User Interface - Bug #2792

Bug # 2677 (New): fix drawing and functional differences between P2J GUI and 4GL GUI

image loading issues

10/25/2015 11:18 AM - Constantin Asofiei

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Feature #2476: window icon support improvements	**Closed**

History

#1 - 10/25/2015 11:22 AM - Constantin Asofiei

- Parent task set to #2677

There are a few issues with loading images for the GUI project:

1. on server-side, images are not searched via the PROPATH - just a getResourceAsStream is done, to get it from the jar
2. on client-side, the search is done via PROPATH, but on the file-system relative to the client's startup directory; also, the search is done case-sensitive, not insensitive...
3. loading a customer .ico file fails with this:

```
Caused by: java.lang.IllegalArgumentException: Pixel stride times width must be less than or equal to the
scanline stride
    at java.awt.image.PixelInterleavedSampleModel.<init>(PixelInterleavedSampleModel.java:101)
    at java.awt.image.Raster.createInterleavedRaster(Raster.java:641)
    at com.twelvemonkeys.imageio.plugins.ico.DIBImageReader.readBitmap24(Unknown Source)
    at com.twelvemonkeys.imageio.plugins.ico.DIBImageReader.readBitmap(Unknown Source)
    at com.twelvemonkeys.imageio.plugins.ico.DIBImageReader.read(Unknown Source)
    at com.twelvemonkeys.imageio.plugins.ico.ICOImageReader.read(Unknown Source)
    at javax.imageio.ImageReader.read(ImageReader.java:939)
    at com.goldencode.p2j.ui.client.gui.driver.AbstractGuiDriver.createIcon(AbstractGuiDriver.java:288)
    at com.goldencode.p2j.ui.client.gui.WindowTitleBar$WindowIcon.setIconData(WindowTitleBar.java:516)
    at com.goldencode.p2j.ui.client.gui.WindowTitleBar$WindowIcon.loadIcon(WindowTitleBar.java:571)
    at com.goldencode.p2j.ui.client.gui.WindowGuiImpl.loadWindowIcon(WindowGuiImpl.java:1281)
    at com.goldencode.p2j.ui.chui.ThinClient$60.run(ThinClient.java:17944)
    at com.goldencode.p2j.ui.chui.ThinClient.eventBracket(ThinClient.java:13741)
    at com.goldencode.p2j.ui.chui.ThinClient.eventDrawingBracket(ThinClient.java:13682)
    at com.goldencode.p2j.ui.chui.ThinClient.independentEventDrawingBracket(ThinClient.java:13555)
    at com.goldencode.p2j.ui.chui.ThinClient.loadWindowIcon(ThinClient.java:17935)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:497)
    at com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:76)
    at com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:705)
    at com.goldencode.p2j.net.Conversation.block(Conversation.java:319)
    at com.goldencode.p2j.net.Conversation.waitMessage(Conversation.java:257)
    at com.goldencode.p2j.net.Queue.transactImpl(Queue.java:1128)
    at com.goldencode.p2j.net.Queue.transact(Queue.java:585)
    at com.goldencode.p2j.net.BaseSession.transact(BaseSession.java:223)
    at com.goldencode.p2j.net.HighLevelObject.transact(HighLevelObject.java:163)
```

#2 - 10/25/2015 06:38 PM - Greg Shah

- Assignee set to Eugenie Lyzenko

Eugenie: please make this your highest priority. It is the cause of issues with the demo code.

#3 - 10/26/2015 09:50 AM - Eugenie Lyzenko

> Eugenie: please make this your highest priority. It is the cause of issues with the demo code.

OK. Let's clarify some points.
1. What is the correct loading sequence now. Consider:
on the server side:
- loading from application jar file, if not found then loading from the server PROPATH, correct?
on the client side:
- loading from p2j.jar, if not found - loading from client side file system, current or from root directory, if not found - loading from PROPATH on the client side, correct?

2. What is the particular issue with case sensitive file name? Please provide requested image name and the real image name, Windows or Linux?

