User Interface - Feature #3474

OPEN-MIME-RESOURCE

02/10/2018 10:35 AM - 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 #4077: add option to OPEN-MIME-RESOURCE t...	**Closed**

History

#2 - 02/10/2018 10:40 AM - Greg Shah

Implement a variant of P2J-OPEN-URL (see #1815) called OPEN-MIME-RESOURCE. The idea is that where P2J-OPEN-URL is meant to allow the user's browser to load the URL directly, the OPEN-MIME-RESOURCE is meant to load a mime resource (text file, PDF, CSV, XLS, even html... that exists on the FWD client) down into the user's browser. If that resource can be rendered by the browser, we would want it to load in its own tab. As a fallback, any resource that cannot be rendered by the browser should prompt the user for download/open via helper app.

Please see #1795-61 (and later) for more details on the features needed. This should be used for both OUTPUT TO PRINTER as well as for the enhanced browse export feature.

#3 - 03/28/2018 12:16 AM - Eric Faulhaber

- Assignee set to Sergey Ivanovskiy

#4 - 03/29/2018 10:10 AM - Sergey Ivanovskiy

Created 3474a for this task.

#5 - 03/30/2018 09:05 AM - Sergey Ivanovskiy

Please clarify that OPEN-MIME-RESOURCE should be used in OUTPUT TO PRINTER statement as well as for the enhanced browse export feature. What does it mean?

This js framework http://mozilla.github.io/pdf.js/getting_started/ can be used by Swing and Web clients both to render pdf documents in browsers.

Also this java library http://cssbox.sourceforge.net/pdf2dom/ can be evaluated. It transforms pdf documents into html documents.

#6 - 03/30/2018 09:25 AM - Greg Shah

There are many ways for converted 4GL code to generate some kind of output. Once that output is created, it needs to be provided to the end user in some way. In the original 4GL system, the output was already on the end user's desktop system. File resources were on the hard drive, other things might be directly output to printers, email or devices (including the screen).

In the web client the user is assumed to be running the browser on a different system than where the FWD client is executing. Although the local web client is possible, we cannot assume it is in use. Since the user has no direct access to the output, we must provide a facility for the output to be opened via the browser so that the user can view, interact, save and print any such output.

The idea of OPEN-MIME-RESOURCE is to provide a tool to do this which can be used in 4GL converted code (but only in FWD since there is no OpenEdge version). This is an enhancement of the 4GL syntax itself.

OUTPUT TO PRINTER sends the PDF down to the browser. It appears as a "download or open with" prompt. Just like with any other output that is generated by the FWD client, the PDF needs to get down to the user's browser BUT we don't want it to always prompt for the user to open or download it. We really want it to be loaded as a tab in the user's browser with no other interaction needed. This is the same thing we are already doing for OPEN-MIME-RESOURCE, so I don't want to duplicate work. The same solution for OPEN-MIME-RESOURCE should be used "under the covers" in OUTPUT TO PRINTER.

> This js framework http://mozilla.github.io/pdf.js/getting_started/ can be used by Swing and Web clients both to render pdf documents in browsers.

The license is Apache 2, so it is OK. I'm open to this idea. Can you please describe how you envision this to be implemented?

Does this only work for PDF mime types? We must support a range of mime types (text, xls, pdf...). I was thinking we would use the browser's built-in support for mime types rather than build our own custom viewer for each one.

On the other hand, PDF will be a very common one and could be worth a custom implementation if we get a really good result (as compared with using the browser directly).

> Also this java library http://cssbox.sourceforge.net/pdf2dom/ can be evaluated. It transforms pdf documents into html documents.

No, this won't work for our needs. The user must have the original file and be provided to the user so they can save it or interact with it using some native tool (e.g. xls file in Excel as the browser's application helper).

#7 - 03/30/2018 10:10 AM - Sergey Ivanovskiy

Found that pdf.js supports only these browsers https://github.com/mozilla/pdf.js/wiki/Frequently-Asked-Questions#what-browsers-are-supported It is reported that Safari and IE11/Edge have limited supports(?).
If it is fitted our case, then we can create the dedicated mime type servlet handler that sends html output rendered in a web page. Although the viewer web page design can be the issue, pdf.js provides this example for evaluations
http://jsfiddle.net/pdfjs/wagvs9Lf/?utm_source=website&utm_medium=embed&utm_campaign=wagvs9Lf

#8 - 03/30/2018 10:43 AM - Sergey Ivanovskiy

It seems that a servlet that implements OPEN-MIME-RESOURCE should only send a document with its correct mime type, then the target browser can display the target document if its document type is supported by this browser.

#9 - 03/30/2018 11:05 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> It seems that a servlet that implements OPEN-MIME-RESOURCE should only send a document with its correct mime type, then the target browser can display the target document if its document type is supported by this browser.

Note that using the embed HTML 5 tag can increase the chance for the browser to open the document in a new window/tab. This worked OK for Chrome, even when it was configured not to open pdf files in a tab.

#10 - 03/30/2018 01:34 PM - Sergey Ivanovskiy

OK. I will try embed. What are operands of this OPEN-MIME-RESOURCE? Is it the same as for P2J-OPEN-URL?

#11 - 03/30/2018 01:40 PM - Greg Shah

> What are operands of this OPEN-MIME-RESOURCE? Is it the same as for P2J-OPEN-URL?

It is not the same.

1. MIME type as a string/character value (e.g. "text/plain").
2. Some way to reference the resource, usually a filename for the FWD client system is one example. I'm not sure what is needed for usage from OUTPUT TO PRINTER.

#12 - 03/30/2018 03:03 PM - Sergey Ivanovskiy

Thank you. I will follow this design.

