User Interface - Feature #2487

Feature # 2252 (Closed): implement GUI client support

Feature # 2486 (Closed): enhance editor functionality

implement READ-FILE() and SAVE-FILE() methods in the editor widget

01/13/2015 08:32 PM - Greg Shah

Status:	Closed	**Start date:**	01/13/2015
Priority:	Normal	**Due date:**	
Assignee:	Evgeny Kiselev	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 01/13/2015 09:06 PM - Greg Shah

The READ-FILE() method is already supported in methods_attributes.rules (and the runtime stubs are in the EditorInterface and EditorWidget), but SAVE-FILE() has neither conversion nor runtime stub support.

Both methods need to have a full and compatible runtime implementation. Take into account the fact that the files need to be both searched (in the propath) and loaded/saved on the client. Please minimize round trips, perhaps you should handle both the searching and the loading/saving in a single call to the client. That call should be synchronous with the respective method call on the server.

The Windows vs UNIX/Linux line end support should be handled properly to ensure binary compatibility with the 4GL.

If it makes sense (as long as it is functionally compatible), use our existing client support for searching the propath (FileSystemDaemon.searchPath()) and for reading/writing files.

#2 - 02/10/2015 12:22 PM - Evgeny Kiselev

- File evk_upd20150209a.zip added

Added conversion support for SAVE-FILE method

#3 - 02/10/2015 05:59 PM - Evgeny Kiselev

Just want to clarify:
I planed to add in the LogicalTerminal new methods for read/save files. Also I need to add some Map to save opened files (or I need to save this data in config ?)
methods like SEARCH and REPLACE are not implemented yet ?

#4 - 02/10/2015 06:18 PM - Greg Shah

> I planed to add in the LogicalTerminal new methods for read/save files.

That is OK. These should call down to a new ClientExports interfaces specifically for EDITOR: editorReadFile() and editorSaveFile(). The ThinClient counterpart would just redirect these method calls to the specific Editor instance specified. The methods should pass an int widget ID to identify which editor is being used.

> Also I need to add some Map to save opened files (or I need to save this data in config ?)

I don't think either of these methods actually leave the files open when they are done. The work must be delegated to the client. For READ-FILE() the client-side editor code should use FileSystemDaemon.searchPath() to find the file on the client's filesystem. The found file is opened, read and then closed. The contents that are read are put into the editor. Please test this with various inputs (empty files, very large files, filename that is unknown, filename that is empty, filename that doesn't exist, filename that exists in multiple relative paths and must be found via propath...). Make sure to match the error handling.

The SAVE-FILE() just opens the specified file on the client-side (which may be a relative or absolute filename), writes the contents of the editor to it and closes the file. Please test this like the read is tested (lots of bad inputs) and duplicate its error processing.

I don't see any need for a map or any reason to store anything in the config.

> methods like SEARCH and REPLACE are not implemented yet ?

Do you mean the editor methods editor-handle:SEARCH() and editor-handle:REPLACE()? No, they are not implemented yet. If you need them for your testcases, you can implement them as well.
REPLACE

#5 - 02/10/2015 06:20 PM - Greg Shah

Code Review evk_upd20150209a.zip

Looks good so far.

#6 - 02/18/2015 05:31 PM - Evgeny Kiselev

Can you suggest what is common "looking" directories in 4gl ? at least common work dir I can take from Configuration.home(). Also there are Configuration.getParameter("propath") directories. Is it enough ?

#7 - 02/18/2015 05:34 PM - Greg Shah

I don't think you need to do anything like that. The client-side code would call FileSystemDaemon.searchPath(), which already handles searching the propath as configured in P2J.

Am I misunderstanding what you are asking?

#8 - 02/18/2015 05:38 PM - Evgeny Kiselev

Greg Shah wrote:

> I don't think you need to do anything like that. The client-side code would call FileSystemDaemon.searchPath(), which already handles searching the propath as configured in P2J.
>
> Am I misunderstanding what you are asking?

You are right. Problem was in my testcase. I have file in work dir without any extension (just name a). and code in
com.goldencode.p2j.util.FileSystemDaemon.search():
if (file.exists() && (file.isFile() || allowDir)) was false for me.
File with extension work fine.

#9 - 02/18/2015 05:41 PM - Greg Shah

Is there a bug in our implementation of the 4GL built-in function SEARCH() that makes it work differently with files that have no extension?

If your testcase works in the 4GL and not in P2J, then that sounds like a bug we need to fix.

