Wt - Bug #3881

Unexpected "Reentrant statement use is not yet implemented"

03/06/2015 02:31 PM - Emeric Poupon

Status:	Closed	**Start date:**	03/06/2015
Priority:	Normal	**Due date:**	
Assignee:	Koen Deforche	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	3.3.4		

Description

I upgraded Wt today, and I get a new issue that I didn't have before

(From git://github.com/kdeforche/wt

7023b38..43ce974 master -> origin/master)

In the handleRequest of the WResource used to get the cover art of tracks, I have this:

{

Wt::Dbo::Transaction transaction(_db.getSession());

Database::Track::pointer track = Database::Track::getById(_db.getSession(), trackId);

covers = CoverArt::Grabber::getFromTrack(track);

}

In the Track class:

typedef Wt::Dbo::ptr pointer;

Track::pointer

Track::getById(Wt::Dbo::Session& session, id_type id)

{

return session.find().where("id = ?").bind(id);

}

The browser seems to fetch several covers at the same time, I end up with this exception:

(gdb) bt

#0 0x00007ffff2ef6dbd in __cxa_throw () from /usr/lib/x86_64-linux-gnu/libstdc.so.6

#1 0x00007ffff704c26d in Wt::Dbo::SqlConnection::getStatement (this=,

id="select \"id\", \"version\", \"track_number\", \"disc_number\", \"name\", \"artist_name\", \"release_name\", \"duration\", \"date\", \"original_date\", \"genre_list\", \"path\", \"last_write\", \"checksum\", \"has_cover\\" from \"tra"...) at /storage/emeric/MesProgs/wt/wt/src/Wt/Dbo/SqlConnection.C:57

#2 0x00007ffff703f3a1 in Wt::Dbo::Session::getOrPrepareStatement (this=0x7fffd80c0d20,

sql="select \"id\", \"version\", \"track_number\", \"disc_number\", \"name\", \"artist_name\", \"release_name\", \"duration\", \"date\", \"original_date\", \"genre_list\", \"path\", \"last_write\", \"checksum\", \"has_cover\\" from \"tra"...) at /storage/emeric/MesProgs/wt/wt/src/Wt/Dbo/Session.C:1118

#3 0x000000000051d150 in Wt::Dbo::Impl::QueryBase<Wt::Dbo::ptrDatabase::Track >::statements (this=this@entry=0x7fffe3ffbf90, where="(id = ?)", groupBy="", orderBy="", limit=--1, offset=--1) at /usr/include/Wt/Dbo/Query_impl.h:137

#4 0x000000000051d40d in Wt::Dbo::Query<Wt::Dbo::ptrDatabase::Track, Wt::Dbo::DynamicBinding>::resultList (this=this@entry=0x7fffe3ffbf90) at /usr/include/Wt/Dbo/Query_impl.h:483

#5 0x0000000000527894 in Wt::Dbo::Query<Wt::Dbo::ptrDatabase::Track, Wt::Dbo::DynamicBinding>::resultValue (this=this@entry=0x7fffe3ffbf90) at /usr/include/Wt/Dbo/Query_impl.h:469

#6 0x0000000000515b28 in operator Wt::Dbo::ptrDatabase::Track (this=) at /usr/include/Wt/Dbo/Query_impl.h:494

#7 Database::Track::getById (session=..., id=3575) at ../../src/database/Track.cpp:165

#8 0x00000000005f92b2 in UserInterface::CoverResource::handleRequest (this=0x7fffc805c0e0, request=..., response=...) at ../../src/ui/resource/CoverResource.cpp:91

#9 0x00007ffff77a4c18 in Wt::WResource::handle (this=this@entry=0x7fffc805c0e0, webRequest=webRequest@entry=0x7fffd00025d0, webResponse=webResponse@entry=0x7fffd00025d0,

continuation=...) at /storage/emeric/MesProgs/wt/wt/src/Wt/WResource.C:216

#10 0x00007ffff79cb41a in Wt::WebSession::notify (this=0x7fffd84d1a20, event=...) at /storage/emeric/MesProgs/wt/wt/src/web/WebSession.C:2212

