Subject: Re: Select Grid Row BY ID
Posted by germax on Sun, 14 May 2017 17:48:31 GMT
View Forum Message <> Reply to Message

Again, thank you,
currently looking at the changes.. apart from the finished flag and the reverse order of the loop (counting down, which indeed does make sense in this case)
I don't see a major difference yet *shrugs*
BTW I just counted up for the more "prominent" visuals [aka seeing the grid update immediately ;))

I'll give it a shot and see where it get's me.

I know asynchroneous is the goal of all multithreading essentially;
but since I "pre-process" the same data that I later touch with the actual processing, asynchroneous wouldn't help at all without
having the precheck and process in the same Method or passing the checked items directly to the processor instead of reiterating over the grid later on.
In which case maintaining either will be more confusing in the future (as if I could remember next year what I did today ;)).

And the precheck must remain optional
(since it basically doubles the execution time and is only necessary with certain files;)
it cannot just be part of the processing (which then of course can be asynchroneous as intended)
So the precheck is synchronised by design
(for convenient maintenance purpose, and not to pass the prechecked items to the prosessing individually)

Anyways..
I know it's basically pointless to even start a seperated thread when syncing anyways.
non blocking code was my main goal initially (and that you solved already :d)

NOW true multithreading became my goal for speed reasons just recently.
It works well enough with files up to 500 items or something without any additional worker thread at all.
(as long as the mysql server sees only low traffic and load it's not even necessary to think about improving it)
The thing is, as soon as the server load increases, the response times drop..
At which point I'd rather make use of the ability to check not one but ten or thirty or fifty lines at a time, to make use of all the allowed mysql connections the current user is having;)

IDK if it'd improve the execution time by much, but with large files and on busy days, every little bit helps.

The single worker is nice to have,
for me in this particular usage case it's nothing but a lesson really.
You are right, it does not make much of a sense in this app at all.
But the more I know now, the less I need to ask about later, right? :d

hehe I forgot about the Upp csv parser.. I use my own in projects like this mainly... :blush:
I just removed it since it's rather large and uneccesary for the fixed file at hand ;)

NOW..
since I just tested your alterations while typing,
I'm still getting memory leaks like crazy *shrugs*
Quote:
Heap leaks detected:

```
--memory-breakpoint__ 137368 : Memory at 0x06750290, size 0xA0 = 160
    +0 0x06750290 03 00 00 00 97 00 00 00 73 65 6C 65 63 74 20 2A     ........select *
   +16 0x067502A0 20 66 72 6F 6D 20 4F 55 54 42 20 77 68 65 72 65      from OUTB where
   +32 0x067502B0 20 4F 52 44 45 52 49 44 20 3D 20 27 31 32 33 33      ORDERID = '1233
   +48 0x067502C0 36 20 50 50 27 20 61 6E 64 20 49 54 45 4D 20 3D     6 PP' and ITEM =

...................

*** TOO MANY LEAKS (274) TO LIST THEM ALL
****************** PANIC: Memory leaks detected! (final check)
```

tested on both desktop pcs so far
(one trusty old amd 3000+ single core 2GB ram and a dual 8core xeon with 88GB ram)

But wait.. I compiled in MinGW (32bit on the single core and 64bit on the 16core)
You MSVC'd maybe?