User Interface - Bug #2956

Bug # 2677 (New): fix drawing and functional differences between P2J GUI and 4GL GUI

When switching windows with Web GUI task bar the source window is not deactivated

01/13/2016 10:41 AM - Hynek Cihlar

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 - 01/13/2016 10:46 AM - Hynek Cihlar

JS client doesn't send window activation/deactivation messages in some cases, which causes the subjected window to not properly activate/deactivate itself. The following is the list of known defective use cases.

1. Multiple windows are visible, windows are switched with the task bar.
2. A window is displayed and not activated, the window title-bar is clicked.

#2 - 01/29/2016 12:11 PM - Sergey Ivanovskiy

The current JS side implementation sends MSG_WINDOW_ACTIVATED (0x0f) message iff the top window is changed by clicking MouseHandler.processWindowFocus. The java peer side on MSG_WINDOW_ACTIVATED posts WindowActivated. If task icons are switched, then MSG_WINDOW_ICONIFY messages are send by the JS driver. The java peer side on MSG_WINDOW_ICONIFY posts WindowEvent(window, Keyboard.SE_WINDOW_MINIMIZED) or WindowEvent(window, Keyboard.SE_WINDOW_RESTORED) depending on the window state notified by the JS side.

Hynek, could you clarify what are the connectedness between these events WindowActivated and WindowEvent(window, Keyboard.SE_WINDOW_MINIMIZED) or WindowEvent(window, Keyboard.SE_WINDOW_RESTORED). Does the windows management logic depend on the special order of these events?

For example, WindowActivated should be the first one and the WindowEvent(window, Keyboard.SE_WINDOW_RESTORED) should be the second, or the opposite statement is hold as the WindowActivated event follows the WindowEvent(window, Keyboard.SE_WINDOW_RESTORED).

#3 - 01/29/2016 01:09 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> he current JS side implementation sends MSG_WINDOW_ACTIVATED (0x0f) message iff the top window is changed by clicking MouseHandler.processWindowFocus. The java peer side on MSG_WINDOW_ACTIVATED posts WindowActivated. If task icons are switched, then MSG_WINDOW_ICONIFY messages are send by the JS driver. The java peer side on MSG_WINDOW_ICONIFY posts

> WindowEvent(window, Keyboard.SE_WINDOW_MINIMIZED) or WindowEvent(window, Keyboard.SE_WINDOW_RESTORED) depending on the window state notified by the JS side.
>
> Hynek, could you clarify what are the connectedness between these events WindowActivated and WindowEvent(window, Keyboard.SE_WINDOW_MINIMIZED) or WindowEvent(window, Keyboard.SE_WINDOW_RESTORED). Does the windows management logic depend on the special order of these events?
>
> For example, WindowActivated should be the first one and the WindowEvent(window, Keyboard.SE_WINDOW_RESTORED) should be the second, or the opposite statement is hold as the WindowActivated event follows the WindowEvent(window, Keyboard.SE_WINDOW_RESTORED).

WindowEvent is used to notify the app layer that a window has been (de)iconified. WindowActivated on the other hand is used to notify about window activation/deactivation. These are separate events. For example, when a window is iconified, WindowActivated must be sent to notify about the window being deactivated and then the WindowEvent to notify the new state.

When deiconified windows are switched using the task bar the driver should only send WindowActivated events for the deactivated and the activated windows.

I suggest you to log the ctor of both WindowActivated and WindowEvent and try the possible cases in Swing GUI. Both drivers should send the same window events in the same order.

#4 - 01/29/2016 01:18 PM - Sergey Ivanovskiy

Thank you, planning to check these cases in Swing Gui and then to simulate window events by the JS driver.

#5 - 01/31/2016 12:15 PM - Sergey Ivanovskiy

Committed revision 10992 should fix these cases. The JS client sends the window active event to the server if the target window should be active driven by the user's activity.

#6 - 01/31/2016 12:16 PM - Sergey Ivanovskiy

- File 2956_window_events_1.txt added

The current 10992 diff is here.

#7 - 01/31/2016 12:56 PM - Greg Shah

- Start date deleted (01/13/2016)

- Assignee set to Sergey Ivanovskiy

- Target version set to Milestone 12

Code Review Task Branch 1811t Revision 10992

I'm fine with the changes. Can I close this task?

#8 - 01/31/2016 01:03 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> The current 10992 diff is here.

Sergey, are you also sending a window-deactivated message for a deactivated window (note that in Java code, we use WindowActivated for both window-activated and window-deactivated events)? That is, when switching windows, first window-deactivated must be sent for the source window and window-activated for the target window.

#9 - 02/01/2016 01:23 AM - Sergey Ivanovskiy

No, window deactivated events aren't sent. It requires to extend MSG_WINDOW_ACTIVATED (0x0f) message by adding new parameter that indicates the target window should be active or not. Planning to fix it.

#10 - 02/01/2016 02:50 AM - Sergey Ivanovskiy

Greg, the current 1811t needs to be rebased because the p2j has improvements in the windows management system (2837). Now ./demo/simple_windows.p breaks to work properly if the JS client sends window deactivated events. Thus I need to rebase 1811t in order to have a consistent code with the current p2j windows management system. Do you agree to rebase 1811t?

#11 - 02/01/2016 05:25 AM - Greg Shah

Yes, go ahead with the rebase. I had hoped to wait for the #2038 changes, but I don't want to hold you up. Go ahead.

#12 - 02/01/2016 07:20 AM - Sergey Ivanovskiy

Starting rebase 1811t at revision 10992.

