User Interface - Bug #2968

disconnect/reconnect and manual reload must re-establish the websocket connection, needs to update the websock references in all GuiWebEmulatedWindows

01/25/2016 05:16 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Sergey Ivanovskiy	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Feature #1811: implement the AJAX client driver	**Closed**	**11/01/2013**	**03/06/2014**

History

#1 - 01/25/2016 05:16 PM - Greg Shah

Put changes in 1811t.

#2 - 02/12/2016 03:39 PM - Sergey Ivanovskiy

1) Disconnect/reconnect usecase
Greg, please could you explain disconnect/reconnect? Does it mean the main connection is failed, but then it is restored? For an example, a client network interface is down, and then it is up again.
Should we notify a user about the disconnected state? The JS client can't be off-line. Thus we need to prevent any user's actions if the web socket connection is failed. We can start a background task that checks periodically the websocket connection.

2) The manual reload usecase means that a user clicks on the reload button or tries to refresh the current page pressing F5 or CTRL + F5 or another reload shortcuts.
In this case we can reconnect to the java peer, but we should restore the JS client state with all objects and mouse listeners or replay somehow the java objects again (is it possible?) Please correct that we don't need to update the GuiWebSocket references because this object continues its existing between new web socket connections. The JS client is only required to restore its previous state.

#3 - 02/19/2016 12:54 PM - Greg Shah

> 1) Disconnect/reconnect usecase
> Greg, please could you explain disconnect/reconnect? Does it mean the main connection is failed, but then it is restored? For an example, a client network interface is down, and then it is up again.

Yes, it could mean that network connectivity was unavailable for a short period of time. It could be a browser that was closed and then reopened.

Reload is the other case. A reload is basically a disconnect/reconnect that occurs over a very short time interval. This is the most likely case that we need to support.

The situation was originally described in #1811-310 where Marius explained that he implemented a watchdog timer thread in each web client (Java

side) to kill the Java client if the JS side disconnects and doesn't reconnect within a specific timer period.

I want to make sure that:

1. Our recent changes have not broken that support for web ChUI.
2. Our web GUI client has the same level of support.

We do NOT need to support a reconnection after an extended period of network outage. I'm only caring about the pretty quick disconnect/reconnect that is mostly represented by the browser reload (via reload button or F5).

> Should we notify a user about the disconnected state? The JS client can't be off-line. Thus we need to prevent any user's actions if the web socket connection is failed. We can start a background task that checks periodically the websocket connection.

We can consider this case as part of #2969.

> but we should restore the JS client state with all objects and mouse listeners or replay somehow the java objects again (is it possible?)

The objective would be to have the screen look exactly like it did before the reload occurred. I can accept differences if there was some mouse/key events that were not fully transferred to the Java side before the reload. But otherwise we do want the full Java-side state to be rendered on the JS side.

> Please correct that we don't need to update the GuiWebSocket references because this object continues its existing between new web socket connections. The JS client is only required to restore its previous state.

You're sure that the reconnect will occur to the same GuiWebSocket instance as previously used? If so, that is good (and less work).

#4 - 02/22/2016 11:57 AM - Sergey Ivanovskiy

The watch dog timeout is given by this directory configuration path "webClient/watchdogTimeout@value" with the default value that equals -1, it works because the watch dog thread sleeps for one minute at its start, and then it checks if the given watchdog time is elapsed. Thus one minute is the default time interval within which the JS client can be reconnected to the Java web client.
The maximal web socket idle time "webClient/webSocketTimeout@value" if it is provided by the server directory can override the maximal idle time configured by the Jetty websocket parameter given by WebSocket annotation. Now these annotation parameters are not used and this logic is moved

to WebSocketConfig. WebSocketConfig is applied to the web socket policy at the web server starting time, but the maximal idle time "webClient/webSocketTimeout@value" (if it is not -1) is applied to the web socket policy at the websocket connect time and can override these settings.
Planning:
1) to add MSG_PING_PONG
2) to implement the Idle Timer on the JS client that is responsible to send a ping message to the Java side and the Java side must respond with the same received message. The Idle Timer sends ping message only if no messages occur within the specified period that must be less than the maximal web socket idle time. The Idle Timer is canceled if a web socket error or a close event occurs.
3) to implement the Reconnect Timer that is responsible to open new web socket connection to the Java side within the specified reconnect interval that must be less than the watch dog timeout. The timer must be started if a web socket error or a close event occurs and it must be canceled if a web socket open event occurs.
4) to save/serialize the JS client state on reload/close the application web page in order to be able to restore the target JS objects and mouse listeners if the page is reload or opened.

