User Interface - Feature #4077

add option to OPEN-MIME-RESOURCE to delete the original source file when the send is complete

05/08/2019 01:39 PM - Greg Shah

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

Description

Related issues:

Related to User Interface - Feature #3474: OPEN-MIME-RESOURCE	**Closed**

History

#1 - 05/08/2019 01:47 PM - Greg Shah

OPEN-MIME-RESOURCE is often used when 4GL code generates some output and wants to get it to the user. There is a common case where this output is transient and only needs to exist until it has been loaded into the user's browser.

If I recall correctly, today we copy the input content into a temporary location, send it and then delete the temporary copy.

This task is intended to add a DELETE-SOURCE-CONTENT optional keyword to the OPEN-MIME-RESOURCE syntax. If present, we would do one of the following:

- Avoid doing a temporary copy (which should save time) and delete the original input file when the transfer is complete.
- Still do the temporary copy but make sure to delete the input file just before OPEN-MIME-RESOURCE returns.

The result is the same from the 4GL developer's perspective, but if we can save a copy that is good.

#2 - 05/08/2019 01:47 PM - Greg Shah

- Related to Feature #3474: OPEN-MIME-RESOURCE added

#3 - 10/30/2019 10:18 AM - Sergey Ivanovskiy

- Status changed from New to WIP

- Assignee set to Sergey Ivanovskiy

#4 - 10/31/2019 10:50 AM - Sergey Ivanovskiy

The current implementation uses WebBrowserManager.openMimeResource to generate java code for OPEN-MIME-RESOURCE. Planning to add a shortcut key word of DELETE-SOURCE-CONTENT as KW_DEL_S_C according to presenting shortcuts in progress.g with this rule

```
/**
 * Matches the FWD-extension language statement OPEN-MIME-RESOURCE and a mandatory
 * pair of resource mime type character and its url character expression. If an optional
 * DELETE-SOURCE-CONTENT is present, then it forces to delete the original input file
 * when the transfer is complete.
 */
```

```
open_mime_resource_stmt
   :
      KW_OPENMIME^ expr expr (expr)? (KW_DEL_S_C)? stmt_term
   ;
```

Could we add type conversion rules to language_statements.rules for child nodes of this type == prog.kw_openmime in order to use java types?
Now I need to add openMimeResource for all possible combinations of character|String and logical|boolean.
Also there are ambitious parameters "EMBEDED" and "DELETE-SOURCE-CONTENT" boolean flag. Planning to add true value node as a first child node if DELETE-SOURCE-CONTENT is present by this rule

```
      <rule>type == prog.kw_del_s_c and
            parent.type == prog.kw_openmime
         <action>createJavaAst(java.bool_true, "true", closestPeerId, 0)</action>
      </rule>
```

#5 - 11/01/2019 05:29 AM - Sergey Ivanovskiy

For the swing client we open the target file by external programs which don't send us notifications that the target file is opened successfully. Don't know if it makes sense to copy the target content into a temporary file that will be deleted on JVM exit event. In this case the source file can be deleted immediately after its content has been copied successfully.

#6 - 11/01/2019 05:02 PM - Sergey Ivanovskiy

However, for Linux we can use lsof | grep "absolute file path" in order to check if this file is opened. Another way to check an access attribute using stat. Thus, we can wait until the file access attribute is changed so the file is opened.

#7 - 11/01/2019 05:53 PM - Sergey Ivanovskiy

- File 4077.patch added

Prepared this diff for the commit candidate.

#8 - 11/01/2019 06:01 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> However, for Linux we can use lsof | grep "absolute file path" in order to check if this file is opened. Another way to check an access attribute using stat. Thus, we can wait until the file access attribute is changed so the file is opened.

This whole think is tricky. Both for Swing and Web clients. You cannot assume that once the file is closed it is safe for deletion. The process displaying the mime resource may access the file any time during its execution. Similarly for Web, the resource can be fetched multiple times by the browser - the typical cases are when the browser shows the blocking dialog and only shows the resource when the user confirms it is safe. Or when the resource is displayed and then printed on user request. Or when the user refreshes the browser window showing the mime resource. Also the behavior may vary depending on the browser caching settings.

I think we need to scope the file lifetime with the user session. Once the user session terminates, the resource files served in the terminated session would be deleted. The disadvantage is that we could end up with many pending files, but the delivery would be reliable.