#13 - 02/01/2016 07:49 AM - Sergey Ivanovskiy

Greg, could you help to resolve history entries. I have only history entries conflicts in

```
Text conflict in src/com/goldencode/p2j/ui/client/gui/driver/web/GuiWebDriver.java
Text conflict in src/com/goldencode/p2j/ui/client/gui/driver/web/GuiWebSocket.java
Text conflict in src/com/goldencode/p2j/ui/client/gui/driver/web/res/p2j.screen.js
3 conflicts encountered.
```

I am confused in the case if my commit to the branch 1811t is first. But the trunc commits have more priorities, don't?

```
**     SBI 20151201 Changed to provide processMouseWidgets with calculated widgets bounds, added
**                  the special EditorGuiImpl case to consider its dimension as its viewport size
**                  in order to fix a mouse cursor type over its scroll bars.
<<<<<<< TREE
** 011 HC  20160104 Added the capability to identify active window in gui drivers.
** 012 EVL 20160106 Adding overlay window support.
** 013 EVL 20160122 Change the logic for setting active window.  Overlay window can not be active
**                  from the higher layer activation point of view.  Removing isWindowActive()
**                  method.  Logic is now in WindowManager.
=======
** 011 SBI 20151208 Added new parameters, scroll amount and scroll unit for wheel events and
**                  a mouse modifiers mask for all mouse events.
```

```
>>>>>>> MERGE-SOURCE
```

Should these entries be kept as they are in the trunc?

#14 - 02/01/2016 08:01 AM - Greg Shah

> But the trunc commits have more priorities, don't?

Yes.

> Should these entries be kept as they are in the trunc?

Yes.

Please change your history entry to be H014. It doesn't matter that the date is earlier.

#15 - 02/01/2016 09:12 AM - Sergey Ivanovskiy

Thank you, the rebase is done. The committed revision 11004 has several text conflicts in the history entries and one change in

```
=== modified file 'src/com/goldencode/p2j/ui/client/gui/OverlayWindow.java'
--- src/com/goldencode/p2j/ui/client/gui/OverlayWindow.java    2016-01-25 17:05:09 +0000
+++ src/com/goldencode/p2j/ui/client/gui/OverlayWindow.java    2016-02-01 14:03:44 +0000
@@ -674,7 +674,7 @@
    @Override
    public void drawBorder(GuiDriver gd, NativeDimension d)
    {
-      GuiWindowCommons.drawBorder(gd, d);
+      GuiWindowCommons.drawBorder(gd, d, this);
    }

    /**
```

#16 - 02/01/2016 11:12 AM - Sergey Ivanovskiy

This code of GuiWebDriver is logically confused

```
   public void windowActivated(int windowId)
   {
      // resolve window object
      TopLevelWindow<?> window = (TopLevelWindow<?>) WindowManager.findWindow(windowId);

      // this protects from activation already active window
      // it become important when overlay window is on screen
      // in this case double activation will incorrectly dismiss overaly
      TopLevelWindow<?> currActiveWindow = WindowManager.getActiveWindow();

      // check if overlay window exists
      OverlayWindow ow = WindowManager.findOverlayWindow();

      // we do not need to activate already active window is there is no overlay on the screen
      if (window == currActiveWindow && ow == null)
      {
         return;
      }

      // if overlay window exists and not a target of activation - remove it first
      if (ow != null && ow != window)
      {
         ThinClient.getInstance().postOSEvent(new WindowActivated(ow, false, false));
      }
      else
      {
         ThinClient.getInstance().postOSEvent(new WindowActivated(window, true, false));
      }
   }
```

It is supposed that it will send the window active event for the target window with its windowId, but if the overlay window is present, this code sends only window deactivated event for the overlay window. This windows management logic looks unclear.

#17 - 02/01/2016 11:42 AM - Eugenie Lyzenko

Sergey Ivanovskiy wrote:

> This code of GuiWebDriver is logically confused
> [...]
> It is supposed that it will send the window active event for the target window with its windowId, but if the overlay window is present, this code sends only window deactivated event for the overlay window. This windows management logic looks unclear.

The new activation subsystem ideas implementation has been completely described in #2837.

Note the sections where noted the case the overlay window exist and shown. There are to layers of activation subsystem. The P2J code as client of the activation engine works on high layer. The main window remains active from the P2J code point of view. More details can be found in appropriate sections of the 2837 task.

#18 - 02/01/2016 11:43 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> This code of GuiWebDriver is logically confused
> [...]
> It is supposed that it will send the window active event for the target window with its windowId, but if the overlay window is present, this code sends only window deactivated event for the overlay window. This windows management logic looks unclear.

I think the code attempts to compensate for the missing events not sent from JS. It will probably need some rework to make it work correctly with your changes.

Consider the following cases.

Case 1:
Main window A has a combo, user clicks on the drop-down button (gui driver sends WindowActivated for the overlay being activated), drop-down is displayed, user clicks on Main window B (gui driver sends WindowActivated for the overlay being deactivated, another WindowActivated for the Main window A being deactivated and yet another WindowActivated for window B being activated).

Case 2:
Main window A has a combo, user clicks on the drop-down button (gui driver sends WindowActivated for the overlay being activated), drop-down is displayed, user clicks to Main window A outside of the drop-down (gui driver sends WindowActivated for the overlay being deactivated). Note that at the app-level we don't consider the main window to change its activation status.

#19 - 02/01/2016 01:02 PM - Sergey Ivanovskiy

Thanks, after all your explanations the windows management logic becomes more clear for me. The main window means the owner of an overlay window, doesn't it? I added a combobox to the ./demo/simple_windows.p to test with an overlay window and found a new JS specific bug. If we display the combobox with its overlay window and then collapse it, than its JS canvas isn't deleted. Planning to fix this bug too.

