Wt - Support #6692

Question: Offline update of WStandardItemModel

10/31/2018 05:31 AM - Mike Ackley

Status:	Feedback	**Start date:**	10/31/2018
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Hi,

I have a WStandardItemModel which I populate with an online 'report', then use a WTableView to display it.

The report then needs to be printed, so I copy the contents of the WStandardItemModel into a WTABLE which I use to render this into a PDF.

I want the printed report to have the same content as the online report, so I created a "offlineWStandardItemModel" instance and updated this using the same function as the online one, but from a secondary thread. The offlineWStandardItemModel is not connected to a view like the online one (which uses SetModel in the TableView).

However, I am getting into a bit of performance/locking issues because WT is treating (I assume correctly) the OfflineStandardItemModel instance as part of the Widget Tree; I wanted to update this offline model in my secondary thread, not 'joined' but just sending posts back to the main window progress bar.

However, because the offlineWStandardItemModel is treated as part of the "widget tree" it seems I cannot update this in the background without using a thread-join, or in my case 'grabbing the update lock'.

In my background thread, I grab the lock, then take about 2mins to make my report; but this causes the GUI to flash the 'waiting' prompt on the browser and ties up the online performance.

If I don't use the OfflineWStandardItemModel,I can update offline data in the background thread just fine.

So-is there a way to make my OfflineWStandardItemModel not part of the main GUI? What makes this part of the "widget tree"?

Is there an alternative, e.g. another APP instance, or another WSERVER instance?

I don't need the GUI to be updated , but want to use the same StandardItemModel from the GUI because it has formatting etc, and then I can use the same function to populate the report like the online one.

The online report is for 'one month' of data but the offline one is repeated for many months, thus the background thread.

I have studied and modelled the current solution on the server-push method, but really it's not quite what I need, since the main artefact to update is the StandardItemModel, and the only GUI interaction I need is the progress bar and a text box showing which month I am up to.

Are there any suggestions of best-practice for this kind of requirement in WT?

Any help would be greatly appreciated.

Regards

Mike

History

#1 - 10/31/2018 10:45 AM - Roel Standaert

I would expect that changing a WStandardItemModel without grabbing the update lock would be possible as long as no view uses it as their model (the model updates the view when it fires its signals, and signals/slots in Wt are not thread safe), and (in Wt 3) if it does not have a WObject that's part of the current WApplication as their parent. You're sure though that all you do is create this WStandardItemModel and then populate it in another thread, completely independently?

#2 - 10/31/2018 10:46 AM - Roel Standaert

- Status changed from New to Feedback

#3 - 10/31/2018 12:04 PM - Mike Ackley

Hi Roel,

Thank you for your reply. There is some additional information that I can share.

A summary of the code is below. The Report4Page is displayed in the Application, with a button that triggers the reportRefresh() method. This calls a thread to do updates. I get segfault in the **printTable** clear if I don't use the appLock. I tried with app->threadJoin; I use WT::Log, and was getting a lot of (apparently) non-threadsafe crashes in WT::Log when the PDF Writer tried to write out "FONT:" messages but I think this is due to my not setting up my own log file and clashing with WT. I was also getting crashes (segFaults) in the WStandardItemModel update as well.

If I make the WStandardItemModel into a new class/subclass, and maybe initiate it from the thread, I don't know if this will put it outside the main loop?

Alternatively, I might need to run the Queries (there are about 60 of them that populate the Wt::WStandardItemModel (it's a summary table, pulling data)... into c container, and manage the online activity separately.

I was taking a short cut by just refreshing/copying the WStandardItemModel using the same update function as I used for the online version that already existed.....

```
class Report4Page : Public Wt::WContainerWidget
{
public:
   Report4Page (..)


{
   ..buttons, WTextFields , ComboBox's..

   Wt::WPushButton *doReport = ...
   doReport->clicked().connect(this, Report4Page::ReportRefresh);.


   WT::WStandardItemModel *onlineTrusteeReportSumaryModel_ptr = new... ;
   WT::WTableView *tableView_ptr = new Wt::WTableView();
   tableView_ptr->setModel(onlineTrusteeReportSumaryModel_ptr);

   WT::WStandardItemModel *batchTrusteeReportSumaryModel_ptr =  new... ;
   [ no setModel for the batch one]

}

void Report4Page::EventFunction (percent, message...)
{

   progress_ptr.set (percent);
   message.setText(message);

}


void Report4Page::ReportRefresh ()
{

  thread=boost::thread (Report4Page::MyThread);


}

void Repor4Page::MyThread ()
{
   refreshMyReport()
}

void refreshMyReport()
{

  [... Grab update lock ]

  if (lock)
  {
```

```
    Wt::WTable *printMyReport_ptr = new Wt::WTable();
    Wt::WContainerWidget *container1_ptr = new Wt::WContainerWidget();
    container1_ptr->addWidget(new WT::WText("Heading"));
    container1_ptr->addWidget(printMyReport_ptr);


    for (i=0; i<mths; i++)
   {
      printMyReport_ptr->clear();
      refreshReport(mth, batchTrusteeReportSumaryModel_ptr);
      copyStandardItemModeltoTable(batchTrusteeReportSumaryModel_ptr, printMyReport_ptr)

      Wt::WServer::Instance()->post (app_ptr->sessionId(), boost::bind ( ..eventFunction, percentComplete.)
      app_ptr->processEvents();    // Update Status
      ...
      makePdf(printMyReportPtr);   // Render container1 ...contains header and table...
   }


  }



}
```