#9 - 11/04/2019 03:28 AM - Sergey Ivanovskiy

For the web client we copy the file and then deliver its copied content. This temporary file is deleted only after its content has been send by DocumentOutputHandler.handle. Thus for the web client if DELETE-SOURCE-CONTENT is present then the target file can be deleted immediately after its content is send.
For the swing client this file is opened by an external program and it can't be deleted immediately unless this file is copied into a temporary file. All these temporary files can be deleted after the user session terminates. In this case we can't delete this target file if DELETE-SOURCE-CONTENT is provided. Is there a way to get rid of coping the target file content into a temporary file?

#10 - 11/04/2019 08:00 AM - Sergey Ivanovskiy

- Status changed from WIP to Review

- % Done changed from 0 to 100

Please review the committed revision 11339 (4077a). Now OPEN-MIME-RESOURCE is defined by

```
OPEN-MIME-RESOURCE <mime-type> <url> [NOT-EMBEDDED] [DELETE-SOURCE].
```

Where:

```
mime-type is the resource MIME type. It can hold any valid character expression.


url is the resource URL (the path to the resource) and it can be any valid character expression.


NOT-EMBEDDED represents that the given resource is not displayed as an embedded document in the web browser, t
he given resource becomes embedded by default if this flag is omitted.


If DELETE-SOURCE is present and the given resource is local file, then it must be deleted after this method re
turns.
    If DELETE-SOURCE is omitted, then the given resource is not deleted.
```

Fixed [NOT-EMBEDDED-CONTENT] in the committed revision 11340 (4077a).
Renamed [NOT-EMBEDDED-CONTENT] to [NOT-EMBEDDED] and [DELETE-SOURCE-CONTENT] to [DELETE-SOURCE] in the committed revision 11342 (4077a).

#11 - 11/04/2019 10:59 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Is there a way to get rid of coping the target file content into a temporary file?

Can we actually move the file? This would be more efficient and wouldn't need additional storage.

#12 - 11/04/2019 10:59 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> NOT-EMBEDED-CONTENT represents that the given resource is not embedded into template document, the given resource becomes embedded by default if this flag is omitted.

The correct spelling should be NOT-EMBEDDED-CONTENT.

#13 - 11/04/2019 11:29 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Is there a way to get rid of coping the target file content into a temporary file?
>
> Can we actually move the file? This would be more efficient and wouldn't need additional storage.

Yes, it is a good idea, planning to use it in the case if DELETE-SOURCE-CONTENT is provided.

#14 - 11/04/2019 01:16 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> The correct spelling should be NOT-EMBEDDED-CONTENT.

Please review the committed revision 11340 (4077a). Fixed this key word, added history changes and used move instead of copy for the swing client.

#15 - 11/04/2019 05:20 PM - Hynek Cihlar

Code review 4077a.

KW_NOT_EMBEDDED should be shortened to 11 chars.

coping => copying

Does it make sense to shorten DELETE-SOURCE-CONTENT to DELETE-SOURCE and NOT-EMBEDDED-CONTENT to NOT-EMBEDDED?

private static final String FILE_SCHEMA should be moved below protected static final Logger LOG.

AGD.tmpDir
I don't think we need to fallback in a ~/fwd dir for a temp location. The Java runtime uses java.io.tmpdir property to read the temp dir location (on Unix systems it is /tmp by default). Just use File.createTempFile(), if needed we can simply set this property to change the temp location.

Doing this throw new MalformedURLException(ex.getMessage()); the call stack and type of the original exception is lost. Maybe openMimeResource should throw URISyntaxException? Or just a RuntimeException similarly as for IOException.

Why the change from urlString to uri.normalize().toASCIIString()? I don't think this is needed. However in TemplateHelper.fill() encoding must be specified to match the template. Ideally a Reader instead of an InputStream should be passed to fill(), since only the caller knows what encoding the passed in template source uses.

GuiWebDriver.FILE_SCHEMA, shouldn't the file scheme be "file"?

GuiWebDriver, again I think we should rely on the java.io.tmpdir for temp dir, respectively just use File.createTempFile

#16 - 11/05/2019 10:48 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> I don't think we need to fallback in a ~/fwd dir for a temp location. The Java runtime uses java.io.tmpdir property to read the temp dir location (on Unix systems it is /tmp by default). Just use File.createTempFile(), if needed we can simply set this property to change the temp location.