#20 - 02/01/2016 01:05 PM - Eugenie Lyzenko

Sergey Ivanovskiy wrote:

> Thanks, after all your explanations the windows management logic becomes more clear for me. The main window means the owner of an overlay window, doesn't it?

Yes, exactly.

#21 - 02/02/2016 01:05 AM - Sergey Ivanovskiy

- File 2956_window_events_3.txt added

Committed revision 11005. This revision has the following found bugs reproduced for ./demo/simple_windows.p
1) The frame titled "second frame" doesn't have the initial location coordinates, but the frame "P2j GUi Client" has the correct coordinates.
2) At the first appearance of these two windows if we minimize the frame "P2j GUi Client", than the second frame gets the "iconify" message from the java side. It should not be, because its initial state isn't minimized.
3) If we add a combobox to the frame "second frame", than its canvas isn't removed after the dropdown of this combobox is collapsed.

#22 - 02/02/2016 04:36 AM - Sergey Ivanovskiy

Clicking on the minimized caption button can invoke the resize action for the window under the caption button if the minimized window is on the top of the other.

#23 - 02/02/2016 05:01 AM - Sergey Ivanovskiy

Cleanup the committed revision 11007. I have troubles with the current windows management system, the lost events, widgets focuses. It needs to debug/log all systems layers from java to JS again and again.

#24 - 02/02/2016 12:44 PM - Sergey Ivanovskiy

Hynek, please could you clarify the statement

```
Window activeWindow = WindowManager.getActiveWindow();
```

Should the active window returned by this call have its title bar highlighted? If there is only one window and this window is minimized, than this method returns this minimized window, doesn't it?

#25 - 02/02/2016 01:22 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek, please could you clarify the statement
> [...]
> Should the active window returned by this call have its title bar highlighted?

Not in general. WindowManager.getActiveWindow() should follow the semantics of ACTIVE-WINDOW 4GL system handle. That is it returns the last activated window. So when you switch out to a non-P2J window then this method will still return the last active P2J window.

> If there is only one window and this window is minimized, than this method returns this minimized window, doesn't it?

Good question, frankly I am not sure. I guess it will return either the minimized window or CURRENT-WINDOW assuming no other P2J window will be activated after the window is minimized.

To figure out whether a window should be OS-active (and the title bar highlighted) you should call WindowManager.isWindowActive().

#26 - 02/02/2016 01:34 PM - Sergey Ivanovskiy

Let us consider the most simple case with only one window. The actual situation is that we send an event from JS client that indicates the window should be restored, but the internal state of WindowManager isn't changed after this window has been minimized and currentActiveWnd points to the same window. Should the GuiWebDriver manage the focused window, for example if the window has the current focus is planned to be deactivated, then the driver can clear the current focus window, can't it?

#27 - 02/02/2016 01:45 PM - Sergey Ivanovskiy

Let us consider the activation usecase. The current window is minimized, the current focus window is cleared. The JS client sends an activating event to the GuiWebDriver. Who is responsible for setting the current focus and the current activation?
Can the web gui driver execute this in order to activate the target window

```
WindowManager.setFocusWindow(window)
ThinClient.getInstance().postOSEvent(new WindowActivated(window, true, false));
```

#28 - 02/02/2016 02:00 PM - Sergey Ivanovskiy

Hynek, I think we should develop the general conception for all usecases: with one window, two windows, one window that opens a modal window and more cases. The SwingGuiDriver listens the OS focus events, but in the web case the focus gained events aren't supported. Now, I have no such conception, do you have one, please share it.

#29 - 02/02/2016 02:20 PM - Sergey Ivanovskiy

- File 2956_window_events_4.txt added

#30 - 02/02/2016 02:27 PM - Sergey Ivanovskiy

Greg, Hynek, Eugenie please look at the committed revision 11008. We need a conception to resolve the cases. Here I considered only the case with one created window and tested it with two windows having comboboxes usecase. But didn't test modal dialogs and more complicated cases with modal dialogs and overlays.

#31 - 02/02/2016 02:59 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Let us consider the activation usecase. The current window is minimized, the current focus window is cleared. The JS client sends an activating event to the GuiWebDriver. Who is responsible for setting the current focus and the current activation?
> Can the web gui driver execute this in order to activate the target window
> [...]

The GUI drivers should be responsible only for sending the correct messages in the correct order. That is WindowActivated and WindowEvent messages. The app-logic above the gui drivers (TopLevelWindow and descendants, WindowManager, etc.) handle the rest.

This means you do not call WindowManager.setFocusWindow() from gui driver but only post the correct message with TC.postOSEvent().

#32 - 02/02/2016 03:03 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek, I think we should develop the general conception for all usecases: with one window, two windows, one window that opens a modal window and more cases. The SwingGuiDriver listens the OS focus events, but in the web case the focus gained events aren't supported. Now, I have no such conception, do you have one, please share it.

In Swing gui driver, the focus gained event is only a synonym for an activated window. Since we have full control of the window system and how it is implemented in JS we can emulate the same behavior there - JS will notify Java that window has been activated as a result of user input for example.

#33 - 02/02/2016 03:14 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
>> Let us consider the activation usecase. The current window is minimized, the current focus window is cleared. The JS client sends an activating event to the GuiWebDriver. Who is responsible for setting the current focus and the current activation?
>> Can the web gui driver execute this in order to activate the target window
>> [...]
>
> The GUI drivers should be responsible only for sending the correct messages in the correct order. That is WindowActivated and WindowEvent messages. The app-logic above the gui drivers (TopLevelWindow and descendants, WindowManager, etc.) handle the rest.
>
> This means you do not call WindowManager.setFocusWindow() from gui driver but only post the correct message with TC.postOSEvent().

