Subject: Select Grid Row BY ID
Posted by germax on Fri, 05 May 2017 17:32:46 GMT
View Forum Message <> Reply to Message

Hi, got a severe problem atm which needs to be solved by tomorrow :(

It's this:
I have a gridGrtl which is filled with csv values..

Now I need to "check" it's entries for duplicates against a database (external mysql)
so far so good, in single threaded mode I just go through each rows backwards
(starting from the last)

remove each row which is a duplicate until I hit row zero.

With a local db that's reasonably fast and everything is great..
with a remote db that takes a loong time, during which the GUI tends to blank out
not refreshing properly and basically looking "crashed"

So I added a CoWorker to do all the checking against the MySQL.
And that just doesn't want to work out for me..

In that worker-function
If I use the row-id as a reference (grid.GetRow(rownum))
since the multiple threads are not "ordered by descending rowids" the rowid of a specific row
changes as a previous thread deleted a row... causing all sorts of errors occasionally (by chance)

if I use the rows preset ID (grid.AddIndex(ID);)
and accessing the row by ID (int rownum = grid.Find(rowid, ID); grid.GetRow(rownum))
the program compiles flawlessly but terminates with a runtime error
(I assume the row Id changes during execution since a concurrent thread deletes a row)

it's a sheer PITA to not being able to just grid[rowid].doStuff() with a ficxed rowid not irritated by
sortorder or number of rows or position of row on the grid...

Any advice?

When I use a more linear MT model (say the one of the

Subject: Re: Select Grid Row BY ID
Posted by deep on Mon, 08 May 2017 13:07:34 GMT
View Forum Message <> Reply to Message

Hi,

May be you can add one more column to mark row as deleted.
Actual delete after complete processing.

Quote:
if I use the rows preset ID (grid.AddIndex(ID)Wink
and accessing the row by ID (int rownum = grid.Find(rowid, ID); grid.GetRow(rownum))
the program compiles flawlessly but terminates with a runtime error
(I assume the row Id changes during execution since a concurrent thread deletes a row)

it's a sheer PITA to not being able to just grid[rowid].doStuff() with a ficxed rowid not irritated by sortorder or number of rows or position of row on the grid...

So let me get this straight:

You are trying to access and manipulate a Grid object (in which you store some information) from within worker threads, right?

If this is the case, the first rule you need to remember is that GUI related stuff in U++ should be done in the main thread.
Hence you'll need serialized access. And if you want serialized access to gui elements from within threads, you should consider using PostCallback() function.

GuiMT example in the U++ reference examples in principle demonstrates this behaviour.

E.g.

If you are going to do stuff in a Grid, you can:

```
// Let us assume that we've defined a Grid object as grid in MyApp;
// Then in your worker thread you can call PostCallback().
// Of course you'll need a more complicated version of control code,

//WorkerFoo() represents a  worker thread function.

MyApp::WorkerFoo()
{
	auto rowid = 1;
	PostCallback( [&] { grid(rowid).DoStuff();  } );
}
```

Of course you need to design your code keeping in mind that removing elements from arrayctrls

and grids invalidate references and pointers. (see docs and examples)

Regards,

Oblivion.

Subject: Re: Select Grid Row BY ID
Posted by germax on Wed, 10 May 2017 10:33:18 GMT
View Forum Message <> Reply to Message

Hi,
thanks for your answers..

I already removed all GUI stuff from the worker threads,
I created a boolean Vector which entries (well pointers to it's entries)
are passed to the worker threads to know which line needs to be deleted when I return to main thread ;)
And apart from me forgetting to wait for the workers to finish at first (ooops)
that part is working really well...

The problem (the reason why I went with multithreading in the first place) is still persistent.
the GUI blanks out and the App is irresponsive for quite some time (until all workes are finished)

I narrowed it down (I think)
I made a simple testapp as an example

IDK.. it's still not responding as long as it's "working"
(and yes, compiled as GUI MT)

shrugs

There is something I miss ... but I do not see it.. (I still don't for some reason)

PS any three column csv as long as last column is numeric should do

PPS oh .. I just recognize there's a whole chunk missing in my initial post ... uhm I don't know how tbh.. and frankly I cannot even recall what should've been there... Sorry about that!