I reused the temporary directory that is used already for our code. Don't understand the matter of these proposed changes.

> Doing this throw new MalformedURLException(ex.getMessage()); the call stack and type of the original exception is lost.

I think it doesn't matter to do this change because the call stack follows from this context URL.toURI().

> Why the change from urlString to uri.normalize().toASCIIString()? I don't think this is needed. However in TemplateHelper.fill() encoding must be specified to match the template. Ideally a Reader instead of an InputStream should be passed to fill(), since only the caller knows what encoding the passed in template source uses.

What is incorrect?

> GuiWebDriver.FILE_SCHEMA, shouldn't the file scheme be "file"?

If I understand correctly, you propose to use "file" instead of GuiWebDriver.FILE_SCHEMA.

> GuiWebDriver, again I think we should rely on the java.io.tmpdir for temp dir, respectively just use File.createTempFile

I guess that you mean AbstarctGuiDriver.moveResource(URL url, boolean deleteOnExit) implementation where I used this code because it creates a simple path instead of empty file. If my guess is incorrect, then please express your proposals more thoroughly.

Agree with all other found issues.

#17 - 11/05/2019 11:28 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > I don't think we need to fallback in a ~/fwd dir for a temp location. The Java runtime uses java.io.tmpdir property to read the temp dir location (on Unix systems it is /tmp by default). Just use File.createTempFile(), if needed we can simply set this property to change the temp location.

> I reused the temporary directory that is used already for our code. Don't understand the matter of these proposed changes.

I'm questioning the logic around AbstractGuiDriver.tmpDir, in initDragAndDrop. I don't think it is needed, as it should be covered by File.createTempFile.

But I see now, that you just moved it up from GuiWebDriver.

> Doing this throw new MalformedURLException(ex.getMessage()); the call stack and type of the original exception is lost.

> I think it doesn't matter to do this change because the call stack follows from this context URL.toURI().

While this is a minor issue, I still think it unnecessarily complicates things. When this exception happens you will have to inspect the code to figure out what was the exact cause. Having the exact call stack will help the clarity.

> Why the change from urlString to uri.normalize().toASCIIString()? I don't think this is needed. However in TemplateHelper.fill() encoding must be specified to match the template. Ideally a Reader instead of an InputStream should be passed to fill(), since only the caller knows what encoding the passed in template source uses.

> What is incorrect?

I don't think it is incorrect, I'm just not sure if the change is needed.

> GuiWebDriver.FILE_SCHEMA, shouldn't the file scheme be "file"?

> If I understand correctly, you propose to use "file" instead of GuiWebDriver.FILE_SCHEMA.

The protocol name should not include colon.

> GuiWebDriver, again I think we should rely on the java.io.tmpdir for temp dir, respectively just use File.createTempFile

> I guess that you mean AbstarctGuiDriver.moveResource(URL url, boolean deleteOnExit) implementation where I used this code because it creates a simple path instead of empty file. If my guess is incorrect, then please express your proposals more thoroughly.

This is already covered above in the first point.

#18 - 11/05/2019 11:34 AM - Sergey Ivanovskiy

- File 4077a_up_to_rev11340.patch added

Hynek, it looks like you examined the old code because I did this change in rev 11340 - changed this constant file: to this one "file".

#19 - 11/05/2019 11:56 AM - Sergey Ivanovskiy

This change from urlString to uri.normalize().toASCIIString() is needed because uri can point to the temporary file here (it is your proposed change). It can be that uri.normalize() is not required but it needs to be sure that tmpDir has no "." and ".." in its path

```
      if (embedded)
      {
         try
         {
            uri = buildOpenResourcePage(mimeType, uri.normalize().toASCIIString());
         }
         catch (IOException ex)
         {
            throw new RuntimeException(ex);
         }
      }
```

#20 - 11/05/2019 12:36 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> This change from urlString to uri.normalize().toASCIIString() is needed because uri can point to the temporary file here (it is your proposed change). It can be that uri.normalize() is not required but it needs to be sure that tmpDir has no "." and ".." in its path
> [...]

I don't think that toASCIIString() removes "." and "..". These are valid ASCII chars.

#21 - 11/05/2019 01:55 PM - Sergey Ivanovskiy

Yes, uri.normalize() should do it.