I don't agree because SwingEmulatedWindow uses WindowManager. It isn't the problem to move this code to WebEmulatedWindow, but is it necessarily? From the JS client we send all possible events in correct order, because all these events are generated by user's actions.

#34 - 02/02/2016 03:23 PM - Hynek Cihlar

In general the GUI driver window logic should be pretty simple

- a window is activated/deactivated as a result of a user input => the driver posts WindowActivated message
- a window is requested to be activated by GuiDriver.moveToTop() => the driver processes the request and posts WindowActivated message
- a window is iconified/deiconified as a result of user input => the driver posts WindowActivated message and WindowEvent message
- when a modal window is active, all other windows must not process user input.

#35 - 02/02/2016 03:30 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
>> Sergey Ivanovskiy wrote:
>>
>>> Let us consider the activation usecase. The current window is minimized, the current focus window is cleared. The JS client sends an activating event to the GuiWebDriver. Who is responsible for setting the current focus and the current activation?
>>> Can the web gui driver execute this in order to activate the target window
>>> [...]

> > The GUI drivers should be responsible only for sending the correct messages in the correct order. That is WindowActivated and WindowEvent messages. The app-logic above the gui drivers (TopLevelWindow and descendants, WindowManager, etc.) handle the rest.
> >
> > This means you do not call WindowManager.setFocusWindow() from gui driver but only post the correct message with TC.postOSEvent().
>
> I don't agree because SwingEmulatedWindow uses WindowManager.

It uses WindowManager only to (1) get to the registered windows (it needs to resolve window instances from window ids), (2) to emulate modal windows and (3) manage z-order.

(1) is also needed in web gui driver.
(2) since we implement the window system in JS ourselves no such logic is needed in web gui driver.
(3) again this is Swing specific, we have to correct z-order because Swing in some cases changes it by itself.

#36 - 02/02/2016 03:49 PM - Eugenie Lyzenko

> In Swing gui driver, the focus gained event is only a synonym for an activated window.

This is not always the true. If window does not have focusable widget it still can be active not having the input focus.

> I don't agree because SwingEmulatedWindow uses WindowManager. It isn't the problem to move this code to WebEmulatedWindow, but is it necessarily? From the JS client we send all possible events in correct order, because all these events are generated by user's actions.

1. We need to strongly keep activation events sequence order. Passing event to TC is a good way. Direct calling to WindowManager is a way to become out of sync in several conditions.
2. There are two activation subsystem layers, one is the OS wide(we gets the message from it in SwingEmulatedWindow or JS endpoint). The other layer is the way we work with activation inside P2J code is dependent on window active state. The P2J application code takes events from TC event queue, not from low level OS based activation.
3. From the P2J concept overlay window is not active, while from OS point of view it is active and this is one example when two layers approach works. When drawing the window title window should ask the WindowManager if from P2J point of view the window is active, because from OS point of view the answer will be wrong(when the overlay is on).

#37 - 02/02/2016 03:57 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> > In Swing gui driver, the focus gained event is only a synonym for an activated window.
>
> This is not always the true. If window does not have focusable widget it still can be active not having the input focus.
>
> > I don't agree because SwingEmulatedWindow uses WindowManager. It isn't the problem to move this code to WebEmulatedWindow, but is it necessarily? From the JS client we send all possible events in correct order, because all these events are generated by user's actions.
>
> 1. We need to strongly keep activation events sequence order. Passing event to TC is a good way. Direct calling to WindowManager is a way to become out of sync in several conditions.

You are right, that is why TC.invokeLater() is used there.

> 2. There are two activation subsystem layers, one is the OS wide(we gets the message from it in SwingEmulatedWindow or JS endpoint). The other layer is the way we work with activation inside P2J code is dependent on window active state. The P2J application code takes events from TC event queue, not from low level OS based activation.

Yes.

> 3. From the P2J concept overlay window is not active, while from OS point of view it is active and this is one example when two layers approach works. When drawing the window title window should ask the WindowManager if from P2J point of view the window is active, because from OS point of view the answer will be wrong(when the overlay is on).

Yes, this is how it is implemented now, at least for the Swing stack.

#38 - 02/02/2016 04:22 PM - Sergey Ivanovskiy

Thanks for your shares. If my understanding is correct the code that manages to work this method WindowManager.isWindowActive() is from TopLevelWindow.public void processEvent(Event event)

```
            // do not change the active window if overlay is activating
            if (!(oldActive != null && oldActive.isOverlay()) ||
                (isOverlay() && activated.withNativeWindow()))
            {
               WindowManager.setActiveWindow(null);
            }
```

In WebGuiDriver we post activating events with unset "withNativeWindow" parameter. That is the problem, because of this I tried to use WindowManager.setFocusWindow(). Planning to find another way to set "withNativeWindow" correctly.

#39 - 02/02/2016 07:19 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Thanks for your shares. If my understanding is correct the code that manages to work this method WindowManager.isWindowActive() is from TopLevelWindow.public void processEvent(Event event)
> [...]

In Web GUI you cannot have an active non-P2J window, so isn't the code branch with WindowManager.setActiveWindow(null) irrelevant for Web stack?

#40 - 02/03/2016 12:22 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Thanks for your shares. If my understanding is correct the code that manages to work this method WindowManager.isWindowActive() is from TopLevelWindow.public void processEvent(Event event)
> > [...]
>
> In Web GUI you cannot have an active non-P2J window, so isn't the code branch with WindowManager.setActiveWindow(null) irrelevant for Web stack?

I think the task icon of the minimized window is the case of the active non-P2j window.

#41 - 02/03/2016 12:32 AM - Sergey Ivanovskiy

Does the code of WindowManager implement the specific Swing logic?

#42 - 02/03/2016 02:44 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Does the code of WindowManager implement the specific Swing logic?