#5 - 02/24/2016 05:55 PM - Sergey Ivanovskiy

- File 2968_1.txt added

Committed revision 11033, 11034 implemented 1,2,3 except of the usecase "to reload/to close the application web page".

#6 - 02/25/2016 06:06 PM - Sergey Ivanovskiy

The following problem appears if the application webpage is reloaded or refreshed on F5. We have two WebPageHandlers. The first one is handled requests to the root "/" as a target, and the second has "/index.html" as a target. The security token is provided for these handlers by the spawner. The first GET request for "/" with the security token (/?token=b3537bbdc481457d0755643c666fe2ce) is handled by the first handler. This handler generates and sets the authorization token in the cookie and return the generated template "index.html". The browser address field is cleared from /?token=b3537bbdc481457d0755643c666fe2ce to "/index.html" by the loaded p2.js. On the reload the GET request for "/index.html" can be handled by the second handler, but it doesn't happen due to this handler expects to get a security token but the cookie is already set by the first handler. Thus the second handler has an uninitialized state that is inconsistent with the current application has already an authorized user. I think it is enough to attach only one handler of "/index.html" and changing the redirect url to this /index.html?token=b3537bbdc481457d0755643c666fe2ce should help. I tried to use the application cache but didn't succeeded in it. It could be that the added manifest file index.appcache has an unknown content type. But we can download the required resources from the web server again because the application can't work to be off-line. Continue working on to restore the application state from the local storage. Planning to try IndexedDB API or save its state on the web side, because Safari doesn't support IndexedDB API.

#7 - 02/26/2016 05:52 PM - Sergey Ivanovskiy

Planing to finish this task this weekend due to the tests to restore the document tree with the embedded fonts and the fonts objects are passed. Decided to use the Web Storage API to save the application persistent state. There is the 5Mb quote for Firefox, and it forces to save only the required numeric data to request the web server to resize all existing windows.

#8 - 02/28/2016 02:36 PM - Sergey Ivanovskiy

I encountered the problems to restore the web application if the application web page is reloaded or reopened.
There are 3 ways to save the JS client modules states
1) to use localStorage object (Web Storage API)
Tried this way to save modules (p2j, p2j.fonts, p2j.screen) states on the JS client side using the Web Storage API.

There are several problems due to the usage of the drawing caches. We can't save all data due to the 5Mb is only available for the storage space. I tried to clean these drawing cashes, but didn't succeeded only parts of the application window was displayed for unknown reasons.
The embedded fonts and p2j.fonts module state can be restored after the page reloading, but the 5Mb quote can be the problem here, because the persisted document tree can require more free space.

2) to use IndexedDB API

If IndexedDB API is used, then there is only one possibility to save the JS client modules states if we schedule these operations due to IndexedDB API provides only asynchronous access to the supported storage. These operations can impact the web performance because if the write transactions are failed, they must be rescheduled and at the moment of the web page reload there are no guaranties that all modules states are saved and the saved states are consistent.