3. For issue with java.lang.IllegalArgumentException: I need to have the picture/icon that causes this exception for detailed investigation.

#4 - 10/26/2015 10:00 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> > Eugenie: please make this your highest priority. It is the cause of issues with the demo code.
>
> OK. Let's clarify some points.
> 1. What is the correct loading sequence now. Consider:
> on the server side:
> - loading from application jar file, if not found then loading from the server PROPATH, correct?

Yes, but before going into the filesystem, check the PROPATH relative to the root package (i.e. com.goldencode.testcases) in the customer's jar.

> on the client side:
> - loading from p2j.jar, if not found - loading from client side file system, current or from root directory, if not found - loading from PROPATH on the client side, correct?

Yes, I think this is correct.

> 2. What is the particular issue with case sensitive file name? Please provide requested image name and the real image name, Windows or

Linux?

On Linux, a path like this (in the 4GL code, via load-icon) graphics/ICONs/someicon.ico will not work if the file is stored in a graphics/icons/someicon.ico path. I think this might be solved via a directory configuration, but I don't know what is it.

3. For issue with java.lang.IllegalArgumentException: I need to have the picture/icon that causes this exception for detailed investigation.

I'll send you a separate email.

#5 - 10/26/2015 11:41 AM - Eugenie Lyzenko

The case sensitivity is the config issue. For file to be case insensitive it is required to properly set up container:

```
...
      <node class="container" name="standard">
        <node class="container" name="runtime">
          <node class="container" name="default">
            <node class="container" name="file-system">
              ...
              <node class="boolean" name="case-sensitive">
                <node-attribute name="value" value="FALSE"/>
              </node>
...
```

in server's directory.xml file.

#6 - 10/26/2015 01:54 PM - Eugenie Lyzenko

The root cause for this:

```
...
    Caused by: java.lang.IllegalArgumentException: Pixel stride times width must be less than or equal to the
scanline stride
        at java.awt.image.PixelInterleavedSampleModel.<init>(PixelInterleavedSampleModel.java:101)
        at java.awt.image.Raster.createInterleavedRaster(Raster.java:641)
        at com.twelvemonkeys.imageio.plugins.ico.DIBImageReader.readBitmap24(Unknown Source)
        at com.twelvemonkeys.imageio.plugins.ico.DIBImageReader.readBitmap(Unknown Source)
```

```
        at com.twelvemonkeys.imageio.plugins.ico.DIBImageReader.read(Unknown Source)
        at com.twelvemonkeys.imageio.plugins.ico.ICOImageReader.read(Unknown Source)
        at javax.imageio.ImageReader.read(ImageReader.java:939)
...
```

is the incompatibility of the TwelveMonkeys 3.0.1 libraries we use to load icon with the icon format used by the customer's icons. So the solution I offer is to change to the newer TwelveMonkeys 3.1.2 version where this icon is loading fine(I have tested). This is better than convert all such icons to the format the old TwelveMonkeys can read.

Is it OK? I'll check in the 3.1.2 libraries into 2677a.

#7 - 10/26/2015 05:18 PM - Greg Shah

> on the server side:
> - loading from application jar file, if not found then loading from the server PROPATH, correct?

Yes, but before going into the filesystem, check the PROPATH relative to the root package (i.e. com.goldencode.testcases) in the customer's jar.

I think it is only important to check the customer jar using the root package + each PROPATH segment in turn. We don't need to check the server's filesystem.

> Is it OK? I'll check in the 3.1.2 libraries into 2677a.

Yes, it is OK.

#8 - 10/26/2015 06:14 PM - Eugenie Lyzenko

Task branch 2677a for review updated to revision 10963.

This is TwelveMonkeys 3.1.2 refresh.

Continue working on issue. The client side part works fine with absolute path but unable to load file relative to the client start-up current directory for case-insensitive name. The Utils.getCaseInsensitiveFilenameMatches(fileName) need to be fixed to take into account current directory rather than start searching from root.

All these cases must be considered as current directory based(if the file separator is "/"):

```
./DIR/filename
../DIR/filename
DIR/filename
filename
```

Correct?