#11 0x00007ffff79c4f58 in Wt::WebSession::handleRequest (this=0x7fffd84d1a20, handler=...) at /storage/emeric/MesProgs/wt/wt/src/web/WebSession.C:1619

#12 0x00007ffff79b6267 in Wt::WebController::handleRequest (this=0xadf170, request=0x7fffd00025d0) at /storage/emeric/MesProgs/wt/wt/src/web/WebController.C:713

#13 0x00007ffff6b2d6ef in operator() (a1=0x7fffd00025d0, p=0xadf170, this=) at /usr/include/boost/bind/mem_fn_template.hpp:165

#14 operator()<boost::_mfi::mf1<void, Wt::WebController, Wt::WebRequest*>, boost::_bi::list0> (a=, f=, this=)

at /usr/include/boost/bind/bind.hpp:313

#15 operator() (this=) at /usr/include/boost/bind/bind_template.hpp:20

-> ~/MesProgs/wt/wt/src/Wt/Dbo/SqlConnection.C:57-> throw Exception("A collection for '" + id + "' is already in use." \" Reentrant statement use is not yet implemented.\");

What do you think ?

History

#1 - 03/06/2015 03:37 PM - Emeric Poupon

As a workaround, I tried to set a global mutex on this part of code in the WResource.

Now I have a 'Wt: Exception while streaming resourceOperation requires an active transaction' that shows up after about 15 resources fetch.

#2 - 03/09/2015 11:40 PM - Koen Deforche

- Status changed from New to Feedback

- Assignee set to Koen Deforche

The first issue seems to be that your resource cannot handle concurrent requests. Wt will allow concurrent requests to resources. If you can't have that, then you should indeed protect access with a mutex.

How does the following work:

```
{
Wt::Dbo::Transaction transaction(_db.getSession());
Database::Track::pointer track = Database::Track::getById(_db.getSession(), trackId);
covers = CoverArt::Grabber::getFromTrack(track);
}
```

You are probably not holding on to the transaction long enough? Wt::Dbo implements several things using lazy-loading.

#3 - 03/11/2015 12:01 AM - Emeric Poupon

Ok, let's focus on the first issue.

Here is the full source code:

- Resource: https://github.com/epoupon/lms/blob/master/src/ui/resource/CoverResource.cpp

- Database::Track::getById here : https://github.com/epoupon/lms/blob/master/src/database/Track.cpp#L163

- CoverArt::Grabber::getFromTrack : https://github.com/epoupon/lms/blob/master/src/cover/CoverArtGrabber.cpp#L76

Doing this in my resource code seems to solve the problem (dunno yet why it didn't work the other day...)

```
{
   std::unique_lock<std::mutex> lock(_mutex);
   Wt::Dbo::Transaction transaction(_db.getSession());
   Database::Track::pointer track = Database::Track::getById(_db.getSession(), trackId);
   covers = CoverArt::Grabber::getFromTrack(track);
}
```

#4 - 03/11/2015 12:29 PM - Koen Deforche

Hey,

There is still a risk that you are allowing concurrent request to a particular 'covers', which may be dangerous if this is a Wt::Dbo::ptr, or contains Wt::Dbo::ptr's. Since these ptr's lazy load, or may lazy-load associated objects, you still have the risk of concurrency issues.

Koen

#5 - 03/11/2015 08:40 PM - Emeric Poupon

Ok, here is a reduced test case of my custom resource :

```
                {
                        static std::size_t i = 0;
                        LMS_LOG(MOD_UI, SEV_DEBUG) << "Resource request " << i++;

                        Wt::Dbo::Transaction transaction(_db.getSession());

                        Database::Track::pointer track = Database::Track::getById(_db.getSession(), trackId);
                        sleep(1);
                }
```

In the Track class:

```
typedef Wt::Dbo::ptr<Track> pointer;
```

```
Track::pointer
Track::getById(Wt::Dbo::Session& session, id_type id) {
   return session.find<Track>().where("id = ?").bind(id);
}
```

As you can see, I don't use the track's contents.