Please review the committed revision 11341 that fixed the rest of the found issues listed in #4077-15.

#22 - 11/05/2019 01:58 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Yes, uri.normalize() should do it.

So toASCIIString() is not needed, correct?

#23 - 11/05/2019 02:02 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Yes, uri.normalize() should do it.
>
> So toASCIIString() is not needed, correct?

I think it is needed because it transforms url to ascii string.

#24 - 11/05/2019 02:03 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Sergey Ivanovskiy wrote:
> >
> > > Yes, uri.normalize() should do it.
> >
> > So toASCIIString() is not needed, correct?
>
> I think it is needed because it transforms url to ascii string.

The template is UTF-8 encoded, it doesn't have to be transformed. My point is that this may potentially complicate debugging.

#25 - 11/05/2019 02:11 PM - Sergey Ivanovskiy

While checking the diff I found missed changes in progress.g due to renaming. So the committed rev 11342 (4077a) looks to be a final candidate.

#26 - 11/07/2019 04:17 AM - Hynek Cihlar

Code review 4077a revisions 11342 and 11341. The changes are good. My only concern is #4077-24, which still remains.

#27 - 11/18/2019 12:12 PM - Greg Shah

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Hynek Cihlar wrote:
> >
> > > Sergey Ivanovskiy wrote:
> > >
> > > > Yes, uri.normalize() should do it.
> > >
> > > So toASCIIString() is not needed, correct?
> >
> > I think it is needed because it transforms url to ascii string.
>
> The template is UTF-8 encoded, it doesn't have to be transformed. My point is that this may potentially complicate debugging.

Sergey: Please respond on this.

#28 - 11/18/2019 12:20 PM - Sergey Ivanovskiy

I didn't respond to this note #4077-24 because it is not an issue and this change doesn't complicate debugging. The input url is not encoded so there is no reason that the debugging will be potentially complicated.

#29 - 11/18/2019 12:36 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I didn't respond to this note #4077-24 because it is not an issue and this change doesn't complicate debugging. The input url is not encoded so there is no reason that the debugging will be potentially complicated.

Sergey, with the call toASCIIString() you change the url passed to the HTML template, all non-ASCII chars are escaped. This is (1) unnecessary as the template is UTF-8 and (2) may complicate debugging - to find the generated URL in the template or in the code you have to remember to encode/decode the url.

#30 - 11/18/2019 12:44 PM - Sergey Ivanovskiy

Hynek, I don't understand why url should not be encoded. Urls can be external and it seems that they can have native chars?

#31 - 11/18/2019 12:52 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek, I don't understand why url should not be encoded. Urls can be external and it seems that they can have native chars?

URLs are encoded in order to overcome charset limitations of the target systems. If you wanted to persist a URL in an ASCII text file for example, then you would have to encode the URL (i.e. use toASCIIString to convert all the non-ASCII chars to ASCII chars). But in your case, this is not needed, you persist the URL in an HTML template which is UTF-8 encoded and so capable to encode all unicode chars.

#32 - 11/18/2019 12:57 PM - Sergey Ivanovskiy

Yes, if these url strings are UTF-8, then we can leave them as they are. Is it possible that urls have non unicode strings?

#33 - 11/18/2019 01:05 PM - Sergey Ivanovskiy

It seems that if we would remove toASCIIString() usage here, then the following code would be incorrect

```
   public void openMimeResource(boolean deleteContent,
                                String  mimeType,
                                String  urlString,
                                boolean embedded)
   throws MalformedURLException,
          URISyntaxException
   {
      // test that urlString is well-formed.
      URL url = new URL(urlString);

      URI uri = url.toURI();

      String id = uri.toASCIIString();
```

```
      boolean local = false;

      if (isLocalResource(url))
      {
         local = true;

         id = resolveResourceId(deleteContent, mimeType, url);
      }

      websock.openMimeResource(id, mimeType, embedded, local);
   }
```

#34 - 11/18/2019 01:12 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> It seems that if we would remove toASCIIString() usage here, then the following code would be incorrect
> [...]

You can pass encoded or un-encoded URI to openMimeResource, both will work fine. However the call to toASCIIString in openMimeResource is needed so that the URL can be passed later on in a URL parameter, see buildOpenResourceUrl in p2j.socjet.js.

