User Interface - Feature #7709

ABLUnit support for web clients

08/16/2023 03:33 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 08/16/2023 03:38 PM - Greg Shah

Our UI tests being written in #6856 need to be executed in both Swing and the web client. Today, the architecture of the web client makes it incompatible with how we implement ABLUnit tests. This task is meant to address that limitation. I think there are two areas that need work:

- Rework the login/spawning approach to work from ABLUnit. ABLUnit is essentially its own custom FWD client process. With a bootstrap cfg, one can specify that the client should be a Swing GUI client. Currently there is no way to make this a web client.
- We need to be able to launch the browser and point it to the web client's URL.

#2 - 08/17/2023 03:37 AM - Galya B

Greg Shah wrote:

> - Rework the login/spawning approach to work from ABLUnit. [..]
> - We need to be able to launch the browser and point it to the web client's URL.

Maybe keeping the WebClientLauncher network interface used to spawn web clients is going to help here? ABLUnit will start a proc that calls the spawn method, receives back the url and runs the web UI test engine with its browser engine as OS-COMMAND and wait for completion. It can work with any tool installed on the same system standalone. The cmd can have the url as arg. On completion the test engine can write a file for 4GL to parse if it's success / failure. On failure the full report will have to be manually read. This is quite a simplistic approach. Requires tests written for that tool specifically. I see Marian has already discussed the approach in #6856-7. I'm not familiar with Sikuli, but what I've described can be achieved with Cypress: the go-to nowadays, very solid web UI testing framework with its own browser engines. Please disregard if not applicable. What concerns me is the first part: keeping the network interface for testing purposes.

#3 - 08/17/2023 04:54 AM - Greg Shah

I was thinking that we already have a client process so we don't need to spawn. I'd prefer to avoid the spawning unless really needed.

The primary reason we provide the URL back to the server is so that the login page response can redirect to the new web client process' Jetty web port. In this case, if we **are** the web client process we have direct access to the code that can tell us the URL, so that should be pretty easy.

In regard to the OS-COMMAND, that is useful for converted 4GL code but I think a more direct approach. We have DesktopHelpers.desktopOpen(). Or we could use the java.awt.Desktop class directly to open a browser.

#4 - 08/17/2023 04:57 AM - Greg Shah

> but what I've described can be achieved with Cypress: the go-to nowadays, very solid web UI testing framework with its own browser engines. Please disregard if not applicable.

Traditional web testing framewors like Selenium or Cypress assume a DOM model and we have no DOM model that maps to our UI controls. All our widgets are drawn on a canvas. Sikuli is different because it operates at the user level (pushing key/mouse input and processing visual outputs). This has the advantage of working with FWD and also possibly working with OpenEdge, so that the same test could run in both places. In practice, doing that is difficult because of differences in our font rendering but the possibility exists.

#5 - 08/21/2023 08:47 AM - Hynek Cihlar

Greg Shah wrote:

> I was thinking that we already have a client process so we don't need to spawn. I'd prefer to avoid the spawning unless really needed.
>
> The primary reason we provide the URL back to the server is so that the login page response can redirect to the new web client process' Jetty web port. In this case, if we **are** the web client process we have direct access to the code that can tell us the URL, so that should be pretty easy.

Shouldn't the spawning process as it is now be also subject to testing?

#6 - 08/23/2023 10:15 AM - Greg Shah

Yes, it is a good point. There is a benefit to testing the spawning process itself.

On the other hand, spawning is more complex and is very dependent on the environment being setup properly. I was thinking we should avoid those complications/fragility but perhaps the benefit of testing spawning outweighs the costs.