3) to save the JS client modules states on the web java side

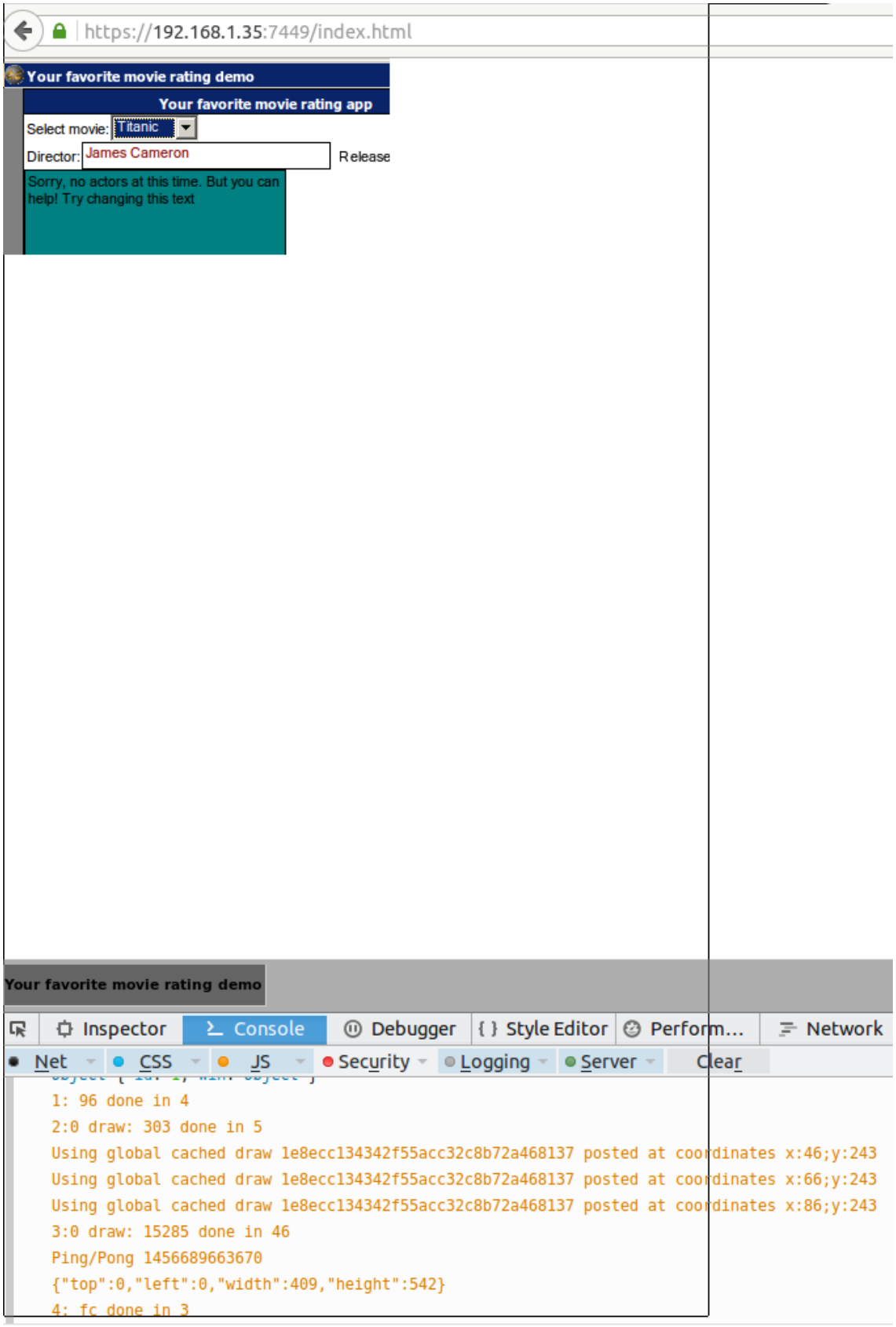
Icons can be sent to the JS client if icons caches are cleaned, but the drawing data are already stored in the caches on the web java side.
I didn't try it but the same problems will appear as in **the 1) way**.

Greg, is this use case required to be implemented? The "stay on page or leave page" dialog can be displayed if beforeunload event is properly handled. It can help a user to stay on the application page if a user accidentally presses some shortcuts to reload or to close the application page. I propose to add the default "stay on page or leave page" dialog and to postpone this use case for a suitable time. Do you agree? Should I continue **this 3) way**?
Thus if a user tries to close or to reload the web application page, then The "stay on page or leave page" dialog appears. If a user selects "leave page", the application page will be closed or will be redirected to the login page. Is it enough for this use case?

#9 - 02/28/2016 03:06 PM - Sergey Ivanovskiy

- File page_reload_close_use_case.png added

The picture result is probably due to the current clipping region on the Web java side. It adds a complexity to the target use case


https://192.168.1.35:7449/index.html
Your favorite movie rating demo
Your favorite movie rating app
Select movie: Titanic
Director: James Cameron
Release
Sorry, no actors at this time. But you can help! Try changing this text
Your favorite movie rating demo
Inspector
Console
Debugger
Style Editor
Perform...
Network
Net
CSS
JS
Security
Logging
Server
Clear
1: 96 done in 4
2:0 draw: 303 done in 5
Using global cached draw 1e8ecc134342f55acc32c8b72a468137 posted at coordinates x:46;y:243
Using global cached draw 1e8ecc134342f55acc32c8b72a468137 posted at coordinates x:66;y:243
Using global cached draw 1e8ecc134342f55acc32c8b72a468137 posted at coordinates x:86;y:243
3:0 draw: 15285 done in 46
Ping/Pong 1456689663670
{"top":0,"left":0,"width":409,"height":542}
4: fc done in 3

#10 - 02/29/2016 08:55 AM - Greg Shah

I don't want to implement any kind of caching, local storage or index db. Can we just make the reload work by forcing a full draw cycle on all windows?