The implementation is certainly affected by some of the Swing limitations, for example restackWindows() is there to work around the limited z-order capabilities, but the code doesn't and should not depend on Swing or any other UI backend employed by any specific GUI driver.

#43 - 02/03/2016 02:51 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Sergey Ivanovskiy wrote:
> >
> > > Thanks for your shares. If my understanding is correct the code that manages to work this method WindowManager.isWindowActive() is from TopLevelWindow.public void processEvent(Event event)
> > > [...]
> >
> > In Web GUI you cannot have an active non-P2J window, so isn't the code branch with WindowManager.setActiveWindow(null) irrelevant for Web stack?
>
> I think the task icon of the minimized window is the case of the active non-P2j window.

Can you describe the case? What are the expected and what are the actual results?

#44 - 02/03/2016 05:26 AM - Sergey Ivanovskiy

Sorry for this delay. The considered case is one window that is minimized and then restored by clicking on the task icon.
WindowManager.isActiveWindow() is useless here because it will return true in this case, unless some code will reset this value.

#45 - 02/03/2016 05:34 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Sorry for this delay. The considered case is one window that is minimized and then restored by clicking on the task icon.
> WindowManager.isActiveWindow() is useless here because it will return true in this case, unless some code will reset this value.

WindowManager.isActiveWindow() is used during window draw. But when it is minimized the window should not be drawn anyway, or at least the canvas is not visible. When the window is restored, then the driver should send WindowActivated message to the respective TopLevelWindow Java counterpart and the window should correctly draw as active.

#46 - 02/03/2016 07:30 AM - Sergey Ivanovskiy

The Swing simple case with one window produces these events on minimize/restore
Deiconify

```
windowDeiconified java.awt.event.WindowEvent[WINDOW_DEICONIFIED,opposite=null,oldState=0,newState=0] on frame0
WindowEvent [source=WindowGuiImpl [id=1, title=], action=RESTORED, origin=-1, nextEvent=<null>, ]
WindowActivated [source=WindowGuiImpl [id=1, title=], activated=true, withNativeWindow=true]
```

Iconify is slightly different in order because mimimize event here is initiated by the caption button WindowGuiImpl.iconify()

```
WindowEvent [source=WindowGuiImpl [id=1, title=], action=MINIMIZED, origin=-1, nextEvent=<null>, ]
windowIconified java.awt.event.WindowEvent[WINDOW_ICONIFIED,opposite=null,oldState=0,newState=0] on frame0
WindowActivated [source=WindowGuiImpl [id=1, title=], activated=false, withNativeWindow=true]
```

The parameter "withNativeWindow" is true in this usecase.

#47 - 02/03/2016 08:15 AM - Sergey Ivanovskiy

In the case of two windows (./demo/simle_windows.p)
Restoring the "P2Gui Client" window and then the "second window" produces these activation events

```
windowDeiconified java.awt.event.WindowEvent[WINDOW_DEICONIFIED,opposite=null,oldState=0,newState=0] on frame1
WindowEvent [source=WindowGuiImpl [id=1, title=], action=RESTORED, origin=-1, nextEvent=<null>, ]
WindowActivated [source=WindowGuiImpl [id=1, title=], activated=true, withNativeWindow=true]
windowDeiconified java.awt.event.WindowEvent[WINDOW_DEICONIFIED,opposite=null,oldState=0,newState=0] on frame0
WindowEvent [source=WindowGuiImpl [id=27, title=second window], action=RESTORED, origin=-1, nextEvent=<null>,
]
WindowActivated [source=WindowGuiImpl [id=1, title=], activated=false, withNativeWindow=false]
WindowActivated [source=WindowGuiImpl [id=27, title=second window], activated=true, withNativeWindow=false]
```

The first window activation event has the "withNativeWindow" parameter, but the second activation has this parameter unset due to the current focus is moved from the P2j window to another P2j window.

#48 - 02/03/2016 08:42 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> The Swing simple case with one window produces these events on minimize/restore
> Deiconify
> [...]
> Iconify is slightly different in order because mimimize event here is initiated by the caption button WindowGuiImpl.iconify()
> [...]
> The parameter "withNativeWindow" is true in this usecase.

Right, the OS chooses a non-P2J as the next active window, that is why withNativeWindow is set to true. I think it would make sense to also set this flag in your case, when no other P2J window is activated.

#49 - 02/03/2016 10:39 AM - Sergey Ivanovskiy

Eugenie, I found that the Swing version of combobox created the Overlay windows (by clicking the combobox dropdown button), but didn't destroy them and these created windows are registered with different ids. Should I fix this bug unless you are planning to fix it?

#50 - 02/03/2016 10:47 AM - Eugenie Lyzenko

Sergey Ivanovskiy wrote:

> Eugenie, I found that the Swing version of combobox created the Overlay windows (by clicking the combobox dropdown button), but didn't destroy them and these created windows are registered with different ids. Should I fix this bug unless you are planning to fix it?

You can fix it. But let me take a look at the changes you make first.

#51 - 02/03/2016 12:54 PM - Sergey Ivanovskiy

Eugenie Lyzenko wrote:

> Sergey Ivanovskiy wrote:
>
> > Eugenie, I found that the Swing version of combobox created the Overlay windows (by clicking the combobox dropdown button), but didn't destroy them and these created windows are registered with different ids. Should I fix this bug unless you are planning to fix it?
>
> You can fix it. But let me take a look at the changes you make first.