#35 - 11/18/2019 01:55 PM - Sergey Ivanovskiy

What is about native no unicode characters?

#36 - 11/18/2019 02:24 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> What is about native no unicode characters?

Sorry, can you rephrase the question?

#37 - 11/18/2019 02:56 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:

> > > What is about native no unicode characters?

> > Sorry, can you rephrase the question?

Can these urls given in no unicode characters be used in UTF-8 template without encoding them into UTF-8?
Examples, http://παραδειγμα.δοκιμη or http://правительство.рф

#38 - 11/18/2019 03:07 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Sergey Ivanovskiy wrote:
> >
> > > What is about native no unicode characters?
> >
> > Sorry, can you rephrase the question?
>
> Can these urls given in no unicode characters be used in UTF-8 template without encoding them into UTF-8?
> Examples, http://παραδειγμα.δοκιμη or http://правительство.рф

Java strings are already unicode (encoded as UTF-16). So the value of the String variable passed to the HTML template can be serialized as UTF-8. So the URLs above will work fine.

#39 - 11/18/2019 03:22 PM - Greg Shah

What about strings with spaces, tabs and control characters?

#40 - 11/18/2019 03:30 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Hynek Cihlar wrote:
> >
> > > Sergey Ivanovskiy wrote:
> > >
> > > > What is about native no unicode characters?
> > >
> > > Sorry, can you rephrase the question?
> >
> > Can these urls given in no unicode characters be used in UTF-8 template without encoding them into UTF-8?
> > Examples, http://παραδειγμα.δοκιμη or http://правительство.рф
>
> Java strings are already unicode (encoded as UTF-16). So the value of the String variable passed to the HTML template can be serialized as UTF-8. So the URLs above will work fine.

It seems that this serialization can be incorrect because it depends on the java default encoding but urls can have different encoding. Does this point make sense?

#41 - 11/18/2019 03:39 PM - Hynek Cihlar

Greg Shah wrote:

> What about strings with spaces, tabs and control characters?

These are illegal for the URI class, you would get URISyntaxException before the ASCII encoding would take place.

It is not big deal to have the toASCIIString modify the URLs. I just don't think it is necessary and I can imagine it could cause minor troubles when tracing the execution.

#42 - 11/18/2019 03:42 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> > Java strings are already unicode (encoded as UTF-16). So the value of the String variable passed to the HTML template can be serialized as UTF-8. So the URLs above will work fine.
>
> It seems that this serialization can be incorrect because it depends on the java default encoding but urls can have different encoding.
>
> Does this point make sense?

Yes and no. In fact the default encoding being used is another issue which can potentially cause troubles. While the HTML template is assumed to be UTF-8, the default encoding could be different from UTF-8. This by itself is not correct and may result in the template being rendered wrong. Instead of default encoding the UTF-8 should be used when serializing the HTML template.

#43 - 11/18/2019 03:46 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > > Java strings are already unicode (encoded as UTF-16). So the value of the String variable passed to the HTML template can be serialized as UTF-8. So the URLs above will work fine.

> It seems that this serialization can be incorrect because it depends on the java default encoding but urls can have different encoding.
>
> Does this point make sense?

> Yes and no. In fact the default encoding being used is another issue which can potentially cause troubles. While the HTML template is assumed to be UTF-8, the default encoding could be different from UTF-8. This by itself is not correct and may result in the template being rendered wrong. Instead of default encoding the UTF-8 should be used when serializing the HTML template.

I think that this point makes sense so the current solution produces a correct result even if the default java encoding is not or is UTF-8. Thus, we can conclude that we should leave it as is now. Any case there is always a possibility to rewrite the code totally.

#44 - 11/18/2019 03:57 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> > Yes and no. In fact the default encoding being used is another issue which can potentially cause troubles. While the HTML template is assumed to be UTF-8, the default encoding could be different from UTF-8. This by itself is not correct and may result in the template being rendered wrong. Instead of default encoding the UTF-8 should be used when serializing the HTML template.
>
> I think that this point makes sense so the current solution produces a correct result even if the default java encoding is not or is UTF-8. Thus, we can conclude that we should leave it as is now. Any case there is always a possibility to rewrite the code totally.

Besides the minor URL encoding issue, the default encoding is more serious. This will break as soon as somebody tries to put a localized title in the HTML template on a system where UTF-8 is not the default encoding, like Windows.