I the issue in note 6 present with the web ChUI too? We had the web ChUI reload case working properly at one point.

#11 - 02/29/2016 09:01 AM - Greg Shah

> Due to this issue I would like to create a new perspective task to do the web application according to a standard web application specification (SRV.9.1 Web Applications Within Web Servers from the servlet specification) with the deployment descriptor web.xml and the directory structure having abilities to reconfigure.

You mentioned this in your status report, related to note 6. I don't understand why this is something useful, but I will say that I do not want to shift to a deployment descriptor approach. I prefer our current approach of managing our configuration through our hand-crafted code.

> Thus the second handler has an uninitialized state that is inconsistent with the current application has already an authorized user. I think it is enough to attach only one handler of "/index.html" and changing the redirect url to this /index.html?token=b3537bbdc481457d0755643c666fe2ce should help.

That is fine. We have no functional reason for multiple handlers.

> I tried to use the application cache but didn't succeeded in it. It could be that the added manifest file index.appcache has an unknown content type. But we can download the required resources from the web server again because the application can't work to be off-line. Continue working on to restore the application state from the local storage. Planning to try IndexedDB API or save its state on the web side, because Safari doesn't support IndexedDB API.

Please don't pursue this further. This adds a level of complexity that is unwanted.

#12 - 02/29/2016 09:09 AM - Sergey Ivanovskiy

The web ChUI has the same problem. I don't know how to repeat all window drawings. How can we restore the embedded fonts? I only know how to save the current HTML DOM tree and then use it to restore the embedded fonts and the p2j.fonts module state.

#13 - 02/29/2016 09:30 AM - Sergey Ivanovskiy

Refresh/reopen requires to redraw all windows, to restore embedded fonts and mouse widgets regions. I tried to restore mouse widgets regions by invoking GuiWebDriver.registerMouseWidgets() without success. The web java state depends on the JS client state. The widgets regions are cached, the drawings are cached, the icons are cached.

#14 - 02/29/2016 09:31 AM - Sergey Ivanovskiy

Fonts are cached separately.

#15 - 02/29/2016 10:18 AM - Sergey Ivanovskiy

> I don't want to implement any kind of caching, local storage or index db. Can we just make the reload work by forcing a full draw cycle on all windows?

Let me explain. We need to send all commands to create and to draw windows, but it is controlled by the server logic. We can't repaint or redraw the all existing windows. If we follow this approach, then we need to save the current JS client state. And the worse thing is I don't know how to repeat all window drawings and to repeat fonts creations because it is done during initialization

```
ThinClient: public static void initializePost(BootstrapConfig cfg, Session session, Object context)
      // initialize the font manager
      FontManager.init(tc.server);
```

I can only restore the JS client state, but on this way I need the large storage space.

> I the issue in note 6 present with the web ChUI too? We had the web ChUI reload case working properly at one point.

The web ChUI has the same problem.

#16 - 02/29/2016 10:37 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Due to this issue I would like to create a new perspective task to do the web application according to a standard web application specification (SRV.9.1 Web Applications Within Web Servers from the servlet specification) with the deployment descriptor web.xml and the directory structure having abilities to reconfigure.

You mentioned this in your status report, related to note 6. I don't understand why this is something useful, but I will say that I do not want to shift to a deployment descriptor approach. I prefer our current approach of managing our configuration through our hand-crafted code.

Ok, agree.

> Thus the second handler has an uninitialized state that is inconsistent with the current application has already an authorized user. I think it is enough to attach only one handler of "/index.html" and changing the redirect url to this /index.html?token=b3537bbdc481457d0755643c666fe2ce should help.

That is fine. We have no functional reason for multiple handlers.

I checked that the proposed solution fixed it.

> I tried to use the application cache but didn't succeeded in it. It could be that the added manifest file index.appcache has an unknown content type. But we can download the required resources from the web server again because the application can't work to be off-line. Continue working on to restore the application state from the local storage. Planning to try IndexedDB API or save its state on the web side, because Safari doesn't support IndexedDB API.