#9 - 10/26/2015 08:00 PM - Eugenie Lyzenko

Task branch 2677a for review updated to revision 10964.

This is the client side image loading fix. Added functionality to detect/handle current directory based image path.

Continue working with server side part.

#10 - 10/26/2015 09:57 PM - Eugenie Lyzenko

Task branch 2677a for review updated to revision 10965.

This is the server side image loading fix. If the image is not found it is searching in a PROPATH elements as a file name header.

#11 - 10/27/2015 03:28 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Task branch 2677a for review updated to revision 10965.
>
> This is the server side image loading fix. If the image is not found it is searching in a PROPATH elements as a file name header.

Only one concern: in LogicalTerminal.getImageStreamFromApplication() you have this code:

```
imgName.startsWith(File.separator)
```

This uses the server's OS file separator, but the PROPATH may use legacy separators (i.e. windows-like). Do we have this configured somewhere? An alternative would be to replace all windows-like separators (\) with / - this is OK because searching the jar always requires the /, not \.

#12 - 10/27/2015 04:23 AM - Constantin Asofiei

Eugenie, another issue: LogicalTerminal.getImageStreamFromApplication() needs to prepend the root package name, not just use the PROPATH values directly. So insteazd of having something like somepropath/graphics/image.bmp, you need to use /path/to/root/package/somepropath/graphics/image.bmp.

#13 - 10/27/2015 04:47 AM - Constantin Asofiei

And, in addition to note 12, the server-side search needs to be done case sensitive/insensitive, as the client-side search is done.

#14 - 10/27/2015 07:28 AM - Greg Shah

> All these cases must be considered as current directory based(if the file separator is "/"):
> ...
> Correct?

Yes.

#15 - 10/27/2015 07:32 AM - Greg Shah

> An alternative would be to replace all windows-like separators (\) with / - this is OK because searching the jar always requires the /, not \.

Yes, this is the right way when searching jars on the server.

Also: our code should support multiple customer jars on the server. Converted applications often have multiple jars (one for DMOs and/or one for UI frames and another for business logic). We don't want to assume that the resources are in the first jar, they could be in any of the customer jars.

#16 - 10/27/2015 07:44 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, another issue: LogicalTerminal.getImageStreamFromApplication() needs to prepend the root package name, not just use the PROPATH values directly. So insteazd of having something like somepropath/graphics/image.bmp, you need to use /path/to/root/package/somepropath/graphics/image.bmp.

Here you assume the initial image to be loaded is: graphics/image.bmp, right?

#17 - 10/27/2015 08:04 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, another issue: LogicalTerminal.getImageStreamFromApplication() needs to prepend the root package name, not just use the PROPATH values directly. So insteazd of having something like somepropath/graphics/image.bmp, you need to use /path/to/root/package/somepropath/graphics/image.bmp.
>
> Here you assume the initial image to be loaded is: graphics/image.bmp, right?

Correct.

#18 - 10/27/2015 08:17 AM - Eugenie Lyzenko

> And, in addition to note 12, the server-side search needs to be done case sensitive/insensitive, as the client-side search is done.

I think here we need to get some agreement to not over-complicate everything. The images in jar is completely our things to do so we can make our rules, correct? Say for example all names must be in lower case(what I would prefer to have) or case sensitivity is mandatory? What do you think?

#19 - 10/27/2015 08:58 AM - Greg Shah

> The images in jar is completely our things to do so we can make our rules, correct?

Sadly, no. Although we will create the scripts (e.g. build.xml) that puts images into the jar file, we do not and cannot control the filenames that are given at runtime. Those filenames may have arbitrary case and will not necessarily match our changed version in the jar file. Although some of these names are string literals, some of them can be given by end-user input or some kind of calculation. Neither of these cases can be controlled. Considering the names may not match in case at any time, we MUST implement case-insensitive matching.