#13 - 04/02/2018 12:02 AM - Eric Faulhaber

- Status changed from New to WIP

#14 - 04/02/2018 01:51 AM - Sergey Ivanovskiy

Working on OPEN-MIME-RESOURCE. According to P2J-OPEN-URL

```
/**
 * Matches the P2J-extension language statement P2J-OPEN-URL and a mandatory
 * following character expression.
```

```
 */
p2j_open_url_stmt
   :
      KW_OPEN_URL^ expr DOT!
   ;
```

the target statement requires to define a pair of strings or to define 2-tuple in order to state something like

```
open_mime_rsrc_stmt
   :
      KW_OPEN_MIME_RESOURCE^ pair DOT!
   ;
```

#15 - 04/02/2018 02:01 AM - Sergey Ivanovskiy

Actually, there is a helpful example how to state 2 or 3 parameters

```
/**
 * Matches the COLLATE keyword and the required following
 * 2 or 3 character expressions.  The tree is rooted on the
 * COLLATE keyword and all parenthesis and commas are
 * dropped from the tree.  The resulting tree will have 2 or 3 children,
 * each an expression.
 * <p>
 * Used by {@link #preselect_phrase} and the subtree is rooted by the
 * keyword (which is the reason for using a separate rule for something so
 * simple).
 */
collate_clause
   :
      KW_COLLATE^ LPARENS! expr COMMA! expr (COMMA! expr)? RPARENS!
      (KW_DESCEND)?
   ;
```

#16 - 04/02/2018 05:29 AM - Sergey Ivanovskiy

Finally, this construction works properly

```
/**
 * Matches the FWD-extension language statement OPEN-MIME-RESOURCE and a mandatory
 * pair of resource mime type character and its url character expression.
 */
open_mime_resource_stmt
   :
      KW_OPEN_MIME_RESOURCE^ expr expr DOT!
   ;
```

#17 - 04/02/2018 10:47 AM - Sergey Ivanovskiy

Working on OpenMimeResourceHandler that should handle print requests from js web clients by loading html document with the embedded content to the target resource, then PDFPrintOutputHandler loads this document to the browser.
It seems that if we consider the case when java web clients are under proxy server, then the following code shouldn't work

```
         case types.MSG_OPEN_PRINT_OUTPUT:
            var offset = 1;
            var textLength = message[offset];
            offset = offset + 1;

            var id  = me.readStringBinaryMessageByLength(message, offset, textLength);
            offset = offset + textLength * 2;

            window.open("/print/" + id, "_blank");
            break;
```

because JS clients are out of the proxy server network. According to our web client design we have webRoot path as a root for all available web resources.

#18 - 04/02/2018 11:42 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> According to our web client design we have webRoot path as a root for all available web resources.

Btw. why don't we reference the resources relative to the index file?

So instead of
<script type="text/javascript" src="${webRoot}/common/p2j.js"></script>
having this
<script type="text/javascript" src="common/p2j.js"></script>.

#19 - 04/02/2018 11:58 AM - Sergey Ivanovskiy

Yes, it seems we can use relative path if there are no missed design features.
It seems that web socket requires absolute path in this template index.html

```
'url' : 'wss://' + window.location.host + '${webRoot}/ajax',
```

#20 - 04/02/2018 04:16 PM - Sergey Ivanovskiy

I tested PDFPrintOutputHandler and encountered this exception

```
WARNING: //localhost:7449/print/7d9b2414-a5a5-4b19-a0ca-e7b5eaa9311c
java.lang.RuntimeException: org.eclipse.jetty.io.EofException
    at com.goldencode.p2j.util.GUIPrinterStreamSupport.lambda$closeStream$1(GUIPrinterStreamSupport.java:351)
    at com.goldencode.p2j.ui.client.gui.driver.web.PDFPrintOutputHandler.handle(PDFPrintOutputHandler.java:149
)
    at org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:1162)
    at org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:1096)
    at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:141)
    at org.eclipse.jetty.server.handler.HandlerCollection.handle(HandlerCollection.java:119)
    at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:134)
    at org.eclipse.jetty.server.Server.handle(Server.java:518)
    at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:308)
    at org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:244)
    at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:273)
    at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:95)
    at org.eclipse.jetty.io.ssl.SslConnection.onFillable(SslConnection.java:186)
    at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:273)
    at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:95)
    at org.eclipse.jetty.io.SelectChannelEndPoint$2.run(SelectChannelEndPoint.java:93)
    at org.eclipse.jetty.util.thread.strategy.ExecuteProduceConsume.produceAndRun(ExecuteProduceConsume.java:2
46)
    at org.eclipse.jetty.util.thread.strategy.ExecuteProduceConsume.run(ExecuteProduceConsume.java:156)
    at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:654)
    at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java:572)
    at java.lang.Thread.run(Thread.java:748)
Caused by: org.eclipse.jetty.io.EofException
    at org.eclipse.jetty.io.ChannelEndPoint.flush(ChannelEndPoint.java:197)
    at org.eclipse.jetty.io.ssl.SslConnection$DecryptedEndPoint.flush(SslConnection.java:814)
    at org.eclipse.jetty.io.WriteFlusher.flush(WriteFlusher.java:419)
    at org.eclipse.jetty.io.WriteFlusher.write(WriteFlusher.java:313)
    at org.eclipse.jetty.io.AbstractEndPoint.write(AbstractEndPoint.java:141)
    at org.eclipse.jetty.server.HttpConnection$SendCallback.process(HttpConnection.java:724)
    at org.eclipse.jetty.util.IteratingCallback.processing(IteratingCallback.java:241)
    at org.eclipse.jetty.util.IteratingCallback.iterate(IteratingCallback.java:224)
    at org.eclipse.jetty.server.HttpConnection.send(HttpConnection.java:512)
    at org.eclipse.jetty.server.HttpChannel.sendResponse(HttpChannel.java:668)
    at org.eclipse.jetty.server.HttpChannel.write(HttpChannel.java:722)
    at org.eclipse.jetty.server.HttpOutput.write(HttpOutput.java:179)
    at org.eclipse.jetty.server.HttpOutput.write(HttpOutput.java:163)
    at org.eclipse.jetty.server.HttpOutput.flush(HttpOutput.java:299)
    at java.io.FilterOutputStream.flush(FilterOutputStream.java:140)
    at java.io.FilterOutputStream.close(FilterOutputStream.java:158)
    at org.apache.pdfbox.pdfwriter.COSWriter.close(COSWriter.java:311)
    at org.apache.pdfbox.pdmodel.PDDocument.save(PDDocument.java:1254)
    at com.goldencode.p2j.util.GUIPrinterStreamSupport.lambda$closeStream$1(GUIPrinterStreamSupport.java:346)
    ... 20 more
Caused by: java.io.IOException: Broken pipe
```

```
    at sun.nio.ch.FileDispatcherImpl.write0(Native Method)
    at sun.nio.ch.SocketDispatcher.write(SocketDispatcher.java:47)
    at sun.nio.ch.IOUtil.writeFromNativeBuffer(IOUtil.java:93)
    at sun.nio.ch.IOUtil.write(IOUtil.java:51)
    at sun.nio.ch.SocketChannelImpl.write(SocketChannelImpl.java:471)
    at org.eclipse.jetty.io.ChannelEndPoint.flush(ChannelEndPoint.java:175)
    ... 38 more
```

when this program is executed

```
def var h1 as handle.

create window h1 assign title = "h1".

def var path as character format "x(256)" label "URL".

def button go1 label "Go".

form path skip go1 with frame f1 title "fh1".

enable all with frame f1 in window h1.

on "choose" of go1 in frame f1 do:
  /* assign path.
   message "Go " + path in window h1.
   p2j-open-url path.
   open-mime-resource string("text/plain") path.*/

   def var i as int.

   message "Starting.".

   output to printer page-size 10.

   repeat i = 1 to 1000:
     display "this is a test line" i.
   end.

   output close.

   message "Finished.".
end.

view h1.

wait-for close of current-window.
```

#21 - 04/02/2018 04:17 PM - Sergey Ivanovskiy

Committed revision 11245.(3474a).