Ok, found two methods that are supposed to clean up resources, but actually they don't clean up.
ComboBoxGuiImpl

```
   public void exitDropDown()
   {
      // call generic cleanup from the superclass
      super.exitDropDown();

      // clean up registry from drop-down widgets added in GUI code
      OutputManager.instance().getRegistry().removeWidgetsRecursive((Widget) dropDown);
   }
```

This method OutputManager.instance().getRegistry().removeWidgetsRecursive((Widget) dropDown) does nothing for the drop down.
The second code is found in OverlayWindow

```
      addFocusListener(new FocusListener()
      {
         @Override
         public void onFocusLost(FocusEvent event)
         {
            destroy();
         }

         @Override
         public void onFocusGained(FocusEvent event)
         {
            // do nothing
         }
      });
```

This listener doesn't get FocusEvent events.
The first method is used and can be repaired but it needs to cast dropDown to DropDownGuiImpl or develops the fabric to delete widgets that is the opposite to the fabric (GuiWidgetFactory) that creates widgets.

#52 - 02/03/2016 01:10 PM - Eugenie Lyzenko

I think the best place is ComboBoxGuiImpl.postprocessDropDown(). Here we can identify OverlayWindow and call

```
...
            gd.deregisterWindow(ow.getId().asInt());
            WindowManager.remove(ow);
...
```

Where ow is the OverlayWindow instance we need to terminate.

#53 - 02/03/2016 02:44 PM - Sergey Ivanovskiy

- File combobox_2.txt added

Eugenie, please look at this diff, it cleans resources for web and gui. Planning to postpone this fix because the root causes are not difficult, but the design conception is a problem here. According to the DropDown classes hierarchy it can be disposed as a widget and as a window.

#54 - 02/03/2016 03:07 PM - Eugenie Lyzenko

Sergey Ivanovskiy wrote:

> Eugenie, please look at this diff, it cleans resources for web and gui. Planning to postpone this fix because the root causes are not difficult, but the design conception is a problem here. According to the DropDown classes hierarchy it can be disposed as a widget and as a window.

Looking good. One concern - need to test how it will work with real drop-down overlay window when closing it:
- by pressing comb-box button while drop-down is opened
- by selecting something in the drop-down by mouse or keyboard
- by click mouse outside the drop-down but inside the P2J window while drop-down is opened, including window title
- by click mouse outside the P2J window with some other native OS window while drop-down is opened

All these closing ways should work without NPE or other issues.

#55 - 02/03/2016 04:37 PM - Sergey Ivanovskiy

- File 2956_window_events_5.txt added

Hynek, please look at the committed revision 11010, the diff is attached, taking into account the following bug found for ./demo/simple_windows.p. If we use TAB/SHIFT+TAB keys, then at the beginning these keys help to move between widgets within one focused window. But after some actions to minimize/restore windows and to open dropdowns the TAB/SHIFT+TAB keys move the focus from the focused window to the next window, and it can be restored after doing minimize/restore for these two windows in order to set some internal focus state again. May be the FocusManager is responsible for it.

#56 - 02/04/2016 05:33 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> If we use TAB/SHIFT+TAB keys, then at the beginning these keys help to move between widgets within one focused window. But after some actions to minimize/restore windows and to open dropdowns the TAB/SHIFT+TAB keys move the focus from the focused window to the next window, and it can be restored after doing minimize/restore for these two windows in order to set some internal focus state again. May be the FocusManager is responsible for it.

Btw. there is already an issue related to focus management, see #2954. It would probably make sense to solve it together.

#57 - 02/04/2016 05:56 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek, please look at the committed revision 11010, the diff is attached, taking into account the following bug found for ./demo/simple_windows.p. If we use TAB/SHIFT+TAB keys, then at the beginning these keys help to move between widgets within one focused window. But after some actions to minimize/restore windows and to open dropdowns the TAB/SHIFT+TAB keys move the focus from the focused window to the next window, and it can be restored after doing minimize/restore for these two windows in order to set some internal focus state again. May be the FocusManager is responsible for it.

Sergey can you please explain in what situations is the parameter focusOut needed?

WebGuiDriver.windowActivated() should not use @WindowManager.getActiveWindow() and WindowManager.isWindowActive(). Both methods have slightly different semantics than "active window" understood by GUI drivers. Also the methods are implemented by means of state influenced by GUI driver, so we have a logical circular reference here.

Rather, web GUI driver should itself keep track of currently active top-level window and should not try to understand business logic implemented by the higher application level. This also means there should be no reference or knowledge about the specifics of overlay windows in GUI drivers.

Currently all the specifics of overlay windows are implemented in WindowManager, OverlayWindow and TopLevelWindow. There is no need to leak this logic down to GUI drivers (unless we identify performance issues with this model).

GUI drivers should only understand these primitives: Window (a canvas in Web GUI), window activation status, window iconification state, window task bar visibility, window z-order, window silence state (used to implement modality).

#58 - 02/05/2016 12:24 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
>> Hynek, please look at the committed revision 11010, the diff is attached, taking into account the following bug found for ./demo/simple_windows.p. If we use TAB/SHIFT+TAB keys, then at the beginning these keys help to move between widgets within one focused window. But after some actions to minimize/restore windows and to open dropdowns the TAB/SHIFT+TAB keys move the focus from the focused window to the next window, and it can be restored after doing minimize/restore for these two windows in order to set some internal focus state again. May be the FocusManager is responsible for it.
>
> Sergey can you please explain in what situations is the parameter focusOut needed?
>
> WebGuiDriver.windowActivated() should not use @WindowManager.getActiveWindow() and WindowManager.isWindowActive(). Both methods have slightly different semantics than "active window" understood by GUI drivers. Also the methods are implemented by means of state influenced by GUI driver, so we have a logical circular reference here.
>
> Rather, web GUI driver should itself keep track of currently active top-level window and should not try to understand business logic implemented by the higher application level. This also means there should be no reference or knowledge about the specifics of overlay windows in GUI drivers.
>
> Currently all the specifics of overlay windows are implemented in WindowManager, OverlayWindow and TopLevelWindow. There is no need to leak this logic down to GUI drivers (unless we identify performance issues with this model).
>
> GUI drivers should only understand these primitives: Window (a canvas in Web GUI), window activation status, window iconification state, window task bar visibility, window z-order, window silence state (used to implement modality).