#45 - 11/18/2019 04:39 PM - Sergey Ivanovskiy

Adding a localized title is a separated task and I believe it will be done by correct changes if this task will be needed to implement. I don't understand what are issues we are trying to resolve?

#46 - 11/18/2019 04:42 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Adding a localized title is a separated task and I believe it will be done by correct changes if this task will be needed to implement. I don't understand what are issues we are trying to resolve?

I'm mentioning this because (1) it is a problem and (2) it is very close to the changes you made.

#47 - 11/18/2019 04:58 PM - Sergey Ivanovskiy

It seems that you didn't provide any proofs or evidences that there are issues. Please be more specific to the code because I don't know what requirements will be changed or what code flow will be in the future.

#48 - 11/18/2019 05:13 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> It seems that you didn't provide any proofs or evidences that there are issues. Please be more specific to the code because I don't know what requirements will be changed or what code flow will be in the future.

In SwingGuiDriver.buildOpenResourcePage PrintWriter is constructed with default charset. When the default charset is not UTF-8, any locale specific strings will be encoded incorrectly in TemplateHelper.fill. The solution is to specify UTF-8 encoding in the writer's ctor.

#49 - 11/19/2019 02:57 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > It seems that you didn't provide any proofs or evidences that there are issues. Please be more specific to the code because I don't know what requirements will be changed or what code flow will be in the future.
>
> In SwingGuiDriver.buildOpenResourcePage PrintWriter is constructed with default charset. When the default charset is not UTF-8, any locale specific strings will be encoded incorrectly in TemplateHelper.fill. The solution is to specify UTF-8 encoding in the writer's ctor.

I added comments that "A key provider must return an ASCII string" and rebased 4077a over the current trunc version rev 11339. Please review

these changes rev 11334 (4077a).

#50 - 11/19/2019 03:08 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Sergey Ivanovskiy wrote:
> >
> > > It seems that you didn't provide any proofs or evidences that there are issues. Please be more specific to the code because I don't know what requirements will be changed or what code flow will be in the future.
> >
> > In SwingGuiDriver.buildOpenResourcePage PrintWriter is constructed with default charset. When the default charset is not UTF-8, any locale specific strings will be encoded incorrectly in TemplateHelper.fill. The solution is to specify UTF-8 encoding in the writer's ctor.
>
> I added comments that "A key provider must return an ASCII string" and rebased 4077a over the current trunc version rev 11339. Please review these changes rev 11334 (4077a).

This is not correct. The title is built from a file name, it can be locale specific.

#51 - 11/19/2019 06:39 AM - Sergey Ivanovskiy

Yes, I agree with this point.

#52 - 11/19/2019 09:32 AM - Sergey Ivanovskiy

Checking my changes I fixed open_mime_resource_stmt rule in progress.g and the found #4077-50. I would like to note that html source files are not displayed by Firefox and Chrome because there is no available plugin for this mime type. Please review the committed revision 11345.(4077a)

#53 - 11/19/2019 09:35 AM - Sergey Ivanovskiy

The browser plugin for this mime type is not supported.

```
<embed id="embeddedDocument" type="plain/html" src="https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%B3%D0%B0%D1%80%D0%B8%D0%BD,_%D0%AE%D1%80%D0%B8%D0%B9_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B5%D0%B5%D0%B2%D0%B8%D1%87" width="1855" height="1056">
```

#54 - 11/19/2019 10:14 AM - Sergey Ivanovskiy

To do homogeneous changes we can use this code with the same results

```
=== modified file 'src/com/goldencode/p2j/ui/client/gui/driver/swing/SwingGuiDriver.java'
--- src/com/goldencode/p2j/ui/client/gui/driver/swing/SwingGuiDriver.java    2019-11-19 14:10:34 +0000
+++ src/com/goldencode/p2j/ui/client/gui/driver/swing/SwingGuiDriver.java    2019-11-19 14:56:04 +0000
@@ -811,7 +811,7 @@
       {
          try
          {
-            uri = buildOpenResourcePage(mimeType, uri.normalize().toASCIIString());
+            uri = buildOpenResourcePage(mimeType, uri.toString());
          }
          catch (IOException ex)
          {
@@ -993,7 +993,8 @@
       keysProvider.put("isStreamed",     () -> "false");
       keysProvider.put("documentTitle", () -> StringEscapeUtils.escapeHtml(
                TemplateHelper.native2Ascii(doc.getAbsolutePath())));
-      keysProvider.put("documentPath",   () -> url);
+      keysProvider.put("documentPath",   () -> StringEscapeUtils.escapeHtml(
+               TemplateHelper.native2Ascii(url)));
       keysProvider.put("documentType",   () -> mimeType);

       TemplateHelper.fill(templatePage, keysProvider, writer);
```

#55 - 11/19/2019 03:57 PM - Hynek Cihlar

Code review 4077a revision 11345.

