User Interface - Bug #2766

implement a text metrics server process

10/20/2015 09:54 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stabilization for GUI		
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to User Interface - Feature #1794: implement font support	**Closed**
Related to User Interface - Bug #2745: fill-in sizing differences	**Closed**
Related to User Interface - Bug #2701: default font text sizing issue	**Closed**
Related to User Interface - Feature #3246: reduce the amount of data being se...	**Closed**
Related to Runtime Infrastructure - Feature #5776: reduce memory requirements...	**Test**

History

#1 - 10/20/2015 10:09 AM - Greg Shah

Please read the #2745 notes carefully. This describes a scenario where AUTO-RESIZE allows a widget to change size **after** dynamic text has been assigned to the widget. Previous text metrics solutions are all based on the idea of static sizing. This new finding shows that there can be a real problem at runtime.

We can potentially capture some static strings that are assigned to widgets before assigning the FONT and in #2745 we are going to handle the initial spaces case. But capturing string assignments is not a complete solution. In the truly dynamic case, we need runtime metrics to be served.

Today we capture metrics using tools/get-text-metrics.p. This uses native WIN32 APIs via Progress' PROCEDURE EXTERNAL. We can take this code and move it into a simpler JNI approach (added to p2j.dll), where we can then write Java code to query the metrics on demand. This would be designed as a small/lightweight server that can be connected to a P2J server in a similar way to an appserver agent. Of course, it can only be run on a Windows OS "client" system. Then dynamic text metric requests in P2J would be routed to this server and the responses provided to and cached in P2J.

Of course, not all customers will have a Windows box to use for this, or they may decide not to run it (it adds complexity). But for customers that do want to solve the problem, this will give us a full solution.

Values that are already known (including our current statically captured metrics) should be used first. Then we would fall back to this code. Finally, if that fails we fall back to calculating our own metrics in the driver as we do today.

In the case where the P2J server is running on Windows, we should run this process locally and with very little configuration needed. But this should be able to be installed on any available Windows box, that may be physically separate from the P2J server and clients.

#2 - 10/31/2015 07:31 AM - Greg Shah

I expect that Golden Code will host a metrics server on our premises. We would want to make it available to all running P2J instances so that there is always a fallback solution.

Permanently caching the queried results (across P2J server restarts) will be important to reduce the number of times that we have to query over the Internet to get metrics. We would optimally also cache these in a way that would be easy for customers to periodically merge the runtime-generated cache with the captured text-metrics.xml when their application jar as rebuilt.

#3 - 03/23/2016 05:08 PM - Greg Shah

- Target version changed from Milestone 12 to Milestone 16

- Assignee set to Constantin Asofiei

#4 - 11/16/2016 12:23 PM - Greg Shah

- Target version changed from Milestone 16 to Cleanup and Stabilization for GUI

#5 - 12/04/2017 06:05 AM - Greg Shah

In addition to the functional issues we see from deviations from the Windows metrics when encountering dynamic text, we are seeing performance issues related to too many trips down to the JS client for metrics calculation. The following details were provided by Constantin:

> I've been looking into the web clients and I think the trips to the browser to measure the text (the '8f' messages) is hurting us a lot - it took forever just to open a large application's main menu. I could try to expand the set of strings we are pre-measuring in text-metricsl.xml (by i.e. navigating through the application screens and extracting these strings from the server.log), but I think we should look for a way to measure the text on the Java side.
>
> Why do I say this: if you try to re-open the a large screen a second time (after you've walked through all the tabs for this window), you will see that it behaves faster. And the JS drawing cache is not involved, as this is kept on a per-window basis, and when you open it a second time, it's another window ID (as the cache doesn't survive once the window gets destroyed).

And he also posted this:

> I've refreshed the text-metrics.xml and now it's a little better, but the idea remains, to avoid as many JS-side trips as possible (and text metrics trips are many).

#6 - 12/04/2017 06:14 AM - Greg Shah

I agree we should resolve this.

If we can calculate suitable metrics in the Java client, I would be fine with that approach. We already have a full Swing implementation for font management and the results are roughly equivalent to the results in the browser (FF/Chrome on Linux). The closer the equivalence, the better this

solution will be.