Please don't pursue this further. This adds a level of complexity that is unwanted.

Ok, According to https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API#Browser_compatibility Safari has a basic support. Are there ways to restore the JS client
without restarting the java side client or without saving the current JS client state?

#17 - 02/29/2016 10:53 AM - Sergey Ivanovskiy

I don't like this approach but it is possible to save the current JS client state in the file on the java side and on reload to restore the JS client state. The other way is to recreate the client session, but in that case the current state will be lost. I think to save the current JS client state is only what we can do.

#18 - 02/29/2016 11:02 AM - Constantin Asofiei

Greg Shah wrote:

> I the issue in note 6 present with the web ChUI too? We had the web ChUI reload case working properly at one point.

I think this might have stopped working when we've added the security token during authentication... previously, IIRC, the same ChUI session could be displayed/used on more than one tab.

> Can we just make the reload work by forcing a full draw cycle on all windows?

We need to consider that all the JS-side data is lost, if the web page is reloaded. If we want to "reload" the client, then we need a driver-level API which will push all windows (with their current size/location/minimized/etc state) to the JS side first, then force a redraw for all the windows.

#19 - 02/29/2016 11:04 AM - Constantin Asofiei

Sergey Ivanovskiy wrote:

> > Thus the second handler has an uninitialized state that is inconsistent with the current application has already an authorized user. I think it is enough to attach only one handler of "/index.html" and changing the redirect url to this /index.html?token=b3537bbdc481457d0755643c666fe2ce should help.
>
> That is fine. We have no functional reason for multiple handlers.
>
> I checked that the proposed solution fixed it.

Does this mean that the security token will appear in the browser's address bar?

Greg: if this is the case, then this might be a security issue, considering the URL is exposed via the network - the client's session will be able to be duplicated on any other browser... by anyone who can "see" this URL.

#20 - 02/29/2016 11:08 AM - Sergey Ivanovskiy

Constantin Asofiei wrote:

> Sergey Ivanovskiy wrote:
>
> > Thus the second handler has an uninitialized state that is inconsistent with the current application has already an authorized user. I think it is enough to attach only one handler of "/index.html" and changing the redirect url to this /index.html?token=b3537bbdc481457d0755643c666fe2ce should help.
>
> That is fine. We have no functional reason for multiple handlers.
>
> I checked that the proposed solution fixed it.

Does this mean that the security token will appear in the browser's address bar?

Greg: if this is the case, then this might be a security issue, considering the URL is exposed via the network - the client's session will be able to be duplicated on any other browser... by anyone who can "see" this URL.

Constantin, you are not right, because this token is used only once to generate the authorization token that is saved in the client cookie. No the refreshed page will be "/index.html" after this proposed fix.

#21 - 02/29/2016 11:10 AM - Constantin Asofiei

Sergey Ivanovskiy wrote:

> Constantin, you are not right, because this token is used only once to generate the authorization token that is saved in the client cookie. No the refreshed page will be "/index.html" after this proposed fix.

OK, I understand.

#22 - 02/29/2016 11:16 AM - Sergey Ivanovskiy

Constantin Asofiei wrote:

> Greg Shah wrote:
>
> > I the issue in note 6 present with the web ChUI too? We had the web ChUI reload case working properly at one point.
>
> I think this might have stopped working when we've added the security token during authentication... previously, IIRC, the same ChUI session could be displayed/used on more than one tab.
>
> Can we just make the reload work by forcing a full draw cycle on all windows?
>
> We need to consider that all the JS-side data is lost, if the web page is reloaded. If we want to "reload" the client, then we need a driver-level API which will push all windows (with their current size/location/minimized/etc state) to the JS side first, then force a redraw for all the windows.

It may be more complex task in compare with saving the JS client state. Because I am almost succeeded in saving the JS client state. I used the localStorage and did't save the icons caches and mouse regions and drawings. If we will save this binary state on the java side, we can restore the JS client state and to continue working with the current JS client.

#23 - 02/29/2016 11:17 AM - Constantin Asofiei

Sergey Ivanovskiy wrote:

> It may be more complex task in compare with saving the JS client state. Because I am almost succeeded in saving the JS client state. I used the localStorage and did't save the icons caches and mouse regions and drawings. If we will save this binary state on the java side, we can restore the JS client state and to continue working with the current JS client.

Quick question: what is the additional overhead for saving this state? Is it done on each API call from the client side to the JS side?