#22 - 04/03/2018 03:59 AM - Sergey Ivanovskiy

It seems the idea applied in PDFPrintOutputHandler to use direct output on document.save(output) should not work because it should close output stream before the actual data was written into https channel.

#23 - 04/03/2018 04:10 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> It seems the idea applied in PDFPrintOutputHandler to use direct output on document.save(output) should not work because it should close output stream before the actual data was written into https channel.

Which output stream do you mean to be closed?

Btw. I think there is a bug in GUIPrinterStreamSupport.closeStream(). The document should be only closed from the finalizer, and not from the consumer.

#24 - 04/03/2018 04:28 AM - Sergey Ivanovskiy

I mean https output stream. For an example, this code works properly

```
        ByteArrayOutputStream bout = new ByteArrayOutputStream();

        output.outputConsumer.accept(bout);

        ServletOutputStream out = httpServletResponse.getOutputStream();

        out.write(bout.toByteArray());
        out.flush();
```

The next finding is that embed tag in my environment doesn't help to display the target pdf. I can watch its content if looking at the source document. view-source:https://localhost:7449/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06. I think we should send pdf output without wrapping it in html template.

```
<html>
<head>
<title></title>
</head>
<body>
<embed src="/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06" type="application/pdf"/>
</body>
</html>
```

#25 - 04/03/2018 04:40 AM - Sergey Ivanovskiy

In my test the requested wrapped document has this url https://localhost:7449/open/resource/?path=8fd81ccd-2f4c-4668-8097-c79a3b510a06&mimeType=application/pdf and the target document has this https://localhost:7449/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06. I got this document

```
<html>
<head>
<title></title>
</head>
<body>
<embed src="/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06" type="application/pdf"/>
</body>
</html>
```

and got /print/8fd81ccd-2f4c-4668-8097-c79a3b510a06 but Firefox didn't show its look and feel.

#26 - 04/03/2018 04:49 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I mean https output stream. For an example, this code works properly
> [...]

The problem is that PdfBox closes the provided stream when PDDocument.save() is called. Instead of creating new memory-backed stream, just create a simple stream wrapper that will provide a no-op override for its close() method and use it in GUIPrinterStreamSupport.closeStream(). GuiDriver.deliverPrintOutput() states "The consumer implementation must not close the provided stream.", so this change should satisfy the requirement.

#27 - 04/03/2018 04:52 AM - Sergey Ivanovskiy

Yes, agree. About embed tag I can test this http://mozilla.github.io/pdf.js/getting_started/ with our own view pdf template open-resource.html if there are no objections.

#28 - 04/03/2018 04:54 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I mean https output stream. For an example, this code works properly
> [...]
> The next finding is that embed tag in my environment doesn't help to display the target pdf. I can watch its content if looking at the source document. view-source:https://localhost:7449/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06. I think we should send pdf output without wrapping

> it in html template.
> [...]

Check the network monitor in the browser on the pdf resource (/print/...), what HTTP status code do you get in the response? Also do you see any errors in PDFPrintOutputHandler?

#29 - 04/03/2018 05:33 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > I mean https output stream. For an example, this code works properly
> > [...]
> > The next finding is that embed tag in my environment doesn't help to display the target pdf. I can watch its content if looking at the source document. view-source:https://localhost:7449/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06. I think we should send pdf output without wrapping it in html template.
> > [...]
>
> Check the network monitor in the browser on the pdf resource (/print/...), what HTTP status code do you get in the response? Also do you see any errors in PDFPrintOutputHandler?

There are no exceptions and pdf document is on the client but it isn't displayed by Firefox. May be it is a template issue (open-resource.html). Please review committed revision 11246 with PDFPrintOutputHandler changes.

#30 - 04/03/2018 06:10 AM - Sergey Ivanovskiy

Another questions follow. We have ReportOutputFormat and GuiDriver.this.deliverPrintOutput(Consumer<OutputStream> consumer, Runnable finalizer, ReportOutputFormat format) and reports of this type ReportOutputFormat.PDF are handled by PDFPrintOutputHandler. Are there handlers for CSV, XLS, XLSX reports? How to test these document types?

#31 - 04/03/2018 06:20 AM - Hynek Cihlar

- File test.html added

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Sergey Ivanovskiy wrote:
> >
> > > I mean https output stream. For an example, this code works properly
> > > [...]
> > > The next finding is that embed tag in my environment doesn't help to display the target pdf. I can watch its content if looking at the source document. view-source:https://localhost:7449/print/8fd81ccd-2f4c-4668-8097-c79a3b510a06. I think we should send pdf output without wrapping it in html template.
> > > [...]
> >
> > Check the network monitor in the browser on the pdf resource (/print/...), what HTTP status code do you get in the response? Also do you see any errors in PDFPrintOutputHandler?
>
> There are no exceptions and pdf document is on the client but it isn't displayed by Firefox. May be it is a template issue (open-resource.html). Please review committed revision 11246 with PDFPrintOutputHandler changes.

Open the attached html file, does it show up correctly? It does for me.

#32 - 04/03/2018 06:27 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Another questions follow. We have ReportOutputFormat and GuiDriver.this.deliverPrintOutput(Consumer<OutputStream> consumer, Runnable finalizer, ReportOutputFormat format) and reports of this type ReportOutputFormat.PDF are handled by PDFPrintOutputHandler. Are there handlers for CSV, XLS, XLSX reports? How to test these document types?

ReportOutputFormat, GuiDriver.deliverPrintOutput() and related were created to support delivery of print outputs to clients. They were created before I was aware of OPEN-MIME-RESOURCE.

But I think these can be generalized easily. Rename deliverPrintOutput to something like deliverMediaResource, PDFPrintOutputHandler to MediaResourceHandler and so on. Also ReportOutputFormat is probably redundant, it can be replaced with com.google.common.net.MediaType.

#33 - 04/03/2018 07:26 AM - Sergey Ivanovskiy

Yes, it works for me too. The problem was "Content-disposition" value: attachment or inline. If attachment is specified in header "Content-disposition", then nothing is displayed and the content isn't downloaded.

