User Interface - Feature #3289

implement SYSTEM-DIALOG-GET-DIR

04/27/2017 02:56 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Bug #1830: implement SYSTEM-DIALOG-GET-FILE support	**Closed**

History

#1 - 07/24/2017 09:16 AM - Ovidiu Maxiniuc

- Related to Bug #1830: implement SYSTEM-DIALOG-GET-FILE support added

#2 - 07/24/2017 09:20 AM - Ovidiu Maxiniuc

- Status changed from New to WIP

- Start date deleted (04/27/2017)

Added conversion support in task branch 1830a, revision 11159.
The full-syntax construct like this:

```
SYSTEM-DIALOG GET-DIR sourcefile
   INITIAL-DIR CAPS("D") + ":" + "\" 
   RETURN-TO-START-DIR
   TITLE "Select " + caps("start") + " directory".
```

will be converted in java as:

```
new FileDialog(sourcefile)
          .setInitialDir(concat(toUpperCase("D"), ":", "\\"))
          .returnToStartDir()
          .setTitle(concat("Select ", toUpperCase("start"), " directory"))
          .getDirectory();
```

Notes:

- the constructor handles the mandatory parameters;
- each of the next method calls (except getDirectory) from the chaining represents an option that sets up the server-side configuration of the statement;
- finally, the getDirectory method does the actual call on client side that will open the chooser dialog, based on options selected.

#3 - 07/24/2017 09:30 AM - Greg Shah

1. I prefer the static SomeClass.create*Dialog() + chaining approach over than the constructor + chaining approach. Is there a reason not to go with that (static method) for the file and dir dialogs?

2. How will the code differentiation between the directory dialog and the file dialog? I would prefer different class names or different static method (createFileDialog()/createDirectoryDialog()) names to make it obvious from the generated code.

#4 - 07/24/2017 09:30 AM - Greg Shah

From Ovidiu:

> I can unify the paradigm of file/dir chooser dialogs with the other system dialogs. I kept the c'tor as the head of the chain because this was the how I first started the implementation for GET-FILE, and I didn't update. When adding support for GET-DIR I noticed the options are similar (the GET-DIR options form a subset of GET-FILE options) and I though to reuse the same code. To differentiate between the two, I emit two different 'execute' methods: getFile() and getDirectory(), respectively.
>
> If you consider to be more visible, I can replace the common constructor with dedicated static methods (FileSystemDaemon.createFileChooserDialog() and FileSystemDaemon.createDirChooserDialog()) and use a common execute() method as the last element in chain that will do the actual work.

#5 - 07/24/2017 09:36 AM - Greg Shah

> I can unify the paradigm of file/dir chooser dialogs with the other system dialogs. I kept the c'tor as the head of the chain because this was the how I first started the implementation for GET-FILE, and I didn't update.

I understand.

Yes, please switch those to the static method approach.

> If you consider to be more visible, I can replace the common constructor with dedicated static methods (FileSystemDaemon.createFileChooserDialog() and FileSystemDaemon.createDirChooserDialog()) and use a common execute() method as the last element in chain that will do the actual work.

Yes, go with this.

I do think it is more visible (I had missed the getFile() and getDirectory() at the end so I didn't see the differentiation.

The converted code should not call the FileSystemDaemon. That code is client-side only. Also I don't want to UI code mixed in with the non-UI code in util. Please implement the server-side dialog code in the com.goldencode.p2j.ui package. The client side code that implements the dialog can be in com.goldencode.p2j.ui.client can can call the FileSystemDaemon to get lists of files/directories etc... as needed. So the file system access can be hidden in helpers in the FileSystemDaemon, but the interactive code should be in the ui packages.

#6 - 11/01/2017 02:22 PM - Greg Shah

The runtime implementation for this task was written in branch 1830b and was merged to trunk as revision 11188.

An update from Ovidiu about the code in 1830b:

> It was working only with mouse. I added support for keyboard navigation yesterday. By the end of day I will add content for some virtual folders (This PC, Libraries), when possible. With this the work for the task should be finished.

In what task branch do the changes exist?

#7 - 11/01/2017 02:28 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> The runtime implementation for this task was written in branch 1830b and was merged to trunk as revision 11188.
>
> An update from Ovidiu about the code in 1830b:
>
> > It was working only with mouse. I added support for keyboard navigation yesterday. By the end of day I will add content for some virtual folders (This PC, Libraries), when possible. With this the work for the task should be finished.
>
> In what task branch do the changes exist?

I waited for the trunk to settle after recently multiple commits. I've created 1830c today and I will commit the changes there.

#8 - 03/23/2018 03:04 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

- Assignee set to Ovidiu Maxiniuc

Branch 1830c was committed to trunk as r11240.