#24 - 02/29/2016 11:21 AM - Sergey Ivanovskiy

No, I used to save this state on reload the web page. But I understand it is complex because many fields and variables should be saved.

#25 - 02/29/2016 11:24 AM - Sergey Ivanovskiy

Constantin Asofiei wrote:

> We need to consider that all the JS-side data is lost, if the web page is reloaded. If we want to "reload" the client, then we need a driver-level API which will push all windows (with their current size/location/minimized/etc state) to the JS side first, then force a redraw for all the windows.

It is a systematic approach in compare with saving the JS client state.

#26 - 02/29/2016 11:25 AM - Constantin Asofiei

Sergey Ivanovskiy wrote:

> I used the localStorage and did't save the icons caches and mouse regions and drawings.

Are you planning to save these, too?

#27 - 02/29/2016 11:29 AM - Sergey Ivanovskiy

Yes, mouse regions, icons, drawings, all required fields for each JS module if we will save the JS client state.

#28 - 02/29/2016 11:32 AM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> Constantin Asofiei wrote:
>
> > We need to consider that all the JS-side data is lost, if the web page is reloaded. If we want to "reload" the client, then we need a driver-level API which will push all windows (with their current size/location/minimized/etc state) to the JS side first, then force a redraw for all the windows.
>
> It is a systematic approach in compare with saving the JS client state.

But I think that your approach is more systematic, and will require new API related to the current application state and to the current window state.

#29 - 02/29/2016 11:40 AM - Constantin Asofiei

Other questions:

1. What happens if the user hits "refresh page" while the JS side is in the middle of processing some drawing (or some other JS code)?
2. What happens if the user hits "refresh page" while the server-side has control (i.e. some business logic is being executed on the server), and while the JS data is being saved (triggered by the "refresh page"), the server-side calls the client to i.e. enable a certain field?

What I'm trying to understand is this:

1. how is your approach safe in terms that, once the page is reloaded and the data saved in the JS localStorage is used to restore the client, what will be drawn on the browser will be the actual application state?
2. what happens if, while the refresh is in progress (client state restore is not yet in progress), the P2J client wants to access the JS side (to i.e. enable a field in a frame)? Will the call fail? Will the WebSocket still be open? How will the P2J client treat these calls, while the JS side is busy saving/restoring the page?

#30 - 02/29/2016 11:53 AM - Greg Shah

Do NOT implement JS caching. It adds too many potential problems and too much complexity.

At this point, if it is not reasonable to reinitialize from the Java side, then we will accept this as a limitation.

> The "stay on page or leave page" dialog can be displayed if beforeunload event is properly handled. It can help a user to stay on the application page if a user accidentally presses some shortcuts to reload or to close the application page. I propose to add the default "stay on page or leave page" dialog and to postpone this use case for a suitable time.

I can accept this for now, though it is not a good solution. But I don't want a generic dialog. It needs to be a much more serious warning that the application session is going to be closed.

#31 - 02/29/2016 12:08 PM - Sergey Ivanovskiy

Constantin Asofiei wrote:

> Other questions:
>
> 1. What happens if the user hits "refresh page" while the JS side is in the middle of processing some drawing (or some other JS code)?
> 2. What happens if the user hits "refresh page" while the server-side has control (i.e. some business logic is being executed on the server), and while the JS data is being saved (triggered by the "refresh page"), the server-side calls the client to i.e. enable a certain field?
>
> What I'm trying to understand is this:
>
> 1. how is your approach safe in terms that, once the page is reloaded and the data saved in the JS localStorage is used to restore the client, what will be drawn on the browser will be the actual application state?
> 2. what happens if, while the refresh is in progress (client state restore is not yet in progress), the P2J client wants to access the JS side (to i.e. enable a field in a frame)? Will the call fail? Will the WebSocket still be open? How will the P2J client treat these calls, while the JS side is busy saving/restoring the page?

No, I don't consider these cases thoroughly, but for the java side the refresh is like any other commands as the mouse movements or any others. I don't follow this approach now.