#20 - 10/27/2015 10:56 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, another issue: LogicalTerminal.getImageStreamFromApplication() needs to prepend the root package name, not just use the PROPATH values directly. So insteazd of having something like somepropath/graphics/image.bmp, you need to use /path/to/root/package/somepropath/graphics/image.bmp.

What is the root package in our case? com.goldencode.p2j.main?

#21 - 10/27/2015 10:59 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, another issue: LogicalTerminal.getImageStreamFromApplication() needs to prepend the root package name, not just use the PROPATH values directly. So insteazd of having something like somepropath/graphics/image.bmp, you need to use /path/to/root/package/somepropath/graphics/image.bmp.
>
> What is the root package in our case? com.goldencode.p2j.main?

The root package is the one defined in the SourceNameMapper.pkgroot. For our testcases project case, the root package name is com.goldencode.testcases.

#22 - 10/27/2015 01:45 PM - Eugenie Lyzenko

> Also: our code should support multiple customer jars on the server. Converted applications often have multiple jars (one for DMOs and/or one for UI frames and another for business logic). We don't want to assume that the resources are in the first jar, they could be in any of the customer jars.

I need some clarifications.

1. When I call getSystemClassLoader() in LogicalTerminal the resources from build/lib/p2jadmin.jar and build/lib/p2j.jar already loaded into memory in

addition to the jar files linked in manifests of these two files(in addition to the all dependencies). So actually I'm searching inside everything is currently loaded at the time of search. Correct?

2. There can be another jar files not currently loaded into memory but having the required resources. The full list of these jars can be obtained by listing the directory from System.getProperty("java.class.path") or System.getProperty("managed.libs.dir", null) or MultiClassLoader.managedCustomerLibs(). Correct?

3. What we need to do is to enumerate all customer jar files, load them into memory, make case insensitive search for resource and load the resource if found. Is this right approach?

#23 - 10/27/2015 04:34 PM - Eugenie Lyzenko

Task branch 2677a for review updated to revision 10973.

This is the reworked server side approach to image loading. The previous implementation should be included as the part of the jar file from the list. The base is to take the complete application jar list and scan for case insensitive match rooted from PROPATH entries.

The case insensitive search is based on mapping between original resource name in jar and lowercase letter version. I requested image is the same in lowercase - the original resource name is used to get the stream from jar.

Need some feedback if this implementation is OK or not. I'm going to prepare standalone jar file with only images to test loading.

#24 - 10/27/2015 05:12 PM - Greg Shah

> 1. When I call getSystemClassLoader() in LogicalTerminal the resources from build/lib/p2jadmin.jar and build/lib/p2j.jar already loaded into memory in addition to the jar files linked in manifests of these two files(in addition to the all dependencies). So actually I'm searching inside everything is currently loaded at the time of search. Correct?

I don't think we need to search any of the P2J jars or their dependencies. These resources will be in the converted application jars only.

The system class loader will already be able to walk the classpath and determine which jars have resources in the pkgroot. You need to enumerate the resources in the pkgroot and do a case-insensitive match on that list.

> There can be another jar files not currently loaded into memory but having the required resources. The full list of these jars can be obtained by listing the directory from System.getProperty("java.class.path") or System.getProperty("managed.libs.dir", null) or MultiClassLoader.managedCustomerLibs(). Correct?

Do we really need to know about the specific jars? Can't we base this on the pkgroot? That would exclude all the other jars in the classpath immediately. Likewise, I don't think we care about whether something has been loaded or not. The class loader takes care of that, doesn't it?

#25 - 10/27/2015 05:22 PM - Eugenie Lyzenko

The short testing results:
The following addition to the server start-up script

```
...
cpath="../../../build/lib/p2jadmin.jar:../../../build/lib/p2j.jar:../../../resources/test.jar:"
...
```

allows the images to be loaded from test.jar on the server side. And it is case insensitive.

I'm still not sure is the place above to define custom application jar file is OK. What do you think? We have some place in server's directory.xml for this purpose too?