I think it is a new requirement to forbid the usage of WindowManager.getActiveWindow() and WindowManager.isWindowActive().
WindowManager.getActiveWindow() has been used by the GuiWebDriver before my changes for this reason explained by

```
      // this protects from activation already active window
      // it become important when overlay window is on screen
      // in this case double activation will incorrectly dismiss overlay
```

```
// we do not need to activate already active window is there is no overlay on the screen
```

Ok, I will use the top active window and the current focus window that is known by the JS client.

#59 - 02/05/2016 12:31 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey can you please explain in what situations is the parameter focusOut needed?

```
    * @param    focusOut
    *           The true value indicates that the current focus is moved from a non P2J window to
    *           a P2J window for an activation event or from a P2j window to a non P2j window
    *           for a deactivation event.
```

It is needed to send WindowActivated(window, state, focusOut) event in the explained cases.

#60 - 02/05/2016 01:43 AM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> I think it is a new requirement to forbid the usage of WindowManager.getActiveWindow() and WindowManager.isWindowActive().
> WindowManager.getActiveWindow() has been used by the GuiWebDriver before my changes for this reason explained by
> [...]
> Ok, I will use the top active window and the current focus window that is known by the JS client.

No, the JS client doesn't send extra activation events thus it is not required to use the top active window known to the JS side. Please explain the reason that you don't permit the usage WindowManager.isWindowActive(). Also, I would like to recall about your note #2956-25
Committed revision 11011 removes unused WindowManager.getActiveWindow(). I would like to close this task. This revision contains known bugs but they are not related to this one.
./demo/simple_windows.p
1) The frame titled "second frame" doesn't have the initial location coordinates, but the frame "P2j GUi Client" has the correct coordinates.
2) If we add a combobox to the frame "second frame", than its resources aren't disposed after the dropdown of this combobox is collapsed. (GUI and Web cases)
3) SHIFT, SHIFT + TAB keys move focus out of the current window.

#61 - 02/05/2016 04:02 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Sergey Ivanovskiy wrote:
>
>> I think it is a new requirement to forbid the usage of WindowManager.getActiveWindow() and WindowManager.isWindowActive(). WindowManager.getActiveWindow() has been used by the GuiWebDriver before my changes for this reason explained by
>> [...]
>> Ok, I will use the top active window and the current focus window that is known by the JS client.
>
> No, the JS client doesn't send extra activation events thus it is not required to use the top active window known to the JS side. Please explain the reason that you don't permit the usage WindowManager.isWindowActive().

Leaking app logic into the driver will cause us troubles in the future. Consider for example the upcoming support for tooltips. Most likely the tooltip will be implemented as a kind of overlay window. If this code stays in the driver then it will brake, but if we stick to the simple window model with simple rules common to all drivers, we will end up with a lot more robust and maintainable code.

#62 - 02/05/2016 04:04 AM - Sergey Ivanovskiy

- File extra_activation_1.txt added

Ok, understand. Good news are these lines of the code can be successfully eliminated committed rev. 11012. Now the related bug is fixed except known bugs note 60 and note 22.

#63 - 02/05/2016 04:23 AM - Sergey Ivanovskiy

- File update_vars_with_frame.p added

I would like to notify here even this task is unrelated about the blocking state between main thread

```
Name: main
State: WAITING on com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver@471ae0db
Total blocked: 741  Total waited: 493

Stack trace: 
java.lang.Object.wait(Native Method)
java.lang.Object.wait(Object.java:502)
com.goldencode.p2j.ui.client.gui.driver.AbstractGuiDriver.selectWindow(AbstractGuiDriver.java:1698)
com.goldencode.p2j.ui.client.widget.AbstractWidget.getTextWidth(AbstractWidget.java:2257)
com.goldencode.p2j.ui.client.widget.AbstractWidget.getTextWidthNative(AbstractWidget.java:1815)
com.goldencode.p2j.ui.client.gui.ButtonGuiImpl.width(ButtonGuiImpl.java:1336)
com.goldencode.p2j.ui.client.AbstractButton.dimension(AbstractButton.java:152)
com.goldencode.p2j.ui.client.widget.AbstractWidget.bounds(AbstractWidget.java:315)
com.goldencode.p2j.ui.client.widget.AbstractWidget.isTotallyObscured(AbstractWidget.java:1107)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.lambda$0(GuiWebDriver.java:992)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver$$Lambda$32/1220524164.test(Unknown Source)
java.util.function.Predicate.lambda$and$19(Predicate.java:69)
java.util.function.Predicate$$Lambda$40/2000410491.test(Unknown Source)
java.util.function.Predicate.lambda$and$19(Predicate.java:69)
java.util.function.Predicate$$Lambda$40/2000410491.test(Unknown Source)
java.util.function.Predicate.lambda$and$19(Predicate.java:69)
java.util.function.Predicate$$Lambda$40/2000410491.test(Unknown Source)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1457)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.collectWidgets(GuiWebDriver.java:1453)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.registerMouseWidgets(GuiWebDriver.java:1114)
com.goldencode.p2j.ui.client.TypeAhead.getKeystroke(TypeAhead.java:324)
   - locked com.goldencode.p2j.ui.client.TypeAhead@42c883f0
com.goldencode.p2j.ui.chui.ThinClient.waitForEvent(ThinClient.java:12889)
com.goldencode.p2j.ui.chui.ThinClient.waitForWorker(ThinClient.java:10870)
com.goldencode.p2j.ui.chui.ThinClient.waitFor(ThinClient.java:10451)
com.goldencode.p2j.ui.chui.ThinClient.promptFor(ThinClient.java:9192)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:497)
com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:76)
com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:705)
com.goldencode.p2j.net.Conversation.block(Conversation.java:319)
com.goldencode.p2j.net.Conversation.waitMessage(Conversation.java:257)
com.goldencode.p2j.net.Queue.transactImpl(Queue.java:1128)
com.goldencode.p2j.net.Queue.transact(Queue.java:585)
com.goldencode.p2j.net.BaseSession.transact(BaseSession.java:223)
com.goldencode.p2j.net.HighLevelObject.transact(HighLevelObject.java:163)
com.goldencode.p2j.net.RemoteObject$RemoteAccess.invokeCore(RemoteObject.java:1425)
com.goldencode.p2j.net.InvocationStub.invoke(InvocationStub.java:97)
com.sun.proxy.$Proxy11.standardEntry(Unknown Source)
com.goldencode.p2j.main.ClientCore.start(ClientCore.java:281)
com.goldencode.p2j.main.ClientCore.start(ClientCore.java:102)
com.goldencode.p2j.main.ClientDriver.start(ClientDriver.java:201)
com.goldencode.p2j.main.CommonDriver.process(CommonDriver.java:398)
com.goldencode.p2j.main.ClientDriver.process(ClientDriver.java:95)
com.goldencode.p2j.main.ClientDriver.main(ClientDriver.java:267)
```