#10 - 02/19/2015 08:00 PM - Evgeny Kiselev

- File evk_upd20150218a.zip added

#11 - 02/19/2015 08:02 PM - Evgeny Kiselev

Added runtime support. I'm not finished testing at the moment, but common functionality is working.

#12 - 02/20/2015 01:36 PM - Greg Shah

Code Review evk_upd20150218a.zip

It is a good step.

1. EditorWidget.read/saveFile(character) need null and unknown value protection (there is no need to call down to the client in that case).

2. I suspect that ThinClient.editorRead/SaveFile() error handling may differ from how the 4GL does it. For example, if an IOException occurs on reading, no content is ever placed in the editor, but in the 4GL what happens?

3. LogicalTerminal.read/saveFile() should have their opening { on the next line.

#13 - 03/01/2015 07:11 PM - Evgeny Kiselev

1) If some IO error while reading/writing, then it read or wrote data can be used, but status of operation will be "no"
2) there is a limit of maximum data that can be stored in editor (2588205 bytes), other data just drop and throw runtime exception:

```
Editor is full. Some lines could not be inserted and have been lost. (5903)
```

maximum stored data depends on INNER-CHARS parameter.

#14 - 03/01/2015 09:20 PM - Greg Shah

Interesting. These sound pretty easy for us to implement.

> maximum stored data depends on INNER-CHARS parameter.

How so?

#15 - 03/03/2015 08:08 PM - Evgeny Kiselev

- File evk_upd20150301a.zip added

1) Added error handling and file stream close
4GL writes data into file till IO error. But for read operation 4gl reads full data and only after tried to set this data into widget.
For example widget has some magic limit for data, and throw exception (see note 13), but this exception I believe thrown while big data is trying to set into widget. It's easy to check:
1) take big file (much more then limit). And time to load this file will be much bigger then time to read file just of size limit + 1.
So I think that error from note 13 should be throw in set data method and not in read-file method.

limit or length depends on:

```
def var v-editor as character view-as editor size 318 by 1 no-box.
/* same */
def var v-editor as character view-as editor inner-chars 318 inner-lines 1 no-box.
```

width size, and not depends on height value.
Widget length

```
 1*1 68280
 1*2 68280
 2*1 103766
 3*1 139232
 4*1 174716
 5*1 174765
 6*1 245657
 7*1 245703
 8*1 245748
 9*1 245787
10*1 245836
11*1 245878

318*1 10254292
```

Widget length is growing very strange. Maximum value that I've found is 318.

Also I need to set modified flag. But didn't find good place to do it. Where I can have access to this flag ?

Full testcase:

```
def var v-editor as character view-as editor size 318 by 1 no-box.
form v-editor with frame fr-one size 350 by 320.
def var c as character.

c = "test.txt".
message "modified before read " v-editor:modified.
message v-editor:READ-FILE(c).
message "modified after read " v-editor:modified.

/*message v-editor:length. */
message "modified before save " v-editor:modified.
message v-editor:SAVE-FILE("test2.txt").
message "modified after save " v-editor:modified.
```

#16 - 03/04/2015 03:58 PM - Greg Shah

> Also I need to set modified flag. But didn't find good place to do it. Where I can have access to this flag ?

Search on config.setModified(). This is directly available in the client-side editor code.

#17 - 03/04/2015 04:24 PM - Greg Shah

Code Review evk_upd20150301a.zip

1. Please name LogicalTerminal.writeFile() as LogicalTerminal.saveFile() to be consistent with the rest of the methods. The javadoc for that method is also incorrect.

2. Please move the logic in ThinClient.editor*File() into the client/Editor widget. The ThinClient methods should just lookup the editor instance and call methods there. ThinClient is already way too big and should not have logic that is editor-specific.

#18 - 03/04/2015 04:26 PM - Greg Shah

> 1) take big file (much more then limit). And time to load this file will be much bigger then time to read file just of size limit + > 1. So I think that error from note 13 should be throw in set data method and not in read-file method.

We don't have to try to read the whole thing to know this is going to generate an error. It would be faster for us to check the file size and compare it with our limit and then generate the error. There is no need for us to actually read the data...

