Runtime Infrastructure - Feature #8651

Centralize inclusion of JS dependencies

04/18/2024 04:50 AM - Hynek Cihlar

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

Related issues:

Related to Runtime Infrastructure - Feature #6692: move FWD to Java 17	**Internal Test**

History

#1 - 04/18/2024 04:53 AM - Hynek Cihlar

There are specific Java Script dependencies needed for FWD to run in embedded mode. If they are included in a single bundle which will be managed inside FWD itself, it will simplify deployment/upgrades.

The need for an upgrade will come from FWD product upgrade, but also from the vulnerability check, which may happen often.

Does it make sense to centralize dependencies for the other modes, virtual and report server? Perhaps when the customer chooses to provide his own welcome pages.

This issue was identified in #6692-72.

#2 - 04/18/2024 04:54 AM - Hynek Cihlar

- Related to Feature #6692: move FWD to Java 17 added

#3 - 04/22/2024 09:01 AM - Galya B

There are a few places where js is used in FWD:

- On the default login page for Virtual Desktop and Embedded modes: It uses only our custom vanilla js sdk for server connection fwd_sdk.js and it should stay as a stand-alone file. No dependencies.
- After the login page for Virtual Desktop and Embedded modes. The imports are used in src/com/goldencode/p2j/ui/client/driver/web/index.html and served by the client process web server in EmbeddedWebServerImpl:
  - Only two external dependencies, exposed as context paths mapped to resource dirs by dedicated web handlers: dojo-toolkit and gettext.js.
  - A lot of custom FWD web resources from common and client.
- Reporting server exposes via ReportWebServer.ThirdPartyDependenciesHandler a lot of 3rd party dependencies used by src/com/goldencode/p2j/report/web/res/index.html:
  - dojo-toolkit, d3, jquery, jquery-ui, bootstrap, tabulator, cbtree, adamwdraper-Numeral-js, webcola, virtualscroller, jquery-loading-overlay.
  - The exposed resources include both .js and .css files.
- Custom Embedded driver pages in hotel_gui (in recent versions it's /webres/web_partial_login_embedded.html) and all customer codes using Embedded driver, who mostly haven't migrated to recent FWD versions. The FWD server process exposes the 3rd party lib dirs in EmbeddedWebHandler:
  - dojo-toolkit, d3, jquery, jquery-ui, bootstrap, tabulator, custom/moment?.
  - Two libs are exposed in EmbeddedWebHandler, but don't seem to be used anywhere in FWD's code: material-design-icons, material-components-web.

Some observations:

- gettext.js is used only by the web server in the client process in one front-end page.

- dojo-toolkit is a common lib dependency in all front-ends and it's a main .js file with packages and multiple .css files to be explicitly imported as part of the configuration, so the whole module / dir should be shared.
- Reporting module and custom embedded pages share some dependencies, but not all. Also they use different web servers and are independent in configuration.
- The customer projects using embedded pages (including hotel_gui) need to be free from any dependency coming from the FWD version. Otherwise, each update to the reporting dependencies or the base driver page will break the customer front-ends.
- Currently the classpaths for server, client, reporting are all the same, the lib dir.

So my proposal would be:

- Decouple hotel_gui from the server dependencies and give different examples to customers who need embedded enabled. Remove the handlers from EmbeddedWebHandler.
- Move reporting dependencies out of the server classpath and add them to the classpath in report.sh. Optionally for convenience rework ReportWebServer.ThirdPartyDependenciesHandler to serve unified and minified js and css files with the necessary resources. There are some ant processors that can do it. The trouble is that in js projects dependencies are auto-selected, while here I'm not sure if it will require to explicitly list the dependencies paths in the build task. If this can't be configured automatically, then the trouble of updating the versions in two places will still exist and the rework won't achieve much. Generally leaving separate script src / css links in html helps with caching, so it has its merits. If this is not solved, the effort might be unnecessary.
- In the client dependencies only dojo-toolkit and gettext.js will be left.
- In the server dependencies no js libs will be required.