web worker thread

```
Name: webtaskworker
State: BLOCKED on com.goldencode.p2j.ui.client.TypeAhead@42c883f0 owned by: main
Total blocked: 463  Total waited: 438

Stack trace: 
com.goldencode.p2j.ui.client.TypeAhead.notifyOSEvent(TypeAhead.java:275)
com.goldencode.p2j.ui.chui.ThinClient.postOSEvent(ThinClient.java:5492)
com.goldencode.p2j.ui.client.gui.driver.EmulatedWindowState.iconifyWindow(EmulatedWindowState.java:229)
com.goldencode.p2j.ui.client.gui.driver.AbstractGuiDriver.iconifyWindow(AbstractGuiDriver.java:1886)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver.windowIconified(GuiWebDriver.java:956)
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebSocket.processBinaryMessage(GuiWebSocket.java:1305)
com.goldencode.p2j.ui.client.driver.web.WebClientProtocol$1.run(WebClientProtocol.java:283)
com.goldencode.p2j.ui.client.driver.web.WebTaskWorker.run(WebTaskWorker.java:79)
```

#64 - 02/05/2016 04:52 AM - Constantin Asofiei

I think the lock in TypeAhead.getKeystroke should be moved from the method into its body, i.e.:

```
   public KeyInput getKeystroke(int msec, boolean honorServerEvents)
   {
      KeyInput event = null;

      int savedMsec = msec;

      ScreenDriver<?> driver = OutputManager.getDriver();

      // mouse events are captured constantly - otherwise, if the drawing is done on a slower
      // system, a mouse press/released/clicked chain will be interrupted, as the released/clicked
      // may be raised while the main thread is busy drawing, but they still need to be processed

      // before starting listening for an OS event, register with the driver all widgets which
      // are ENABLED, VISIBLE and can react to mouse events.  Do this only if the output
      // manager supports mouse events
      if (driver instanceof GuiDriver)
      {
         ((GuiDriver) driver).registerMouseWidgets();
      }

      synchronized (this)
      {
         // the existing try ... finally block here
      }
   }
```

#65 - 02/05/2016 08:12 AM - Sergey Ivanovskiy

Constantin Asofiei wrote:

> I think the lock in TypeAhead.getKeystroke should be moved from the method into its body, i.e.:
> [...]

If your change is applied, then this threads lock isn't reproduced. May I commit it in 1811t?

#66 - 02/05/2016 08:34 AM - Constantin Asofiei

Sergey Ivanovskiy wrote:

> Constantin Asofiei wrote:
>
> > I think the lock in TypeAhead.getKeystroke should be moved from the method into its body, i.e.:
> > [...]
>
> If your change is applied, then this threads lock isn't reproduced. May I commit it in 1811t?

Yes, go ahead and commit it.

#67 - 02/05/2016 08:54 AM - Sergey Ivanovskiy

Committed revision 11015 resolves the TypeAHead lock between the main and web worker threads.

#68 - 03/01/2016 04:01 PM - Greg Shah

Is there anything left to do on this task?

#69 - 03/22/2016 11:49 AM - Greg Shah

Can I close this task? Or is there something left to do?

#70 - 03/22/2016 11:58 AM - Sergey Ivanovskiy

Yes, it seems that the task is done.

#71 - 03/22/2016 11:59 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

#72 - 11/16/2016 12:12 PM - Greg Shah

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

Files

2956_window_events_1.txt	4.39 KB	01/31/2016	Sergey Ivanovskiy
2956_window_events_3.txt	8.46 KB	02/02/2016	Sergey Ivanovskiy
2956_window_events_4.txt	1.94 KB	02/02/2016	Sergey Ivanovskiy
combobox_2.txt	730 Bytes	02/03/2016	Sergey Ivanovskiy
2956_window_events_5.txt	9.01 KB	02/03/2016	Sergey Ivanovskiy
extra_activation_1.txt	956 Bytes	02/05/2016	Sergey Ivanovskiy
update_vars_with_frame.p	2.16 KB	02/05/2016	Sergey Ivanovskiy