#19 - 03/09/2015 06:53 PM - Greg Shah

Please note this text from the Progress docs for EDITOR Phrase:

```
LARGE

Specifies that ABL use a large editor widget rather than a normal editor widget in
Windows. A normal Windows editor can contain up to 20K of data. The LARGE
option allows the editor to contain data up to the limit of your system resources.
However, it also consumes more internal resources and lacks some functionality.
Use the LARGE option only if you have to edit very large sections of text. The
LARGE option applies only to Windows; other interfaces allow for larger editors by
default. This option is ignored in those other interfaces.

MAX-CHARS characters

The maximum number of characters that can be displayed or entered within the
text editor widget. The value characters must be an integer constant. By default,
ABL does not limit the number of characters (although system limits might apply).
```

And this (MAX-CHARS Attribute):

```
The maximum number of characters an editor or combo-box widget can hold.

...

For editor widgets, you can set this attribute only before the widget is realized. In
Windows, the maximum value of MAX-CHARS is approximately 20K for the regular
editor and over 64K for the large editor.
```

I'm not sure if MAX-CHARS is really available or not in ChUI. If it is (contrary to the docs), then that value might control the read limit. Please run a test to see if this has any bearing on your results.

In regard to the ChUI limits you've found, I'm worried that the values are not reliable (that they may vary version or version, or based on runtime state). What do you propose to limit our exposure here? How might we explore the stability of these limits?

#20 - 03/15/2015 08:36 PM - Evgeny Kiselev

- File evk_upd20150315a.zip added

added error handling and modified state
note 17 requests are added
added limitation logic. There is a good place for map with sizes Editor.maxSizes.

At the moment I don't understand MAX-CHARS property. In my testcases this property is not working at all. Maybe on linux this property is not available.
I couldn't even set some value to this property - it was always unknown.

#21 - 03/16/2015 03:17 PM - Greg Shah

Code Review evk_upd20150315a.zip

I only have some minor requests:

1. Add a history entry to Editor.java.

2. Please move the maxSizes map and corresponding static initializer to the top of the class. Per our coding standards, all data must be defined up there. The static initializer also needs to be up there, because it is part of the class-level initialization and reader will easily miss it if specified at the end of the file. Instead, we typically put such initializers after the member data but just before the constructor(s).

3. Please put a TODO comment above the maxSizes static initializer explaining that the specified values were found on a ChUI-based Linux system and it is not known if these values are universally applicable to all runtime conditions and/or platforms.

4. Please put a TODO comment in both Editor.editorReadFile() and Editor.editorSaveFile() to explain that the client-side stream classes should probably be used for read/write of the files (instead of NIO) to ensure that the I18N behavior of the 4GL is honored.

5. The javadoc for editorSaveFile(int wId, String name) and editorReadFile(int wId, String name) in LogicalTerminal, ClientExports and ThinClient should state that the wId must be the ID of a valid editor widget.

6. In ThinClient.editorSaveFile() and ThinClient.editorReadFile(), please check if the found widget is instanceof Editor and return false if not. This will avoid a ClassCastException if someone ever tries to use that API incorrectly.

#22 - 03/17/2015 09:14 PM - Evgeny Kiselev

fixes from note 21 review

#23 - 03/17/2015 09:15 PM - Evgeny Kiselev

- File evk_upd20150317a.zip added

#24 - 03/17/2015 10:22 PM - Greg Shah

Code Review evk_upd20150317a.zip

It looks good.

Please put it into conversion and runtime regression testing.

#25 - 03/24/2015 08:16 PM - Evgeny Kiselev

update evk_upd20150317 has been passed regression and conversion testing.

#26 - 03/25/2015 08:27 AM - Greg Shah

Great! Please commit it and distribute.

#27 - 03/25/2015 11:57 AM - Greg Shah

Actually, you will have to merge up to the 10821 level. Just do the merge and then upload the zip file. I will review it. I think you don't need to re-test, if the merge is done carefully.

Don't check it in until I approve the merged update.

#28 - 03/26/2015 09:06 AM - Evgeny Kiselev

- File evk_upd20150326a.zip added

merged with latest revision

#29 - 03/26/2015 09:22 AM - Greg Shah

Code Review evk_upd20150326a.zip

I'm OK with the merge. Please check it in and distribute it.

#30 - 03/28/2015 12:33 PM - Greg Shah

- Status changed from WIP to Closed

#31 - 11/16/2016 12:13 PM - Greg Shah

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

Files

evk_upd20150209a.zip	24.9 KB	02/10/2015	Evgeny Kiselev
evk_upd20150218a.zip	238 KB	02/20/2015	Evgeny Kiselev
evk_upd20150301a.zip	242 KB	03/04/2015	Evgeny Kiselev
evk_upd20150315a.zip	258 KB	03/16/2015	Evgeny Kiselev
evk_upd20150317a.zip	258 KB	03/18/2015	Evgeny Kiselev
evk_upd20150326a.zip	258 KB	03/26/2015	Evgeny Kiselev