#4 - 04/22/2024 07:23 PM - Greg Shah

Nice analysis!

I'm generally OK with the overall recommendations.

> Decouple hotel_gui from the server dependencies

I would like to achieve a default embedded mode that implements a clean and extendable UI and which is built-in to FWD. We have a prototype of it in one customer application. We may be able to achieve this for ADM/ADM2 applications which would cover a wide range. The Hotel GUI approach is much more customized and has quite a bit of extra exammples of JS code that integrates with the converted code. I'm not against the decoupling, but I do think some of these dependencies will be required for the common code.

On the other hand, we don't have that in place yet. Mostly, I don't want to make the long term objective harder.

> and give different examples to customers who need embedded enabled.

Can you clarify what you mean by this? What are the different examples?

> Remove the handlers from EmbeddedWebHandler.

I wonder if we need this for the long term dependencies.

#5 - 04/23/2024 02:32 AM - Galya B

Greg Shah wrote:

> Nice analysis!
>
> I'm generally OK with the overall recommendations.
>
> > Decouple hotel_gui from the server dependencies
>
> I would like to achieve a default embedded mode that implements a clean and extendable UI and which is built-in to FWD. We have a prototype of it in one customer application. We may be able to achieve this for ADM/ADM2 applications which would cover a wide range. The Hotel GUI approach is much more customized and has quite a bit of extra examples of JS code that integrates with the converted code. I'm not against the decoupling, but I do think some of these dependencies will be required for the common code.

If there is such a case where customers want an embedded mode with a standard front-end, it makes sense to keep the dependencies and they will be maintained by us both on the front-end and the back-end, so updates of FWD versions will keep things in sync. I guess such a choice is just to achieve a more modern look without any efforts on their side.

> On the other hand, we don't have that in place yet. Mostly, I don't want to make the long term objective harder.
>
> > and give different examples to customers who need embedded enabled.
>
> Can you clarify what you mean by this? What are the different examples?

I was imagining though that a customer who wants embedded, wants to integrate it with an existing front-end project or one they will develop independently of FWD. In which case it will be unreasonable to provide them dependencies with FWD. In my imagination a customer may want to develop a modern React app, or Vue.js, or use another popular framework and insert FWD in it. And we can develop a few examples of hotel_gui integrated with popular front-end stacks. But it's an effort not worth it atm, if there are no signs of such intentions in customers.

> > Remove the handlers from EmbeddedWebHandler.
>
> I wonder if we need this for the long term dependencies.

This all comes down to the question if there is indeed such a case where customers want a standard embedded front-end that only GCD will support. I wasn't familiar with this possibility.

#6 - 04/23/2024 03:08 PM - Greg Shah

> If there is such a case where customers want an embedded mode with a standard front-end, it makes sense to keep the dependencies and they will be maintained by us both on the front-end and the back-end, so updates of FWD versions will keep things in sync.

All of our GUI customers have been interested in embedded mode. Half of our GUI customers don't have existing web applications in which to embed the existing 4GL GUI screens. The other half do have their own web application.

> I guess such a choice is just to achieve a more modern look without any efforts on their side.

Correct. The idea is that we would give them a good starting point to evolve from.

> Can you clarify what you mean by this? What are the different examples?

> I was imagining though that a customer who wants embedded, wants to integrate it with an existing front-end project or one they will develop independently of FWD. In which case it will be unreasonable to provide them dependencies with FWD. In my imagination a customer may want to develop a modern React app, or Vue.js, or use another popular framework and insert FWD in it. And we can develop a few examples of hotel_gui integrated with popular front-end stacks. But it's an effort not worth it atm, if there are no signs of such intentions in customers.

About half of the customers are interested in the existing front-end scenario. I don't know that we need multiple examples, but migrating Hotel GUI to be a custom app that embeds a more standardized SDK approach would be useful.