Subject: Re: httpRequests in secondary non-gui-main Thread
Posted by JeyCi on Tue, 03 Nov 2020 12:40:53 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 03 November 2020 09:08 The best way is (after it stops on memory breakpoint) Debug / Copy backtrace of all threads.
? what do you mean "backtrace" - is it this dropdown
[
mirek wrote on Tue, 03 November 2020 09:08Any chance you are doing something like early exit from the thread?
seems not - I'm simplifying testing case :)
mirek wrote on Tue, 03 November 2020 09:08
Ah, and maybe the most important: Are you using Upp::Thread? Or are how do you start the thread?

member Thread work; - in class MyAppWindow : public WithMyAppWindowLayout<TopWindow>
starting work from here:
```
void MyAppWindow:: btnStart_Click()
{
   work.Run ( [=]{ WorkerThread(); } );
}
```
I thought that native for U++ Thread is OK & I do not need creating my own wrapper_class for executing my thread - I like opportunities U++ gives with its Thread class... or do you think I'd better create my separate wrapper_class to work with Thread?.. something like RAII-style... though I don't need any special ways of synchronization - am just using THISFN for interaction with Gui, using GuiLock/Call in these functions...
===
So, here:
```
    MemoryIgnoreLeaksBegin() ;
    w.http.Do();
    MemoryIgnoreLeaksEnd();
```
- with thus it is OK when being executed Workerthread() function...
or do you think it is better to create a separate class to wrap the Thread?..
===
BTW the situation still differs depending on chosen compiler - as I have written - with MINGW_built_in_U++ - OK working... but with my newer mingw_9_3 in Build Method I do have leak in the place I've shown in the code (switching-on MemoryIgnore in that line - function works good)...
because I deleted everything in BuildMethod from U++ but made settings (path, include, lib) only for foreign mingw (loaded with msys2) - perhaps I have lost something important in my settings (when removed native stuff)?
though in msys2 I have even done separate installation additionally to mingw itself:
Installation:pacman -S openssl
===
oh, mirek - I think I can suppose the reason - msys2 is also no more supporting x32 from May 2020... you was right - most modern CPUs support the possibility to increase RAM-size... really...
Anyway it seems I now do know how to find memory_leak in U++ - no other foreign tools needed -- Thank you...

and the testing code still works without memory-leaks in U++-v.13664 - the version in which mingw already build_in

File Attachments

1) 03.11_2.jpg, downloaded 892 times