#26 - 10/27/2015 05:25 PM - Greg Shah

I have looked at rev 10973. I don't think we want to directly use the JarClassLoader and manually load jars. Can't the standard ClassLoader interfaces provide enough enumeration tools to find the resources under pkgroot?

Stanislav: you implemented our multi-class loader infrastructure. Perhaps you have some ideas?

> The following addition to the server start-up script
> ...
> allows the images to be loaded from test.jar on the server side. And it is case insensitive.

The application jar files are already in the classpath. I would hope this is not needed.

#27 - 10/27/2015 08:49 PM - Eugenie Lyzenko

> I have looked at rev 10973. I don't think we want to directly use the JarClassLoader and manually load jars. Can't the standard ClassLoader interfaces provide enough enumeration tools to find the resources under pkgroot?

I did look and think the answer is no. For case insensitive scan we need to have two sets, one - original and other - case neutral(say lowercased) and we need a mapping between them. Then we need to compare requested name normalized to case neutral with what we have inside jar - then using mapping we can get the original name in jar and able to load the resource. In MultiClassLoader I do not see tools to do this, not to get the list of the application jar files. May be I have some configuration issues, say have to setup managed.libs.dir system property or managed-libs-dir in server's directory.xml.

#28 - 10/27/2015 09:06 PM - Eugenie Lyzenko

Task branch 2677a for review updated to revision 10974.

Small exclude directories and classes from resource jar map.

#29 - 10/27/2015 09:20 PM - Greg Shah

> I don't think we want to directly use the JarClassLoader and manually load jars. Can't the standard ClassLoader interfaces provide enough enumeration tools to find the resources under pkgroot?

I did look and think the answer is no.

You've tried ClassLoader.findResources(), ClassLoader.getResources() and ClassLoader.getPackages()?

#30 - 10/27/2015 10:39 PM - Eugenie Lyzenko

> You've tried ClassLoader.findResources(), ClassLoader.getResources() and ClassLoader.getPackages()?

I have considered these calls. ClassLoader.findResources() and ClassLoader.getResources() takes resource name as parameter. And here we have collision. If the requested resource name differs only in letter case these methods will fail when the resource exists. The comparison should be done on a neutral territory, not inside class loader because java names are case sensitive inside jar.

About ClassLoader.getPackages(). Even when we can make this call by adding some methods to MultiClassLoader due to protected access for this method. What we can do with returned Package[] array? Only get the package name? Am I missing for something important?

#31 - 10/28/2015 12:04 AM - Greg Shah

I've done some more research and some of my own tests. You're right. The enumeration capabilities are limited to providing a list of all of the exact name matches across the full set of classpath components (jars and the local filesystem). But there really is no wildcard nor is there an open ended enumerate "what is in this package".

It is sad really, since this seems to be quite a useful thing.

Anyway, keep going with your current approach.

#32 - 10/28/2015 07:50 AM - Constantin Asofiei

Eugenie, in the code in current rev:

1. server-side search needs to be case-sensitive or insensitive based on the directory configuration - now search is done case-insensitive only
2. if PROPATH contains values like ., ./some/path, etc, then they are not appended correctly - it ends up with a root/package/name./some/path/graphics/icons/image.png. (where propath entry is ./some/path). You need to make sure that the path doesn't contain ./, multiple consecutive / chars, etc, so that the match can be done. This is because we can't control how the propath entry or image name is specified - propat entry can or can not start with a ., can end or not with a /.

#33 - 10/28/2015 09:12 AM - Constantin Asofiei

Eugenie, one more thing: try caching the JarClassLoader instances, as initiating them is expensive.

#34 - 10/28/2015 11:19 AM - Eugenie Lyzenko

> server-side search needs to be case-sensitive or insensitive based on the directory configuration - now search is done case-insensitive only

Yes, this is implemented intentionally. My consideration is:
Assume there is /images/icon.ico in the jar file. In case-sensitive loading if someone want to have /images/iCon.ico - the loading will fail. But the user or program can think about exactly this file. Do we really this situation to be true?

The only advantage for the case sensitive life I see is the possibility to have the both /images/icon.ico and /images/iCon.ico inside the jar file. Is this critically important for the image loading?