Output:

```
10.0.1.100 - - [2015-Mar-11 19:09:54.400105] "POST /?wtd=bFkLLzSE3ubYY9X0 HTTP/1.1" 200 1528
[2015-Mar-11 19:09:54.400155] 7202 - [info] "WebRequest: took 9.31ms"
[2015-03-11 19:09:54] [UI] [DEBUG] Resource request 0
[2015-03-11 19:09:54] [UI] [DEBUG] Resource request 1
[2015-03-11 19:09:54] [UI] [DEBUG] Resource request 2
[2015-03-11 19:09:54] [UI] [DEBUG] Resource request 3
[2015-03-11 19:09:54] [UI] [DEBUG] Resource request 4
[2015-Mar-11 19:09:54.426496] 7202 [/ bFkLLzSE3ubYY9X0] [error] "Wt: Exception while streaming resourceA colle
ction for 'select ""id"", ""version"", ""track_number"", ""disc_number"", ""name"", ""artist_name"", ""release
_name"", ""duration"", ""date"", ""original_date"", ""genre_list"", ""path"", ""last_write"", ""checksum"", ""
has_cover"" from ""track""  where (id = ?)' is already in use. Reentrant statement use is not yet implemented.
"
[2015-Mar-11 19:09:54.426626] 7202 [/ bFkLLzSE3ubYY9X0] [error] "Wt: fatal error: A collection for 'select ""i
d"", ""version"", ""track_number"", ""disc_number"", ""name"", ""artist_name"", ""release_name"", ""duration""
, ""date"", ""original_date"", ""genre_list"", ""path"", ""last_write"", ""checksum"", ""has_cover"" from ""tr
ack""  where (id = ?)' is already in use. Reentrant statement use is not yet implemented."
[2015-Mar-11 19:09:54.426714] 7202 - [info] "WebController: Removing session bFkLLzSE3ubYY9X0"
```

I don't understand this behavior since I am not iterating over a Wt::Dbo::collection nor using lazy loading?

#6 - 03/17/2015 04:50 PM - Koen Deforche

But you now removed the lock again?

#7 - 03/17/2015 05:00 PM - Emeric Poupon

Yes indeed: I tried to reproduce a simpler test case in order for you to see what I am doing and why I think there is something wrong.

Just have a transaction that is used to get some unused data from the database and a sleep.

After a very few calls, I get the error.

I do not use collections nor lazy loadings. Everything is done in the transaction scope.

#8 - 03/17/2015 05:16 PM - Koen Deforche

A Wt::Dbo::Session is not thread-safe. So it will not work from a resource like this, without a mutex. The error you get is predictable since two threads will now simultaneously try to use the same connection (and that won't work at all --- that's not something that can be fixed: database connections are single threaded by nature).

Koen

#9 - 03/17/2015 05:40 PM - Emeric Poupon

Ok, since upgrading Wt made the bug appears I missed that point!

Just to be sure: each WApplication (session) will be run by at most one thread?

I mean, the pool of threads is just used to make several sessions run simultaneously?

Then binding a Wt::Dbo::Session to the private data of a Wt::Wapplication is safe?

#10 - 03/17/2015 07:33 PM - Wim Dumon

A WApplication (and its widget tree) is protected by the WApplication::UpdateLock. Before Wt handles a browser event, it will grab the updatelock, so indeed access to a single WApplication is exclusive even if a Wt application typically spawns multiple threads. The threads are indeed mainly used to handle requests for different sessions simultaneously.

WResources are special, in the sense that they are objects that are served by the http server. Many http connections can be querying the same WResource simultaneously. The handleRequest method of a WResource is called by the httpd **without** grabbing the WApplication UpdateLock. It is thus not safe to use data from the WResource from handleRequest, and it is not safe neither to use data from WApplication in handleRequest. State data related to one particular http request is intended to be stored in Http::Request::continuation().

So 'yes', binding a Dbo::Session to WApplication is safe in the sense that if you're allowed to access the WApplication object, you will have exclusive access. But it is not safe to use WApplication from handleRequest (since the update lock will not be held).

Wim.

#11 - 08/26/2015 11:41 AM - Benoit Daccache

- Status changed from Feedback to Closed