Wt - Bug #8305

WRasterImage_gm WResource::setChanged() race can result in segfault

04/05/2021 03:17 PM - Bruce Toll

Status:	Closed	**Start date:**	04/05/2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	4.5.1		

Description

With Wt 4.5.0-rc1-46-g169236d8 (github master), there appears to be a race in WRasterImage-gm.C that can result in a segfault. I encountered the issue testing the bundled styleexample.wt, but was unable to reproduce it reliably. I've attached a test program that is similar in structure, but should hopefully make the issue easy to reproduce.

The race appears to be caused by a call to WResource::setChanged() from WRasterImage::Done(). This ultimately results in a call to WApplication::addExposedResource() with the application lock not held.

I'm not sure why the call to WResource::setChanged() was added. The other WRasterImage implementations do not seem to include this call and it seems like it could result in unnecessary work when a WRasterImage hasn't changed.

I've attached a simple patch for your review that removes the setChanged() call. It has only been lightly tested.

History

#1 - 04/08/2021 11:01 AM - Korneel Dumon

Hi Bruce,

you can make the problem go away by simply using the WRasterImage as the resource in the first place:

```
class ImageResource : public WRasterImage {
public:
  ImageResource(PaintedImage *img_)
    : WRasterImage("png", img_->width(), img_->height())
  {
    WPainter p(this);
    ...
  }
};
```

#2 - 04/08/2021 02:49 PM - Roel Standaert

Right, this is one of those cases where every WResource implicitly belongs to a particular app if there is a WApplication::instance(), which is not always desirable. I don't necessarily like that to be honest, but it would be a breaking change if we made it explicit.

In this case we could say: if we don't have a url, then we don't need to update it.

I believe there was a reason for frequently updating images to refresh their URL, I suppose as a cache-breaking thing? That was sadly not clearly documented when the change was made...

This may just be missing in the Direct2D implementation. I'm not entirely sure if its handleRequest shouldn't also take a lock.

#3 - 04/08/2021 03:12 PM - Bruce Toll

Hi Korneel,

Thanks for following-up and working through the convoluted example. I appreciate your suggested workaround -- which works well.

I believe the suggested change works because painting gets done in the constructor where it is synchronous with respect to the application. As a result, the application lock is held when WRasterImage-gm calls WResource::setChanged().

As a note, I had originally tried to drive the issue with a simpler example using a specialized WPaintedWidget with setPreferredMethod(RenderMethod::PngImage). That also proved to be synchronous with the application, so I was unable to reproduce the issue in that way.

I believe that the approach used in styleexample.wt provides for more parallelism, at the expense of complexity.

I drafted this response before seeing Roel's comment (Thanks Roel!) and may follow-up separately.

#4 - 04/08/2021 03:26 PM - Korneel Dumon

Ah right, I disregarded the fact that you said that's how we do it in the example. Of course you are correct that we should do something about that, fix either the example or allow this usage.

#5 - 04/08/2021 04:12 PM - Bruce Toll

Hi Roel,

Thanks for following-up. Your response reminded me of a feature request that I was working on a while back to provide a more fine-grained approach to setChanged() that goes beyond cache busting. I'll update it for the master branch and file it as a draft for your review. The feature is not directly related to this issue, but I think it has an implicit assumption that setChanged() calls are initiated by an application.

#6 - 04/08/2021 05:59 PM - Bruce Toll

Thanks for the reply, Korneel. I've also added feature request #8321 as potentially relevant to this discussion.

#7 - 04/09/2021 03:05 PM - Bruce Toll

Thinking about this some more, I wonder if WResource::setChanged() might currently be addressing two use cases that could possibly be split:

1. The application becomes aware of a change in state that affects what a WResource would deliver; it calls setChanged().
2. A WResource recognizes that it has completed delivery for a resource request and hence a subsequent access could result in different output; it calls setChanged().

Perhaps, the second case could be handled with a new mechanism, e.g. setDelivered(). That is, there would be a delivered() signal that, if connected, would get emitted with the application lock held when setDelivered() is called by the WResource. An application could connect to the delivered() signal if it needs to update its state based on actual delivery of a resource. This state change might result in a call to WResource::setChanged().

As an example, an application might generate a WImage of an int value painted with drawText(). The application could use the delivered() signal to recognize when the current number had been downloaded/viewed and respond by updating the int variable and calling setChanged() on the associated WResource.

If there are no connections to the delivered() signal, then there should be no need to grab the application lock.

#8 - 04/09/2021 03:21 PM - Roel Standaert

It's not addressing that second use case at the moment. I don't know why you'd think that.

I don't think that second use case needs to be something built in to Wt. Resources that need to do something like that would grab the application lock and emit some more specific signal. It's not very reliable anyway, a resource may still get requested multiple times and you don't really know whether the response was actually received by the browser.

#9 - 04/09/2021 04:05 PM - Bruce Toll

Thanks for following-up, Roel. I think I misinterpreted the intent of the WResource::setChanged() call in WRasterImage in WRasterImage-gm.C. In any case, I agree with you that notification of delivery could be handled in user code and that reliability would be an issue. So, not a good idea....

#10 - 04/09/2021 04:53 PM - Roel Standaert

- Status changed from New to Resolved

- Target version set to 4.6.0

Yeah, WRasterImage is a WResource. The idea is that normally if you want to use it that way you'd create a WRasterImage and paint on it while you have the update lock, and then use its url e.g. for a WImage or WAnchor.

In this case, however, the WRasterImage is being created inside of the handleRequest of some other resource. At that point WApplication::instance() is not null, but we also don't have the update lock. The WRasterImage however just sees that WApplication::instance() is not null and thus calls setChanged() when it's done.

Our fix is to first check whether the WResource is actually already exposed by seeing if it already has a currentUrl_. If not, then we don't need to generate a new URL and expose it.

#11 - 04/09/2021 07:14 PM - Bruce Toll

Thanks for the detailed explanation and quick fix for this issue. Have a great weekend!

#12 - 06/09/2021 07:30 PM - Roel Standaert

- Target version changed from 4.6.0 to 4.5.1

#13 - 08/05/2021 03:40 PM - Roel Standaert

- % Done changed from 0 to 100

#14 - 10/19/2021 11:01 AM - Roel Standaert

- Status changed from Resolved to Closed

Files

test_wrasterimage_gm_race_20210401a.C	2.44 KB	04/05/2021	Bruce Toll
0001-Eliminate-WRasterImage-gm-race.patch	875 Bytes	04/05/2021	Bruce Toll