File Attachments
1) MTtest.zip, downloaded 246 times

Subject: Re: Select Grid Row BY ID
Posted by Oblivion on Wed, 10 May 2017 17:09:26 GMT
View Forum Message <> Reply to Message

Hello germax,

Quote:The problem (the reason why I went with multithreading in the first place) is still persistent. the GUI blanks out and the App is irresponsive for quite some time (until all workes are finished)

Do you really need CoWorker? I don't know the details of your app, but the example you provided is somewhat complicated, imo.
Spawning CoWorkers per row, as your code seems to do, would be an overkill with a considerable overhead.
If all you need is an asynchronous (non-blocking) gui, you can easily use a single worker thread (Thread, not CoWorker) and get rid of the additional vector.
Please find attached a simple example with a csv file over 6000 lines. It simply removes entries one by one without blocking the gui. And it also demonstrates how to halt a thread.
It'll give you the idea, I hope. :)

The thread code will look like:

```
void MTT::processFile()
{
 int rowc = grid.GetRowCount();
 progin.Percent();

        // Below is the actual thread function. We are taking advantage of C++11 lambdas here.
 Thread().Run([=] {
  for(auto i = rowc - 1, j = 0; i >= 0; i--, j++) {
   if(IsShutdownThreads())
    break;
//   int q = grid(i, 2);
//   if(q <= 0)
//    this->RemoveItem(i, j);  // You can also use a dedicated method.
    PostCallback([=] {
     grid.Remove(i);
     progin.Set(j, rowc - 1);
    });

   Sleep(10);
  }
 });
}
```

Actually you may even not need multithreading at all.
if you are using a for/while loop in single threaded environment which thakes time and thus blocks your app, I mean if that's the only problem, you can use Ctrl::ProcessEvents() to refresh the GUI

inside your loop.

E.g.

```
for(auto i : very_large_vector_to_process) {

 DoYourStuff();
 ProcessEvents();
}
```

Regards,

Oblivion

File Attachments

1) MTtest.zip, downloaded 246 times

Subject: Re: Select Grid Row BY ID
Posted by germax on Thu, 11 May 2017 01:33:14 GMT
View Forum Message <> Reply to Message

THANK YOU! (this forum needs a thank you button ;))

I've tried for days, and I've also tried a single thread;
with just a slightly different layout (again passing boolean pointers to NOT update the gui from within said thread)
but it wasn't invoked using thread; it again was basically the same as my mtt with just the for loop moved inside the worker.

I will try that next (tomorrow, at 3 am my fingers get laggy and my brain.. well... isn't up to par either let's say ;))

Your example runs perfectly even on my trusty old single core 32bit xp-box..
(which is amazing compared to the failure of MY example on even a 16core (yes physical cores) 64bit win7 box)

My example is a stripped down version of the original program I'm working on,
it's 2k+ lines of code, and I just kept more or less the same layout the original is dealing with,
my actual worker has ~130 lines
(test against db for perfect matches.. show closest matches where necessary etc. and deal with the rest)
so a lambda wasn't actually a good idea (without code collapse that'd be insane to read next week

LOL)

Anyways, THANKS A MILLION!!

Your example is just what I wanted.. it doesn't explain to me why mine didn't work,
but all I care about now is that it DOES work one way or another :d so I'm good.

Subject: Re: Select Grid Row BY ID
Posted by Oblivion on Thu, 11 May 2017 18:04:35 GMT
View Forum Message <> Reply to Message

Hello germax, and you're welcome.
Ultimate++ forums are full of helpful people. :)

Quote:Your example is just what I wanted.. it doesn't explain to me why mine didn't work,
but all I care about now is that it DOES work one way or another Very Happy so I'm good.

Sorry I forgot to write that, The reason is actually pretty simple. You use CoWork::Finish(), which is blocking.
You should have used CoWork::IsFinished() instead.
In that case, all you need to call is Ctrl::ProcessEvents(). But still, CoWork seems to me an overkill here. :)

All you need to do is to alter your code as below (Of course you don't need to use lambda functions and while in the example accessing the vector from threads does not pose classic conncurrency problems, you should consider using mutex (CoWork::Finlock() will do fine. See docs and tutorials for more explanation.):

```
void MTT::processFile()
{
 int rowc = grid.GetRowCount();
 Vector<bool> del;
 del.SetCount(rowc, false);
 progin.SetTotal(rowc);
 progin.Percent();

 for(int i=0; i < rowc; i++)
  cw & [=, &del] {
   // Now, this isn't the proper way to accaess shared elements (UI or core) from a
   // thread. But ONLY in this example, it won't do harm.
   grid.GetRow(i);
   int q = grid(i,2);
   if(q <= 0)
//   CoWork::FinLock();
   del.Set(i, true);
  };
 // cw.Finiehed()
```

```
 // Non blocking way:
 while(!cw.IsFinished())
  ProcessEvents();

 for(int i = rowc-1; i>=0; i--)
 {
  progin++;
  if(del[i]) grid.Remove(i);
  ProcessEvents();
 }
 RDUMP(del);
}
```

This is all you need to do. :)

Regards.