#34 - 04/03/2018 08:32 AM - Sergey Ivanovskiy

It seems that file resources available from Swing client should be available via web clients. Thus, FileResourceHandler should be implemented. If it is required, then please help with FWD API for file resource searching.

#35 - 04/03/2018 08:55 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> It seems that file resources available from Swing client should be available via web clients. Thus, FileResourceHandler should be implemented. If it is required, then please help with FWD API for file resource searching.

I am not aware of any special requirements for retrieving the file resources. Can you just simply open a file stream and supply it to GuiDriver.deliverPrintOutput() (or the renamed equivalent)?

#36 - 04/03/2018 10:58 AM - Sergey Ivanovskiy

Committed revision 11247 fixed issues with pdf and added new MSG_OPEN_MIME_RESOURCE. Working to implement file resource handler.

#37 - 04/03/2018 10:59 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > It seems that file resources available from Swing client should be available via web clients. Thus, FileResourceHandler should be implemented. If it is required, then please help with FWD API for file resource searching.
>
> I am not aware of any special requirements for retrieving the file resources. Can you just simply open a file stream and supply it to GuiDriver.deliverPrintOutput() (or the renamed equivalent)?

Yes, I will try this way.

#38 - 04/03/2018 02:01 PM - Sergey Ivanovskiy

Please review committed revision 11248. Now this construction is supported by web clients.

```
open-mime-resource string("application/pdf") string("file:/home/sbi/Documents/groups.pdf").
```

I didn't implement it for Swing clients.

#39 - 04/03/2018 02:57 PM - Greg Shah

> I didn't implement it for Swing clients.

It would be useful to launch a browser (we already support that) and then load the same way as in the web client.

Otherwise we have no solution for the Swing case, in printing or anything else like this.

#40 - 04/04/2018 09:05 AM - Sergey Ivanovskiy

It needs to reuse the code that is responsible to fill html templates HtmlResourceHandler. Otherwise these functionality will be duplicated.

#41 - 04/05/2018 01:32 AM - Sergey Ivanovskiy

The implemented method to wrap documents with help of open-resource.html template (embed) doesn't work for binary content types represented by application/octet-stream. It seems that in this case it is needed to request the target document directly from its handler.
Committed rev 11251 has been updated over trunc rev 11244.

#42 - 04/05/2018 02:23 AM - Constantin Asofiei

Sergey Ivanovskiy wrote:

> Finally, this construction works properly
> [...]

Does this allow you to have something like open-mine-resource "text/plain" "a.txt". or is it always needed to have string("text/plain")?

#43 - 04/05/2018 02:43 AM - Sergey Ivanovskiy

I checked that plain string parameters can be used too. Committed rev 11252 added missed java docs.

#44 - 04/05/2018 02:45 AM - Sergey Ivanovskiy

It seems that now url string should be with its protocol, for an example "file:./a.txt".

#45 - 04/05/2018 04:43 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> The implemented method to wrap documents with help of open-resource.html template (embed) doesn't work for binary content types represented by application/octet-stream. It seems that in this case it is needed to request the target document directly from its handler. Committed rev 11251 has been updated over trunc rev 11244.

Embedding the media (using embed tag for example) relies on the browser (or an installed plugin) being capable to handle it. Some media types can be embedded, some not. We should probably make this configurable. Greg, what do you think?

#46 - 04/05/2018 07:18 AM - Greg Shah

> We should probably make this configurable. Greg, what do you think?

Yes, this makes sense. We can embed by default and allow an optional boolean 3rd parameter (true means embed, false means don't embed).

#47 - 04/05/2018 09:58 AM - Sergey Ivanovskiy

OK, planning to add this optional parameter. Committed revision 11254 changed SwingGuiDriver.this.deliverPrintOutput to open printed documents via SwingGuiDriver.this.openURL. I found that SwingGuiDriver.this.deliverPrintOutput is possibly changed in the future

```
      // this implementation is not expected to be deployed in production due to possible file name collisions
      // in multi-user environments and unresolved security concerns

      UnimplementedFeature.experimental("Swing GUI PDF print output is an experimental feature not meant " +
                                        "to be used in production environments.");

      try
      {
         File outFile = new File(getPrintOutputDir(), getPrintOutputFile(format));
         try(OutputStream os = new FileOutputStream(outFile))
         {
            consumer.accept(os);
         }

         openURL(outFile.toURI().normalize().toASCIIString());
.........................
```

#48 - 04/06/2018 01:19 AM - Sergey Ivanovskiy

Greg, please review committed revision 11255.

#49 - 04/06/2018 01:38 AM - Sergey Ivanovskiy

- Status changed from WIP to Review

#50 - 04/06/2018 04:37 AM - Hynek Cihlar

Currently all the url targets are first copied to the machine that runs the FWD client and then streamed to the end-user's browser. This holds true even for urls that are not local to the FWD client system. Do we really want this? Or do we want to directly open the global urls in the user's browser (i.e. http://example.com/example.pdf)?

#51 - 04/06/2018 05:11 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Currently all the url targets are first copied to the machine that runs the FWD client and then streamed to the end-user's browser. This holds true even for urls that are not local to the FWD client system. Do we really want this? Or do we want to directly open the global urls in the user's browser (i.e. http://example.com/example.pdf)?

No, only ids are copied.

#52 - 04/06/2018 05:55 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Currently all the url targets are first copied to the machine that runs the FWD client and then streamed to the end-user's browser. This holds true even for urls that are not local to the FWD client system. Do we really want this? Or do we want to directly open the global urls in the user's browser (i.e. http://example.com/example.pdf)?
>
> No, only ids are copied.

In GuiWebDriver.openMimeResource() method body:

```
public void openMimeResource(String mimeType, String path, boolean embedded)
throws MalformedURLException
{
   final URL url = new URL(path);
   Consumer<OutputStream> consumer = output ->
   {
      try(InputStream input = url.openStream())
      {
         ByteStreams.copy(input, output);
         ...
```

When you supply an absolute URL with a domain, the openStream() and the ByteStreams.copy() calls will effectively fetch the target resource to FWD client.

#53 - 04/06/2018 06:04 AM - Hynek Cihlar

Hynek Cihlar wrote:

> When you supply an absolute URL with a domain, the openStream() and the ByteStreams.copy() calls will effectively fetch the target resource to FWD client.

To clarify, I'm not saying this is necessarily wrong, I just want to point this out whether we want this behavior.

