CoWork" . If so, sorry for that . After all there isn't any documentation and the code is written in a style which - well, let's say - I'm not familiar with .

Anyway, this raises bags of questions of which the following 2 are my favorites:

1.

Under which conditions should "CoWork::Finish" be called?

2. (independent from the answer to question #1)

What is "CoWork::waitforfinish" for?

If "CoWork::Finish" is called there are 2 possibilities:

a) no waiting job, i. e. "todo == 0":

Nothing is done. "CoWork::waitforfinish" remains unchanged.

b) waiting job, i. e. "todo > 0", e. g. "todo == 1":

That means "Pool:jobs.GetCount == 1" too, because both "todo" and "jobs" are incremented by "CoWork::Do" and decremented by "CoWork::Pool::DoJob". That again means "CoWork::Pool::DoJob" increments "CoWork::waitforfinish".

The conclusion is: "CoWork::waitforfinish" is never decremented and might raise without limit .

Werner