Another approach might be to use caching for the calculated metrics. This would have to be done in a persistent way that survives client/server restarts. We would want to aggregate the metrics for all clients. If this was done in a clean way, we could even distribute this cache as part of the application such that the testing of the application would "warm up" the system and ensure that real users will never see the worst of the problem. The problem with this is that these metrics would have to be provided from the server down to each client in some way, which will entail some level of performance cost. If the total amount was not large for real applications, then it might not be too bad.

Still, I think the idea to calculate in Java is the best approach. Can you work on this now?

#7 - 12/04/2017 06:26 AM - Constantin Asofiei

Greg Shah wrote:

> Another approach might be to use caching for the calculated metrics. This would have to be done in a persistent way that survives client/server restarts. We would want to aggregate the metrics for all clients. If this was done in a clean way, we could even distribute this cache as part of the application such that the testing of the application would "warm up" the system and ensure that real users will never see the worst of the problem. The problem with this is that these metrics would have to be provided from the server down to each client in some way, which will entail some level of performance cost. If the total amount was not large for real applications, then it might not be too bad.

The idea here would be to have a H2 database which, with an approach like this:

1. first check local client cache - if it exists, use it
2. second, check server-side cache - if it exists, use it
3. third, measure and updated client and server cache

We could build a strategy like keeping the program(s) which used that text, so we can send them all to the client in one batch, to minimize the trips between client and server.

> Still, I think the idea to calculate in Java is the best approach. Can you work on this now?

Yes, I'll work on it; if this work, the above can be postponed/avoided.

#8 - 12/04/2017 08:02 AM - Greg Shah

Agreed.

#9 - 12/04/2017 10:52 AM - Constantin Asofiei

Some quick and dirty changes in web code allow AWT Font and FontMetrics to be used. But I think I'm not using the correct font, as the metrics are a little off.

#10 - 12/04/2017 01:18 PM - Constantin Asofiei

I was not using the properly derived font (as Swing was doing) when computing the metrics. With current changes, the Web UI looks the same as the old way, which was measuring the text via the JS side.

I need to clean it up and will commit it to 3394a.

Another note: currently FontManager reads the entire font-table (~46k strings) into the client-side's memory. In the long term, we should avoid this. Although, with current replacement fonts, if the metrics are (almost) the same, we might be able to avoid the text-metrics.xml list completely.

#11 - 12/04/2017 01:21 PM - Greg Shah

> Another note: currently FontManager reads the entire font-table (~46k strings) into the client-side's memory. In the long term, we should avoid this. Although, with current replacement fonts, if the metrics are (almost) the same, we might be able to avoid the text-metrics.xml list completely.

Agreed.

You recently added the captured text-metrics.xml to the project. Was there a problem that drove that or did you just do it to avoid possible future issues?

#12 - 12/04/2017 01:27 PM - Constantin Asofiei

Greg Shah wrote:

> You recently added the captured text-metrics.xml to the project. Was there a problem that drove that or did you just do it to avoid possible future issues?

Thanks for asking again, because it reminded me of the old 'we need exact metrics to be able to compute the default layout and widget size' issue. If the metrics differ even by 1px, the default layout/size might not be OK for some widgets. So, I'm not yet completely sure we can remove this dependency.

I'll add a comment somewhere in the code to not forget this, if is not already there.

#13 - 12/04/2017 02:36 PM - Constantin Asofiei

Greg, please review 3394a rev 11232/11233.

#14 - 12/04/2017 03:57 PM - Greg Shah

Code Review Task Branch 3394a Revisions 11232/11233

I'm good with the changes.

#15 - 12/28/2017 11:13 AM - Greg Shah

The improvements for web client font metrics processing are included in task branch 3394a which has been merged to trunk as revision 11214.

The core of the task still remains open.

#16 - 01/03/2018 02:05 PM - Greg Shah

- Related to Feature #3246: reduce the amount of data being sent to the client-side when an UI attribute is being changed added

#17 - 05/24/2022 06:43 AM - Greg Shah

- Related to Feature #5776: reduce memory requirements for the FWD client added