#54 - 04/06/2018 06:42 AM - Sergey Ivanovskiy

> Hynek Cihlar wrote:
>
> > When you supply an absolute URL with a domain, the openStream() and the ByteStreams.copy() calls will effectively fetch the target resource to FWD client.
>
> To clarify, I'm not saying this is necessarily wrong, I just want to point this out whether we want this behavior.

Now I understand your question. Yes, now it is possible to load documents by their URLs. it depends on requirements for OPEN-MIME-RESOURCE.

#55 - 04/06/2018 07:06 AM - Greg Shah

> Yes, now it is possible to load documents by their URLs. it depends on requirements for OPEN-MIME-RESOURCE.

I should have specified this.

Only resources that are already on the FWD systems should be streamed. Any directly accessible URLS should be sent only as a URL, without copying to the FWD client.

#56 - 04/06/2018 07:12 AM - Greg Shah

Please remove this comment and code:

```
      // this implementation is not expected to be deployed in production due to possible file name collisions
      // in multi-user environments and unresolved security concerns

      UnimplementedFeature.experimental("Swing GUI PDF print output is an experimental feature not meant " +
                                        "to be used in production environments.");
```

At this time, I don't think we need to treat this as experimental.

The same issues exist for both Swing and Web clients. Actually, the Swing client is more likely to be run on a single client system, reducing the possible security issues. The primary problem is that we do no checking of the URL and this certainly exposes the user to some level of danger depending on how the URL is obtained.

Regardless, I don't want to suggest in the code that the feature doesn't work or cannot be relied upon.

#57 - 04/06/2018 07:52 AM - Sergey Ivanovskiy

> Only resources that are already on the FWD systems should be streamed. Any directly accessible URLS should be sent only as a URL, without copying to the FWD client.

Greg, please clarify this case. There are server's resources and client's resources. Should P2j-OPEN-URL url. satisfy the same requirement that resources are loaded only if they are on the FWD system?

#58 - 04/06/2018 07:57 AM - Greg Shah

> There are server's resources and client's resources.

As far as I know, at this point all the resources we need to **stream** are already on the FWD client. Those resources would be accessed via a file:// URL.

Does anyone know of a case where we need to load from the FWD server?

Any URL that addresses a resource located outside of the FWD client, should still be sent to the client but it will not be streamed from the FWD client. Instead, the browser can load that resource directly from the URL.

#59 - 04/06/2018 08:30 AM - Sergey Ivanovskiy

We don't specify the case if this command is failed. What should be done in this case? Now if resources are failed to load or specified urls are malformed, then exceptions on the client side are thrown. If resources are external, then only urls will be sent to the client side. Does it need to have a feedback in this case?

#60 - 04/06/2018 10:19 AM - Greg Shah

> If resources are external, then only urls will be sent to the client side. Does it need to have a feedback in this case?

No.

#61 - 04/07/2018 10:38 AM - Sergey Ivanovskiy

Please review committed revision 11256 that fixed #3474-55 and #3474-56.

#62 - 04/09/2018 07:12 AM - Greg Shah

Code Review Task Branch 3474a Revision 11256

This is really good. I think it is almost finished.

1. Please change kw_open_mime_resource to be kw_openmime. We always limit the KW_* constants to a maximum length of 11 characters including the KW_ prefix.

2. On the new Keyword("open-mime-resource"... line, please add a comment // FWD extension, not real 4GL!.

3. Please change the keyword text for "p2j-open-url" to "open-url". It is more consistent with our current approach which **does not** use a p2j-/fwd- prefix.

4. Do we need to have both MSG_OPEN_PRINT_OUTPUT and MSG_OPEN_MIME_RESOURCE messages? Doesn't MSG_OPEN_MIME_RESOURCE handle both needs?

5. It seems that our MSG_OPEN_PRINT_OUTPUT and MSG_OPEN_MIME_RESOURCE message is limited to 255 bytes in size and the URL is also limited to 255 characters (if the entire URL could be fit). I say this because it seems that the size of the message and URL are both limited to a number tat can be fit into a single byte. This seems to be too small and it also might be a kind of buffer overflow problem if the input is larger.

6. Please check if we have other web socket cases where our strings are limited to 255 characters without any checking of actual lengths.

7. PDFPrintOutput is not always a PDF, right? Shouldn't we name this class differently to be more clear?

8. Is TC.openMimeResource() a superset of TC.openURL()? It seems like the server side can just call TC.openMimeResource() with the right parameters to get the same result.

9. For SwingGuiDriver.openMimeResource() can't we honor the mimeType and embedded parameters?

10. build.xml, PDFPrintOutput, GUIPrinterStreamSupport are missing history entries.

#63 - 04/09/2018 09:36 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Code Review Task Branch 3474a Revision 11256

Thank you, planning to fix all these issues.

> 4. Do we need to have both MSG_OPEN_PRINT_OUTPUT and MSG_OPEN_MIME_RESOURCE messages? Doesn't MSG_OPEN_MIME_RESOURCE handle both needs?

Yes, it seems that MSG_OPEN_URL can be delegated to MSG_OPEN_MIME_RESOURCE too.

> 5. It seems that our MSG_OPEN_PRINT_OUTPUT and MSG_OPEN_MIME_RESOURCE message is limited to 255 bytes in size and the URL is also limited to 255 >characters (if the entire URL could be fit). I say this because it seems that the size of the message and URL are both limited to a number tat can be
> fit into a single byte. This seems to be too small and it also might be a kind of buffer overflow problem if the input is larger.

Yes, the id field in MSG_OPEN_PRINT_OUTPUT holds UUID and has 16 byte length, but MSG_OPEN_MIME_RESOURCE can hold URL and the mime type field can be a long string. I missed all these facts.

> 8. Is TC.openMimeResource() a superset of TC.openURL()? It seems like the server side can just call TC.openMimeResource() with the right parameters to get the same result.

Yes.

> 9. For SwingGuiDriver.openMimeResource() can't we honor the mimeType and embedded parameters?

I didn't implement a special code because this command should open a browser with a file resource. It seems that the embedded value, false or true, doesn't have an affect on this case. And the mimeType value looks unimportant in this case.

#64 - 04/09/2018 10:31 AM - Greg Shah

I didn't implement a special code because this command should open a browser with a file resource.

It could be a external URL too.

It seems that the embedded value, false or true, doesn't have an affect on this case. And the mimeType value looks unimportant in this case.

I don't understand this part. If we have generated a PDF and are displaying it, the user will want the PDF to appear as a browser tab in the browser launched for the Swing client, just as they would see in the web client. The only difference between these cases is the fact that we must launch a browser in the Swing case. Otherwise, it should act the same.

#65 - 04/10/2018 05:52 AM - Sergey Ivanovskiy

Please review committed revision 11258 that fixed #3474-62.