With

```
keysProvider.put("documentTitle", () -> StringEscapeUtils.escapeHtml(
               TemplateHelper.native2Ascii(doc.getAbsolutePath())));
```

the end user will be presented with raw bytes in the place of locale specific characters. I don't think this is what we want.

Please reread #4077-48. The solution is really trivial, undo the code above and change PrintWriter writer = new PrintWriter(doc); to PrintWriter writer = new PrintWriter(doc, "UTF-8"); in SwingGuiDriver.buildOpenResourcePage.

#56 - 11/20/2019 01:58 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 4077a revision 11345.
>
> With
> [...]
>
> the end user will be presented with raw bytes in the place of locale specific characters. I don't think this is what we want.
>
> Please reread #4077-48. The solution is really trivial, undo the code above and change PrintWriter writer = new PrintWriter(doc); to PrintWriter writer = new PrintWriter(doc, "UTF-8"); in SwingGuiDriver.buildOpenResourcePage.

Yes, it is my mistake that the title string is given by \\u entities, but it needs to be represented by html entities

```
https://example.org/test.pdf?q=&#x043c;&#x043e;&#x0439;%20&#x0434;&#x043e;&#x043c%20;&#x042d;&#x0444;&#x0438;&#x043e;&#x043f;&#x0438;%20&#x0410;&#x0444;&#x0440;&#x0438;&#x043a;&#x0430;&amp;q=q'werty
```

It can be easily fixed.

#57 - 11/20/2019 02:05 AM - Sergey Ivanovskiy

Testing several documents opened sequentially by 4GL business code I detected race conditions with the Swing client. It is a worse issue.

#58 - 11/20/2019 03:09 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Testing several documents opened sequentially by 4GL business code I detected race conditions with the Swing client. It is a worse issue.

What is the use case?

#59 - 11/20/2019 03:17 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

but it needs to be represented by html entities

The HTML page is encoded in UTF-8, so any unicode char will work OK. The only chars that need to be escaped are &, <, > and ", ' for attribute values. Make sure you use UTF-8 encoding for the PrintWriter instance and only use StringEscapeUtils.escapeHtml.

#60 - 11/20/2019 04:18 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Testing several documents opened sequentially by 4GL business code I detected race conditions with the Swing client. It is a worse issue.
>
> What is the use case?

Actually, it is not a truly race condition. It happens due to the same file name was generated for different resource files.

```
-       File doc = new File(getPrintOutputDir(), getPrintOutputFile(null, MediaType.HTML));
+       File doc = new File(getPrintOutputDir(),
+                           getPrintOutputFile(UUID.randomUUID().toString(),
+                           MediaType.HTML));
```

#61 - 11/20/2019 04:24 AM - Sergey Ivanovskiy

I wouldn't want to use PrintWriter writer = new PrintWriter(doc, "UTF-8"); because of the following reasons. Source strings can be from non UTF-8 and the default java encoding can be different. It seems that we shouldn't code "UTF-8" inside the code. I understand that html template specifies its encoding as UTF-8 but ASCII is a subset of UTF-8.

#62 - 11/20/2019 04:49 AM - Sergey Ivanovskiy

Please review the committed revision 11346. If this solution will not be approved, then PrintWriter writer = new PrintWriter(doc, "UTF-8"); will be applied instead.

#63 - 11/20/2019 05:06 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I wouldn't want to use PrintWriter writer = new PrintWriter(doc, "UTF-8"); because of the following reasons. Source strings can be from non UTF-8 and the default java encoding can be different.

The source Java string is already in unicode (in UTF-16, but this is not that important) so can be represented in UTF-8. Both UTF-16 and UTF-8 are unicodes but with different encodings. These can be interchanged with proper encoders (i.e. when you set UTF-8 in the Writer instance).