Oblivion.

Subject: Re: Select Grid Row BY ID
Posted by germax on Thu, 11 May 2017 18:17:07 GMT
View Forum Message <> Reply to Message

Ah well..
yes, isFinished instead.. with a processEvent... checked (I take notes :d)

The rest :(
This is just insane....

I tested with such loop:

```
for (int i = 0; i < grid.GetCount(); i++)
{
 grid.GetRow(i).Bg(LtGray());
 String item = grid(i,5);
 if (ToUpper(TrimBoth(item)).StartsWith("SHIPPING"))
 {
  continue; // skips the SQL query and i gets increased by the for loop
 }
 SQL * Select(SqlAll()).From(Whatever).Where(....);
 if(SQL.Fetch()){//some if then elses again }
}
```

And "inline" it works, it continues happily and leaves me with the correct iteration of i.

THIS however:

```
Thread().Run([=] {

for (int i = 0; i < grid.GetCount(); i++)
{
 grid.GetRow(i).Bg(LtGray());
 String item = grid(i,5);
 if (ToUpper(TrimBoth(item)).StartsWith("SHIPPING"))
 {
  continue; // will be ignored for some reason *SHRUGS*
 }
 SQL * Select(SqlAll()).From(Whatever).Where(....);
 if(SQL.Fetch()){//some if then elses again }
}

}
```

Just ignores the continue, sends an SQL query and as a result produces all kinds of errors
Worse:
correctly instead of a simple continue it holds a rowdelete

```
if (ToUpper(TrimBoth(item)).StartsWith("SHIPPING"))
{
 grid.Remove(i);
 i--;
 continue;
}
```

works just nicely

```
if (ToUpper(TrimBoth(item)).StartsWith("SHIPPING"))
{
 PostCallback([=] { grid.Remove(i); });
 i--;
 continue;
}
```

removes i-1 instead.
 if there is i-1... if i was zero before I get Vcont.h line 84 Assertion Errors
of course (grid.Get(-1) crashes)
as if i is passed as a reference.

You see me completely baffled!

Ah well.. let's see if a while loop is more reliable inside the thread()...
Or do you have another idea?

Subject: Re: Select Grid Row BY ID
Posted by Oblivion on Thu, 11 May 2017 18:57:30 GMT
View Forum Message <> Reply to Message

It looks like a concurrency proeblem (a serialization issue) from here. But hard to say...
Are you running multiple worker threads? Is the method where your thread code resides executed more than once?
IF so, one thread may be altering the grid or SQL db (removing an entry and invalidating the iterators/indices), while the other one sleeps.
This can happen any time, right in the middle of your loop if you don't serializa access to shared data using mutexes.

Or it might be that the worker threads outlive the data objects (grid, or SQL).
For the sake of simplicity I'll give you a minimal example.
The following code MAY or MAY NOT lead to crash, for we can't be sure how long the thread will take to finish.

```
MyApp::MyMethod()
{

    String text; // String will be destroyed when when MyMethod returns().

    Thread().Run([=] {

      // Do someting that takes time.
      text = "hello world";         // This may or may not lead to crash. Because the thread may outlive the text object and MyMethod.
    });



}
```

So you'll need to take into consideration all kinds of concurrency problems.

If you can provide a simple example (and a very small dummy database) reproducing the crash, I'll loook into it and try to help.

Regards,

Oblivion.

Oblivion wrote on Thu, 11 May 2017 20:57It looks like a concurrency proeblem (a serialization issue) from here. But hard to say...
Are you running multiple worker threads? Is the method where your thread code resides executed more than once?
Just a single worker thread and that code is UNLIKELY to be run twice by accident really
(the calling button is the first thing to be disabled to prevent executing it twice with the same csv file.)
But frankly I'm by no means sure that there is no second instance..
The debugger only shows two threads though (the main one and my worker)
so I'm at least pretty confident there is no second instance ;)
I don't like that debugger too much though,
since using CoWork and having 40something threads running it likes to crash when I just try to change the inspected thread once halted without notice.
gdborig sometimes detaches from TheIDE occasionally and stays running in the background even after the inspected App is no longer running etc..
So yeah, that thing is NOT to my liking and thus I wouldn't want to bet money on it being accurate :(

Oblivion wrote on Thu, 11 May 2017 20:57
Or it might be that the worker threads outlive the data objects (grid, or SQL).
For the sake of simplicity I'll give you a minimal example.
The following code MAY or MAY NOT lead to crash, for we can't be sure how long the thread will take to finish.

```
MyApp::MyMethod()
{

    String text; // String will be destroyed when when MyMethod returns().

    Thread().Run([=] {

      // Do someting that takes time.
      text = "hello world";         // This may or may not lead to crash. Because the thread may outlive the text object and MyMethod.
    });



}
```

So you'll need to take into consideration all kinds of concurrency problems.