#66 - 04/10/2018 06:09 AM - Sergey Ivanovskiy

Please review committed revision 11259 (missed changes) that removed ScreenDriver.this.openURL since openURL was delegated to openMimeResource.

#67 - 04/10/2018 08:50 AM - Greg Shah

Code Review Task Branch 3474a Revision 11259

I'm good with the changes. I did check in some minor formatting and comment cleanup.

Hynek: Please review.

#68 - 04/10/2018 01:46 PM - Hynek Cihlar

Code review 3474a revision 11260.

open-resource.html:
"https://" + window.location.host + "${webRoot}/${documentHandler}/${documentPath}";
The expression misses location.port.

lang_stmts.rules
OPEN_MIME-RESOURCE -> OPEN-MIME-RESOURCE

Why do we need both keywords OPEN-MIME-RESOURCE and OPEN-URL? Is there any use case that OPEN-URL couldn't handle? A file local to FWD client can be expressed with a url such as file:///path/filename.

The doc for the method openMimeResource in several classes mentions path as the second param. But this can

be a full URL according to all the implementations. Please rename the parameter and fix the documentation for it.

GuiWebDriver.openMimeResource()
The opened input stream should be closed in a finalizer. Also the consumer can be called multiple times and so it must reset the stream before it reads it.
The method should validate the input url in all the cases, not only the for the file scheme.

SwingGuiDriver.openMimeResource()
There is a standard class Desktop. I think it performs exactly those operations we need for opening media files and urls. It declares two methods, open() and browse(). The former launches the application associated with the provided file type. The latter opens the desktop default browser and navigates to the provided url. Currently SwingGuiDriver.openMimeResource() always opens the preconfigured browser, so even if the user has Open Office installed, and thus capable to open the open office files directly, the current implementation will anyway launch a browser and download the file to a dir. Not very useful I think.

MSG_OPEN_MIME_RESOURCE message handler in p2j.socket.js doesn't handle absolute urls, buildOpenResourceUrl always prepends p2j.webRoot to the provided path.

SwingGuiDriver.deliverDocumentOutput() still refers to PDF in its javadoc.

DocumentOutputHandler still matches the "print" path.

Please move the PRINT OUTPUT related javadoc from DocumentOutputStorage to GUIPrinterStreamSupport.

Plus I renamed ReportOutputFormat to MediaType and moved it in com.goldencode.p2j.util.

#69 - 04/10/2018 02:32 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 3474a revision 11260.
>
> open-resource.html:
> "https://" + window.location.host + "${webRoot}/${documentHandler}/${documentPath}";
> The expression misses location.port.

This expression is correct according to https://developer.mozilla.org/en-US/docs/Web/API/HTMLHyperlinkElementUtils/host . Please look at this documentation. host expression includes port.

#70 - 04/10/2018 02:46 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> GuiWebDriver.openMimeResource()
> The opened input stream should be closed in a finalizer. Also the consumer can be called multiple times and so it must reset the stream before it reads it.
> The method should validate the input url in all the cases, not only the for the file scheme.

I don't understand why it requires finalizer because the try with resource construction is used. Please explain it more thoroughly.

#71 - 04/10/2018 02:52 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 3474a revision 11260.
> SwingGuiDriver.openMimeResource()
> There is a standard class Desktop. I think it performs exactly those operations we need for opening media files and urls. It declares two methods, open() and browse(). The former launches the application associated with the provided file type. The latter opens the desktop default browser and navigates to the provided url. Currently SwingGuiDriver.openMimeResource() always opens the preconfigured browser, so even if the user has Open Office installed, and thus capable to open the open office files directly, the current implementation will anyway launch a browser and download the file to a dir. Not very useful I think.

It seems that now it is implemented as for the web client. If Desktop class helps to open Open Office documents, then this implementation will have different functionality. I don't object to implement this idea with Desktop.

Greg, please approve this requirement.

#72 - 04/10/2018 02:57 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 3474a revision 11260.
>
> Plus I renamed ReportOutputFormat to MediaType and moved it in com.goldencode.p2j.util.

It seems that it is already done, isn't it?

#73 - 04/10/2018 02:58 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I don't understand why it requires finalizer because the try with resource construction is used. Please explain it more thoroughly.

You are right, the stream is opened only for the duration of the consumer. Please disregard this.

#74 - 04/10/2018 02:59 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> > Plus I renamed ReportOutputFormat to MediaType and moved it in com.goldencode.p2j.util.
>
> It seems that it is already done, isn't it?

Yes.

#75 - 04/10/2018 03:45 PM - Greg Shah

> If Desktop class helps to open Open Office documents, then this implementation will have different functionality. I don't object to implement this idea with Desktop.

Having a different behavior is OK, if it is a better approach for the user. But we tried using Desktop and it caused a hang in Swing. See #1815-25.

Also, there is the case where isDesktopSupported() is false in which case the Desktop class cannot be used.

If the Desktop class will work properly, then I am OK with using it in the Swing client. But we must have a standard fall-back approach that always works.

#76 - 04/10/2018 04:18 PM - Sergey Ivanovskiy

> MSG_OPEN_MIME_RESOURCE message handler in p2j.socket.js doesn't handle absolute urls,
> buildOpenResourceUrl always prepends p2j.webRoot to the provided path.

If the resource is streamed, then it is downloaded from the web client. If it is not streamed, then the template document with absolute url is downloaded
from the web client. Don't understand what is incorrect. Please explain.

#77 - 04/10/2018 04:43 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> MSG_OPEN_MIME_RESOURCE message handler in p2j.socket.js doesn't handle absolute urls,
> buildOpenResourceUrl always prepends p2j.webRoot to the provided path.

> If the resource is streamed, then it is downloaded from the web client. If it is not streamed, then the template document with absolute url is downloaded
> from the web client. Don't understand what is incorrect. Please explain.

If I interpret the program logic correctly, the variable id in case types.MSG_OPEN_MIME_RESOURCE: in p2j.socket.js can be an absolute URL (http://example.com). So when not embedding and local the resulting url in buildOpenResourceUrl is calculated as p2j.webRoot + "/print/" + "http://example.com";.

#78 - 04/10/2018 04:47 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > MSG_OPEN_MIME_RESOURCE message handler in p2j.socket.js doesn't handle absolute urls,
> > buildOpenResourceUrl always prepends p2j.webRoot to the provided path.

> > If the resource is streamed, then it is downloaded from the web client. If it is not streamed, then the template document with absolute url is downloaded
> > from the web client. Don't understand what is incorrect. Please explain.