We must not use the default encoding as this gives unexpected results, depending what the OS default encoding for Java is set to.

> It seems that we shouldn't code "UTF-8" inside the code.

We don't code in UTF-8. Here the UTF-8 encoding is what the HTML template is set to and hence must serialize UTF-8 strings in it.

#64 - 11/20/2019 05:18 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > I wouldn't want to use PrintWriter writer = new PrintWriter(doc, "UTF-8"); because of the following reasons. Source strings can be from non UTF-8 and the default java encoding can be different.
>
> The source Java string is already in unicode (in UTF-16, but this is not that important) so can be represented in UTF-8. Both UTF-16 and UTF-8 are unicodes but with different encodings. These can be interchanged with proper encoders (i.e. when you set UTF-8 in the Writer instance).
>
> We must not use the default encoding as this gives unexpected results, depending what the OS default encoding for Java is set to.

I think that the presented rev 11346 should give expected results. I will use your proposed solution, because it should work if the default java encoding is UTF-8.

#65 - 11/20/2019 05:36 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I think that the presented rev 11346 should give expected results.

Even with 11346 the encoding issue would not be fully resolved. Consider the HTML template localized, in Czech for example. It would get deserialized, template values filled and serialized back. If the serialization was performed in encoding other than UTF-8 (and since the template specifies UTF-8) the result would be garbage for the Czech characters.

You will also have to specify UTF-8 encoding when reading the template. Change BufferedReader reader = new BufferedReader(new InputStreamReader(template)); to BufferedReader reader = new BufferedReader(new InputStreamReader(template, "UTF-8")); in TemplateHelper.fill.

#66 - 11/20/2019 05:48 AM - Sergey Ivanovskiy

Yes, thank you, I will change this too. For localized resources native to ascii conversion is usually used and several templates share the same name with different language suffixes.

#67 - 11/20/2019 06:58 AM - Sergey Ivanovskiy

Please review the committed revision 11347 (4077a).

#68 - 11/20/2019 07:35 AM - Hynek Cihlar

Code review 4077a revision 11347. The changes are good. I have no more concerns for 4077a.

#69 - 11/20/2019 09:03 AM - Greg Shah

Code Review Task Branch 4077a Revision 11347

I'm good with the changes, except some minor things:

1. In progress.g, there is no need for the additional usage of attrsAndMethods.put(). These are not attributes of a handle based resource so this extra code should be removed.

2. In ScreenDriver, please use import java.net.* instead of the explicit exceptions.

Please make these changes and then merge this into 3809e.

#70 - 11/20/2019 10:58 AM - Sergey Ivanovskiy

Committed these fixes in rev 11348 (4077a) and then merged 4077a into 3809e as rev 11390.

#71 - 01/06/2020 08:30 AM - Greg Shah

- Status changed from Review to Test

Sergey: Please update the documentation to fully explain the new syntax. You MUST leave the old syntax there as well, since it is incompatible but is available for a specific range of FWD revisions. Please make it clear which revisions use the old syntax and which ones use the new syntax. The new syntax is not yet in trunk so put a placeholder there for now ("3809e revision ?" where ? is the revision in which you added the new syntax). When 3809e merges to trunk we can change this placeholder to be the trunk revision.

#72 - 01/06/2020 09:22 AM - Sergey Ivanovskiy

Greg, please review these changes https://proj.goldencode.com/projects/p2j/wiki/OPEN-MIME-RESOURCE#section-3.

#73 - 01/06/2020 10:57 AM - Greg Shah

I made some additional edits. Also, please note that the FWD implementation details are now also out of date.

#74 - 01/21/2020 09:24 AM - Greg Shah

3809e was merged to trunk as revision 11340.

Sergey: Please update the FWD implementation details in the OPEN-MIME-TYPE documentation so that it is complete. Remember that we can't delete the old documentation, we just have to make it clear that as of revision 11340 there is a different implementation. Please read my edits to the docs so that you have an idea how to format this.

Once the documentation is complete I will close this task.

#75 - 01/31/2020 01:33 PM - Greg Shah

- Status changed from Test to Closed

Files

4077.patch	18.4 KB	11/01/2019	Sergey Ivanovskiy
4077a_up_to_rev11340.patch	36.8 KB	11/05/2019	Sergey Ivanovskiy