If you can provide a simple example (and a very small dummy database) reproducing the crash, I'll loook into it and try to help.

Regards,

Oblivion.

Yeah.. IDK, the only things thread accesses outside it's own scope of declaration
are the grid itself and The SQL instance, none of which gets destroyed as long as the App is running. *shrugs*

So if it crashes when I attempt to close the App, I'd agree..
but as long as the App remains open ... I don't get it.

I'll see if I can live with the single threaded speed for a while, which is working properly..
and if I can cut the app down to a functional demo with sqlite or something,
to show you at which point it breaks.

I again thank you for your patience and help.

Since you brought that up earlier..
This forum is all BUT full of helpful people
frankly, I've almost given up already..
it looks like most questions here do not get any answer at all
(or at best some general 'read the documentation' crap which is in fact more insult than help)
I've searched this forum a LOT the last few weeks, and basically nothing I was looking for was answered (although all has been asked at least once or twice some questions even four times...)

But this of all threads is not the place to rant, in fact I'm happy I'd give it another try, since you've proved there are exceptions.
And that's really letting me hope again.
And for that as well (on behalf of all newbies to the upp namespace) thank you!

Subject: Re: Select Grid Row BY ID
Posted by Oblivion on Fri, 12 May 2017 06:33:51 GMT
View Forum Message <> Reply to Message

Hello germax

One last thing that comes to my mind is that you are using copy capture in lambda function "[=]". Could you try using capture by reference "[&]"? (since you say that the objects outlive the worker thread, and it seems that your code spawns a single worker thread, it should be safe enough, at

least for testing purpose.)

Regards.

Oblivion.

Subject: Re: Select Grid Row BY ID
Posted by germax on Sat, 13 May 2017 17:13:55 GMT
View Forum Message <> Reply to Message

Hi,

I was just taking some time to throw a few lines together for you
to make a "closer to app" demo that shows you the errors I'm getting.

single thread, one worker thread and multiple co workers
(definition in header.. you'll see ;))

it's crashing flawlessly LOOL

File Attachments

1) MTSQLTest.zip, downloaded 215 times

Subject: Re: Select Grid Row BY ID
Posted by Oblivion on Sun, 14 May 2017 14:18:29 GMT
View Forum Message <> Reply to Message

Hello germax,

I've hopefully "fixed" your code. I only had time for examining the Single- worker threaded variant. It shoudl work now. At least it works on my machines
(A dell laptop with i5-6200i, 4 gb ram, and a destop pc with AMD Fx 6100 six core processor, 16 GB ram.).

I've also made some small comments here and there.

Pleas allow me to point out some issues I've observed while reading your code (It is very clearly written, by the way.):

1) I wouldn't use threads that way. It may work with a single worker thread but you'll definitely run into every kind of concurrency problems if you spawn more than one worker thread. Also, as I've commented in the source code, while(Thread::IsOpen()) ProcessEvents(); is not a good way to handle threads. It goes against almost every purpose thread mechanism is here for. :) What you achieved is a non-blocking application. But this doesn't mean that it is asynchronous. You should

let worker threads work on their own and let them notify you when they're finisihed. That's why I suggested you read GuiMT example. Yet that example is very primitive. If you look for an actual application handling more than one threads, I'd like to suggest you reading the code of FtpBrowser example I wrote for my FTP class (you can find it here: http://www.ultimatepp.org/forums/index.php?t=msg&th=8899 &goto=43053&#msg_43053
While the example is purposefully somewhat crude, it can give you the idea on how to manage threads. Look into FtpAsyncIO() function and into the parts of FtpBrowser code.

2) U++ already has a csv parser. I modified your code. See main.cpp. :)

3) You're using PostCallback somewhat excessively. While it is not wrong at all, you should separate Gui code from (sql) data processing, and call gui code only whenever it is necessary.

Also I'd like to point out that while GridCtrl is very flexible and great, there is a SqlCtrl which is very handy if you're aiming to use SQL with Grids/Arrays.

Regards,

Oblivion

File Attachments

1) MTSQLTest.zip, downloaded 216 times

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?