> If I interpret the program logic correctly, the variable id in case types.MSG_OPEN_MIME_RESOURCE: in p2j.socket.js can be an absolute URL (http://example.com). So when not embedding and local the resulting url in buildOpenResourceUrl is calculated as p2j.webRoot + "/print/" + "http://example.com";.

This url is not local because only file resources can be local.

#79 - 04/10/2018 04:48 PM - Hynek Cihlar

Also I think path in url += "/open/resource/?path=" + path; in buildOpenResourceUrl should be url encoded.

#80 - 04/10/2018 04:51 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Also I think path in url += "/open/resource/?path=" + path; in buildOpenResourceUrl should be url encoded.

Yes, agree.

#81 - 04/10/2018 04:54 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Sergey Ivanovskiy wrote:
> >
> > > MSG_OPEN_MIME_RESOURCE message handler in p2j.socket.js doesn't handle absolute urls, buildOpenResourceUrl always prepends p2j.webRoot to the provided path.
>
> > If the resource is streamed, then it is downloaded from the web client. If it is not streamed, then the template document with absolute url is downloaded
> > from the web client. Don't understand what is incorrect. Please explain.
>
> If I interpret the program logic correctly, the variable id in case types.MSG_OPEN_MIME_RESOURCE: in p2j.socket.js can be an absolute

> URL (http://example.com). So when not embedding and local the resulting url in buildOpenResourceUrl is calculated as p2j.webRoot + "/print/" + "http://example.com";.

> This url is not local because only file resources can be local.

Anyway, buildOpenResourceUrl always prepends the url with the webRoot as already mentioned. When not local the expression will be p2j.webRoot + "/open/resource/?path=" + "http://example.com".

#82 - 04/10/2018 04:56 PM - Hynek Cihlar

Hynek Cihlar wrote:

> Anyway, buildOpenResourceUrl always prepends the url with the webRoot as already mentioned. When not local the expression will be p2j.webRoot + "/open/resource/?path=" + "http://example.com".

The requirement was to use such url directly: window.location = "https://example.com".

#83 - 04/11/2018 12:10 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Hynek Cihlar wrote:
>
> > Anyway, buildOpenResourceUrl always prepends the url with the webRoot as already mentioned. When not local the expression will be p2j.webRoot + "/open/resource/?path=" + "http://example.com".

It is GET request to the application servlet and the servlet responds with the wrapped document with this url = "http://example.com". The request is built according to the web client design, thus it starts from webRoot in this GET request.

> The requirement was to use such url directly: window.location = "https://example.com".

What do you propose? I see two variants:

1. The servlet responds with redirect responses.
2. The document loaded from this servlet makes this redirect via jscript on load.

```
<script type="text/javascript">
function onLoad()
{
   var config = {'isStreamed' : ${isStreamed} };
```

```
   var el = document.getElementById("embeddedDocument");
   if (config.isStreamed)
   {
      el.src = "https://" + window.location.host + "${webRoot}/${documentHandler}/${documentPath}";
   }
   else
   {
      if (not embedded document)
      {
        window.location = "${documentPath}";
      }
      else
      {
      el.src = "${documentPath}";
      }
   }
   el.width  = window.screen.availWidth;
   el.height = window.screen.availHeight;
}

onLoad();
</script>
```

#84 - 04/11/2018 03:43 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > Hynek Cihlar wrote:
> >
> > > Anyway, buildOpenResourceUrl always prepends the url with the webRoot as already mentioned. When not local the expression will be p2j.webRoot + "/open/resource/?path=" + "http://example.com".
>
> It is GET request to the application servlet and the servlet responds with the wrapped document with this url = "http://example.com". The request is built according to the web client design, thus it starts from webRoot in this GET request.
>
> > The requirement was to use such url directly: window.location = "https://example.com".
>
> What do you propose? I see two variants:

When not embedding and not streaming (i.e. having valid absolute url with domain) do window.open(id, "_blank").

#85 - 04/11/2018 04:52 AM - Sergey Ivanovskiy

OK, it is 3rd variant.

#86 - 04/11/2018 07:58 AM - Sergey Ivanovskiy

Hynek, please note that your changes in this code

```
=== modified file 'src/com/goldencode/p2j/ui/client/gui/driver/web/GuiWebDriver.java'
--- src/com/goldencode/p2j/ui/client/gui/driver/web/GuiWebDriver.java    2018-04-10 09:49:19 +0000
+++ src/com/goldencode/p2j/ui/client/gui/driver/web/GuiWebDriver.java    2018-04-10 17:42:14 +0000
@@ -172,7 +172,6 @@

 import com.goldencode.p2j.cfg.*;
 import com.goldencode.p2j.main.*;
-import com.goldencode.p2j.reporting.*;
 import com.goldencode.p2j.ui.*;
 import com.goldencode.p2j.ui.chui.ThinClient;
 import com.goldencode.p2j.ui.client.*;
@@ -2140,7 +2139,7 @@
    @Override
    public void deliverDocumentOutput(Consumer<OutputStream> consumer,
                                      Runnable               finalizer,
-                                     ReportOutputFormat     format)
+                                     MediaType format)
    {
       UUID uuid = UUID.randomUUID();
       DocumentOutput printOut = new DocumentOutput(consumer,
@@ -2184,6 +2183,7 @@
             {
                try(InputStream input = url.openStream())
                {
+                  input.reset();
                   ByteStreams.copy(input, output);
                }
                catch (IOException e)
```

are incorrect. I fixed this change due to exception java.lang.RuntimeException: java.io.IOException: Resetting to invalid mark.

#87 - 04/11/2018 08:00 AM - Sergey Ivanovskiy

I see you already fixed it. Thank you.

#88 - 04/11/2018 08:02 AM - Sergey Ivanovskiy

Greg Shah wrote:

> If Desktop class helps to open Open Office documents, then this implementation will have different functionality. I don't object to implement this idea with Desktop.

> Having a different behavior is OK, if it is a better approach for the user. But we tried using Desktop and it caused a hang in Swing. See #1815-25.
>
> Also, there is the case where isDesktopSupported() is false in which case the Desktop class cannot be used.
>
> If the Desktop class will work properly, then I am OK with using it in the Swing client. But we must have a standard fall-back approach that always works.

I checked that now using Desktop doesn't freeze the Swing client.

#89 - 04/11/2018 08:04 AM - Sergey Ivanovskiy

Please review committed revision 11263.

#90 - 04/11/2018 09:03 AM - Greg Shah

Code Review Task Branch 3474a Revision 11263

I'm good with the changes.

Hynek: Do you have any objections?