#32 - 02/29/2016 12:09 PM - Sergey Ivanovskiy

Greg Shah wrote:

> Do NOT implement JS caching. It adds too many potential problems and too much complexity.
>
> At this point, if it is not reasonable to reinitialize from the Java side, then we will accept this as a limitation.
>
> > The "stay on page or leave page" dialog can be displayed if beforeunload event is properly handled. It can help a user to stay on the application page if a user accidentally presses some shortcuts to reload or to close the application page. I propose to add the default "stay on page or leave page" dialog and to postpone this use case for a suitable time.
>
> I can accept this for now, though it is not a good solution. But I don't want a generic dialog. It needs to be a much more serious warning that the application session is going to be closed.

Ok, planning to do it.

#33 - 03/01/2016 12:56 AM - Sergey Ivanovskiy

- File 2968_3.txt added

Sergey Ivanovskiy wrote:

> Greg Shah wrote:
>
> > Do NOT implement JS caching. It adds too many potential problems and too much complexity.
> >
> > At this point, if it is not reasonable to reinitialize from the Java side, then we will accept this as a limitation.
> >
> > > The "stay on page or leave page" dialog can be displayed if beforeunload event is properly handled. It can help a user to stay on the application page if a user accidentally presses some shortcuts to reload or to close the application page. I propose to add the default "stay on page or leave page" dialog and to postpone this use case for a suitable time.
> >
> > I can accept this for now, though it is not a good solution. But I don't want a generic dialog. It needs to be a much more serious warning that the application session is going to be closed.

> Ok, planning to do it.

Greg, please review the committed revision 11038. It displays the warning dialog on close or reload the web page as a temporary solution until new Application State API is ready. The warning dialog for Firefox is standard and doesn't permit a custom message, but Chrome and IE can display the custom message. In our case it is "Do you confirm to logout the P2J application?"

#34 - 03/01/2016 10:02 AM - Greg Shah

Code Review Task Branch 1811t Revision 11038

1. With your fix to resolve the 2 hander issue, reload should work for ChUI web client (like it used to work). But your changes will disable this. I only want to block reload for the GUI web client, not for ChUI.

2. The text "Do you confirm to logout the P2J application?" should be changed to "ANY PENDING CHANGES WILL BE LOST! Please confirm that it is OK to exit the application.".

3. I'm a little nervous about using the System.exit(0) in WebClientProtocol because it seems quite heavy handed. We should probably at least write something to the log that we are deliberately exiting due to a MSG_QUIT.

Constantin: what do you think?

4. Please add history entries and update the copyright date range for ChuiWebDriver and p2j.js.

#35 - 03/01/2016 10:15 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Code Review Task Branch 1811t Revision 11038
>
> 1. With your fix to resolve the 2 hander issue, reload should work for ChUI web client (like it used to work). But your changes will disable this. I only want to block reload for the GUI web client, not for ChUI.

Thank you, planning to fix it using p2j.isGui in the JS client.

> 3. I'm a little nervous about using the System.exit(0) in WebClientProtocol because it seems quite heavy handed. We should probably at least write something to the log that we are deliberately exiting due to a MSG_QUIT.
> Constantin: what do you think?

I used System.exit(0) because don't know what keys strokes can be send to quit the current client session. The other way is to add a new method like

GuiDriver.shutdown() to ClientProtocolHooks. Could you help?

#36 - 03/01/2016 03:31 PM - Sergey Ivanovskiy

- File 2968_4.txt added

> Code Review Task Branch 1811t Revision 11038

Committed revision 11040, 11041 fixes 1,2,4.
3 Watchdog timer does the same, System.exit() is used to kill the spawn process in order to be able to create a new one on demand

#37 - 03/01/2016 04:11 PM - Greg Shah

> 3 Watchdog timer does the same, System.exit() is used to kill the spawn process in order to be able to create a new one on demand

OK.

Please do put in logging in both places.

#38 - 03/08/2016 11:00 AM - Greg Shah

Is there anything more to do on this task, other than adding comments?

Please document the limitations of our reload behavior in #2675.

#39 - 03/22/2016 12:00 PM - Greg Shah

Can I close this task?

#40 - 03/22/2016 12:01 PM - Sergey Ivanovskiy

Yes, it is done.

#41 - 03/22/2016 12:02 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

#42 - 11/16/2016 12:12 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

Files

2968_1.txt	62.5 KB	02/24/2016	Sergey Ivanovskiy
page_reload_close_use_case.png	59.1 KB	02/28/2016	Sergey Ivanovskiy
2968_3.txt	12.7 KB	03/01/2016	Sergey Ivanovskiy
2968_4.txt	7.42 KB	03/01/2016	Sergey Ivanovskiy