Regards

Mike

#4 - 10/31/2018 01:02 PM - Wim Dumon

Hey Mike,

It's weird to me that you grab the update lock in your thread, since from what you describe that's exactly what you want to avoid - I may have missed the reason for that. Also, calling post() while holding the lock will have no immediate effect since evaluation of the post() requires the lock.

If you need a WApplication object in a thread, you can instantiate a WTestEnvironment and then a WApplication object. You can then attach your thread to this WApplication object (WApplication::attachThread()), if that was the reason why you grabbed the lock.

If your thread need access to classes from your WApplication object, protect those parts of your code with the lock but finer grained (lock, access data, release), so that your application remains responsive while the thread does the work.

Wim.

#5 - 10/31/2018 01:39 PM - Mike Ackley

Hi Wim,

Thanks for your reply. It is weird, as ideally I have no lock, and just post back the events to update a progress bar. I do this completely ok in other widgets..

When I create these objects underneath my Report4Page, even though I do not parent them, does this cause those objects to become part of the main WApplication that is hosting my Report4Page?

I grabbed the lock because I was getting a segfault on the printMyReport_ptr->clear, and when I use the lock, this does not occur.

There is nothing else updating this object, as you can see it was only just created.

While I was not convinced that I had a problem with the event-loop/widget update, according to the documentation, you cannot update "in-loop"(my words) widgets unless you use attachThread or grab the UI lock with serverupdates enabled.

My current conclusion was that either I have a threading/heap/stack-type memory error, or I am illegally attempting to update the WTable from the second thread..

Regards

Mike

#6 - 10/31/2018 03:22 PM - Wim Dumon

Mike Ackley wrote:

> Hi Wim,

> Thanks for your reply. It is weird, as ideally I have no lock, and just post back the events to update a progress bar. I do this completely ok in other widgets..
>
>

Sounds right. Using post() to a session for which you hold the lock (and keep holding it) to report progress makes no sense since the posted function will not be executed until after the lock was released.

> When I create these objects underneath my Report4Page, even though I do not parent them, does this cause those objects to become part of the main WApplication that is hosting my Report4Page?
>
>

No. But it is so that some functionality of Wt assumes it is executed in the context of an application. For example, when a translated WString (using tr()) is converted to text, it will for example look up the current application object to query its message resource bundles. Another example is that some widget will render differently based on the user agent of the browser, for which it needs the WEnvironment attached to the WApplication. This lookup happens through thread local storage (TLS), so when that happens from a thread that was not bound to an application, you get a null pointer access. Grabbing the update lock sets this TLS by calling attachThread; manually calling attachThread is normally dangerous since it usually indicates you're going to access an application object without properly holding a lock.

> I grabbed the lock because I was getting a segfault on the printMyReport_ptr->clear, and when I use the lock, this does not occur.
>
>

A strack trace is needed to conclude what the reason of this segfault is. If it's caused by an access to WApplication::instance(), the use of a WTestEnvironment and a local WApplication can resolve the problem. Otherwise, well then we'll have to see what the problem is :)

Wim.

#7 - 11/01/2018 01:19 AM - Mike Ackley

Hi Wim,

Yes, I think the error was a segfault in WApplication::instance(), but I did not keep a screenshot of the trace.

When I get a moment, I will switch off the lock and it should return pretty quickly, and I will post a stack here.

I was also getting another error, with WT::LOG writes (from this secondary thread) from the PDF renderer, however, I believe this is my own issue, with respect to not naming/initializing my own log file, and also possibly not using WT::LOG in a threadsafe manner; It's understandable that WT writes out some debug messages and I try to also write at the same time from another thread. So this fix was to remove any WT::LOG messages from my second thread.. it's easy for forget sometimes, it's not just 'my code' that's running, WT is doing a lot :-)

I will post again soon...

Regards

Mike