#91 - 04/11/2018 09:22 AM - Hynek Cihlar

Code review 3474a revision 11263

SwingGuiDriver.openMimeResource()
The method will throw an unexpected exception when non-file uri scheme is given to it. The subjected line is Desktop.getDesktop().open(new File(uri));. Use open(File) only for file schemes, for the rest use Desktop.browse.

The uri string must be encoded in buildOpenResourceUrl() in all places where it is used as part of the webRoot url. Use encodeURIComponent. To decode the uri on the server, use URLDecoder.decode(uri, "UTF-8"). Remove String id = uri.toASCIIString(); from GuiWebDriver.openMimeResource().

#92 - 04/11/2018 09:39 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 3474a revision 11263
>
> SwingGuiDriver.openMimeResource()
> The method will throw an unexpected exception when non-file uri scheme is given to it. The subjected line is Desktop.getDesktop().open(new File(uri));. Use open(File) only for file schemes, for the rest use Desktop.browse.

Yes, agree.

> The uri string must be encoded in buildOpenResourceUrl() in all places where it is used as part of the webRoot url. Use encodeURIComponent. To decode the uri on the server, use URLDecoder.decode(uri, "UTF-8"). Remove String id = uri.toASCIIString(); from GuiWebDriver.openMimeResource().

It seems that String id = uri.toASCIIString(); does what is required. Otherwise, it needs to do changes in two files.

#93 - 04/11/2018 10:13 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 3474a revision 11263
>
> SwingGuiDriver.openMimeResource()
> The method will throw an unexpected exception when non-file uri scheme is given to it. The subjected line is Desktop.getDesktop().open(new File(uri));. Use open(File) only for file schemes, for the rest use Desktop.browse.

Committed revision 11264 fixed this issue.

#94 - 04/11/2018 10:20 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

Hynek Cihlar wrote:

> Code review 3474a revision 11263
>
> SwingGuiDriver.openMimeResource()
> The method will throw an unexpected exception when non-file uri scheme is given to it. The subjected line is Desktop.getDesktop().open(new File(uri));. Use open(File) only for file schemes, for the rest use Desktop.browse.

Yes, agree.

> The uri string must be encoded in buildOpenResourceUrl() in all places where it is used as part of the webRoot url. Use encodeURIComponent. To decode the uri on the server, use URLDecoder.decode(uri, "UTF-8"). Remove String id = uri.toASCIIString(); from GuiWebDriver.openMimeResource().

It seems that String id = uri.toASCIIString(); does what is required. Otherwise, it needs to do changes in two files.

toASCIIString() only escapes non-ascii characters, but it doesn't produce valid application/x-www-form-urlencoded content. Consider unsafe characters such as '&' or '?'. These will break the resulting uri when put in a query parameter.

#95 - 04/11/2018 10:27 AM - Sergey Ivanovskiy

Yes, understand now. Committed revision 11265 fixed this issue.

#96 - 04/11/2018 10:36 AM - Sergey Ivanovskiy

Please review the committed revision 11265.

#97 - 04/11/2018 10:48 AM - Greg Shah

Code Review Task Branch 3474a Revision 11265

I'm good with the changes.

Hynek?

#98 - 04/11/2018 12:23 PM - Sergey Ivanovskiy

3474a rev 11266 is rebased over the current trunc rev 11245.

#99 - 04/11/2018 03:09 PM - Hynek Cihlar

Code review 3474a revision 11266.

The changes look good. I have no more objections.

Sergey, also please test the different modes ((no)embedding, (no)streaming, (no)local, etc.) with file/path names having spaces, diacritical marks, unsafe uri characters (like ?, , x%x, etc.), etc.

#100 - 04/11/2018 03:58 PM - Greg Shah

Sergey: You can start ChUI regression testing on devsrv01.

#101 - 04/12/2018 06:37 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Code review 3474a revision 11266.
>
> The changes look good. I have no more objections.
>
> Sergey, also please test the different modes ((no)embedding, (no)streaming, (no)local, etc.) with file/path names having spaces, diacritical marks, unsafe uri characters (like ?, , x%x, etc.), etc.

Planning to add these tests in testcases.

#102 - 04/12/2018 06:40 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Sergey: You can start ChUI regression testing on devsrv01.

3474a passed conversion regression tests. Now main regression tests are running.

#103 - 04/12/2018 02:01 PM - Sergey Ivanovskiy

Please review committed revision 11267 that added UTF-8 charset to open-resource.html in order OPEN-MIME-RESOURCE works properly with embedded html documents given by external urls expressed in native languages

```
open-mime-resource "plain/html" "https://ru.wikipedia.org/wiki/Гагарин,_Юрий_Алексеевич" yes.
```

#104 - 04/12/2018 02:11 PM - Sergey Ivanovskiy

The first round of main-regression tests passed with 3 failed tests. The second round is running now.

#105 - 04/12/2018 03:31 PM - Hynek Cihlar

Code review 3474a revision 11267.

The change is OK.

#106 - 04/12/2018 04:01 PM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> The first round of main-regression tests passed with 3 failed tests. The second round is running now.

In two runs there are 2 common failed tests. But they are false negative. Planning to run them separately.

#107 - 04/12/2018 04:55 PM - Sergey Ivanovskiy

These 2 failed tests passed in 2 single runs.

#108 - 04/12/2018 05:02 PM - Eric Faulhaber

Please wait for task branch 3434a to be merged to trunk, then rebase and merge 3474a to trunk. Please confirm that 3474a builds cleanly after the rebase, but it is not necessary to re-run regression testing.

#109 - 04/12/2018 06:09 PM - Sergey Ivanovskiy

Rev 11268 was rebased over current trunc rev 11246. Committed revision 11269 fixed related to this branch javadoc warnings. The task branch was built successfully.
Now OpenJDK can help to detect only 2 javadoc warnings in these files src/com/goldencode/graphdb/GraphDB.java and src/com/goldencode/p2j/util/EnvironmentOps.java.
Can rev 11269 be merged to trunc?

#110 - 04/12/2018 06:19 PM - Sergey Ivanovskiy

Planning to merge rev 11269.

#111 - 04/12/2018 06:23 PM - Greg Shah

Yes, please do.

#112 - 04/12/2018 06:34 PM - Sergey Ivanovskiy

3474a has been merged into the trunk as bzr revision 11247 and archived.

#113 - 04/15/2018 10:29 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from Review to Closed

#114 - 05/08/2019 01:47 PM - Greg Shah

- Related to Feature #4077: add option to OPEN-MIME-RESOURCE to delete the original source file when the send is complete added

Files

test.html	355 Bytes	04/03/2018	Hynek Cihlar