Subject: Re: Select Grid Row BY ID
Posted by deep on Mon, 15 May 2017 05:44:02 GMT
View Forum Message <> Reply to Message

Hi Germax,

My following suggestions may be issue of programming style.

My option will be to mark "deleted row" in the grid by adding one more column. rather than deleting line from grid.

My first choice will be to use in memory Vector of rows. Rows is Vector of strings. And then process.
Do you really need visual display during processing?

Subject: Re: Select Grid Row BY ID
Posted by germax on Mon, 15 May 2017 12:22:16 GMT
View Forum Message <> Reply to Message

yeah.. not an issue really..
tried both (as you can see in true multithreading I use a vector to store which line gets to be removed from within main thread...
And I've tested that with a single worker as well.

I think the main issue with oblivions code not working properly on my machine is that sql conenctions are not exactly thread safe.
that's why it fails unpredictably at any of the first say twenty rows.
(sometimes on row 2, sometimes at row 12.. but always at an SQL Insert or Update, not on an SQL Select for some unknown reason)

PerThread() is making things worse on my single core for some reason.
(maybe sqlite doesn't like multithreading.. haven't checked *shrugs*)

Anyways, there are many things that do not work well at all;
some are even perplexingly odd in terms of "why you no work you punk"

Take the ProgressInfo for example:
ProgressInfo prog in header (as per oblivions example)

prog++ (as per the ProgressInfo A++ is failing claiming operator ++ not available)
prog.operator++() (that looks sooooo wrong :roll:) is working flawlessly...

Ah well, you win some you loose some ;)

Oh btw.. as soon as there is no sql involved at all.. oblivions code is woking just nicely,
(with line removal) it even works with "forward iteration" removing lines and accessing the same line again (since it's the next unprocessed one)

manipulating the grid either way in a thread is troublesome..
adding a column is rather poinltess imho if that information can be stored elsewhere
for the second it'll be usefull. (gridctrl is quite complex in terms of storage... a vector is of much smaller footprint ;))
passing the row as a raw vector to the function has also been tried..
no improvement at all (but a new container thus more memory in use)
it's always the SQL causing troubles.

What I really "needed" has been resolved by oblivion already (non locking gui) in a single threaded main (no workers whatsoever)
Apart from that, visual representation (coloring etc)
itself is more eye candy than of certain use while the processing is in progress.
but I'd have to mark rows (change background colors) afterwards anyways so why not directly ;)
saves me having another loop to do just that ;)
(and frankly it's the exact same thing as changing a grid fields value, right ;))

Subject: Re: Select Grid Row BY ID
Posted by JeyCi on Thu, 18 Feb 2021 13:45:06 GMT
View Forum Message <> Reply to Message

germax wrote on Sun, 14 May 2017 19:48 Again, thank you,
NOW..
since I just tested your alterations while typing,
I'm still getting memory leaks like crazy *shrugs*
I compiled in MinGW (32bit on the single core and 64bit on the 16core)
You MSVC'd maybe?

I tested in MINGW v9.3 for the interest - no leaks & works OK... just one raw I wouldn't recommend to use in thread without synchronization
outab.cont.GetRow(i).Bg(Yellow());
any synchronization should be done either, I think, - as below same stuff in the code was used in PostCallback... but in for-loop it sometimes behaves strange in your code -- I really don't think this 'yellow' is needed here - therefore am not finding the reason... just getting it away...
I see the date of your post :) but was testing for the interest - & see no problems with Sqlite3 behavior in thread (in the attached example: #define ME_SINGLE) - therefore am not agree with:

germax wrote on Sun, 14 May 2017 19:48 code not working properly on my machine is that sql conenctions are not exactly thread safe... it's always the SQL causing troubles.

- not certainly
p.s.
though concerning input of huger amount than I tested - I can suppose some time is needed for sqlite3 to commit each transaction & if you will commit all transactions at once no lag will be between code's work & db's locked state sometimes while fixing each change to db separately... but it is old well-known feature of sqlite3 db

---

Subject: Re: Select Grid Row BY ID
Posted by JeyCi on Fri, 19 Feb 2021 06:31:52 GMT
View Forum Message <> Reply to Message

BTW, I just created function
void MTT::ChgColor(int i, RGBA bg)
{
 //outab.cont.GetRow(i).Bg(Yellow());
 PostCallback([=] {
    outab.cont.GetRow(i).Bg(bg);
    outab.cont.Refresh();
   });
}
for class MTT : public WithStockLayout<TopWindow>
& calling it in thread like ChgColor(i, LtGreen());
(just to get away duplication of repeated functionality)
But the main feature is Refresh() ;) to see grid_color_changes & any changes at all

---