I do not object to make case-sensitive search. And this is easy to implement. But I need to have a exact vision for our requirements here. Please confirm we need both case-sensitive and insensitive versions.

#35 - 10/28/2015 11:36 AM - Greg Shah

> The only advantage for the case sensitive life I see is the possibility to have the both /images/icon.ico and /images/iCon.ico inside the jar file. Is this critically important for the image loading?

This is generally not the reason. It could be that some obscure 4GL app does rely upon it. It would be a poor choice, but it would work.

But the real reason is that the filename input is evaluated at runtime and may include components that cannot be made consistent. We MUST match the original 4GL application's behavior. If the original app would fail if the mismatching case input is provided, then we must fail. This would happen for 4GL apps running on a case-sensitive filesystem. If the original app would succeed if mismatching case input is provided, then we must succeed. This would happen for the default Windows case-insensitive filesystem.

> Please confirm we need both case-sensitive and insensitive versions.

Confirmed. Yes, we need to support both depending on the directory configuration.

#36 - 10/28/2015 11:42 AM - Eugenie Lyzenko

> Confirmed. Yes, we need to support both depending on the directory configuration.

OK.

#37 - 10/28/2015 02:56 PM - Greg Shah

By the way, the PROPATH issue reported by Constantin in note 32 causes the demo code to fail to be able to load its icon. This problem exists in revision 10975 of 2677a.

Do you have an "ETA" on the remaining fixes in this task?

#38 - 10/28/2015 03:08 PM - Eugenie Lyzenko

Greg Shah wrote:

> By the way, the PROPATH issue reported by Constantin in note 32 causes the demo code to fail to be able to load its icon. This problem exists in revision 10975 of 2677a.

I see it is urgent.

> Do you have an "ETA" on the remaining fixes in this task?

In a 30 min I'll finish testing case sense implementation and PROPATH handling and upload the changes.

Then I need 1-2 hours more to implement jar class loaders caching. This will be separate upload. One question here. The LogicalTerminal is too big and not image specific to store such cache I guess. Can I create separate class in com.goldencode.p2j.ui.* package for this purpose? Say ServerImageWorker or something like this?

#39 - 10/28/2015 03:40 PM - Greg Shah

> Then I need 1-2 hours more to implement jar class loaders caching. This will be separate upload. One question here. The LogicalTerminal is too big and not image specific to store such cache I guess. Can I create separate class in com.goldencode.p2j.ui.* package for this purpose? Say ServerImageWorker or something like this?

Yes, this is a good idea.

#40 - 10/28/2015 04:33 PM - Eugenie Lyzenko

Task branch 2677a for review updated to revision 10977.

The JarClassLoader cache has been implemented here.

#41 - 10/28/2015 09:21 PM - Greg Shah

Code Review Task Branch 2677a Revision 10977

The changes are good.

#42 - 10/29/2015 08:47 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

Code Review Task Branch 2677a Revision 10979

The changes are good.

Testing confirms that the original problem is gone. Many other issues have also been resolved.

#43 - 01/08/2016 03:51 PM - Igor Skornyakov

I have a couple of question regarding ThinClient.loadImage() method:
1. According to my tests when 4GL loads mouse cursor from the file and the extension is not specified it actually looks for **.bmp**, **.ico**, **.cur** and empty extensions - in this order. The ThinClient.loadImage() actually looks for the file name w/o extension first. Is that correct? If course I can test but may be I miss something?
2. The logic based on CommonImageWorker.getExtensionListIterator() for case-sensitive file system looks like the search which is case-sensitive with respect of the name part and case-insensitive with respect of the extension which look a little bit strange. I cannot check it on windev01. Is it really how 4GL works on Linux? And of course this approach is very inefficient on case-insensitive file systems. May be it makes sense to improve it a bit?
Thank you.

#44 - 01/11/2016 11:10 AM - Eugenie Lyzenko

> I have a couple of question regarding ThinClient.loadImage() method:
> 1. According to my tests when 4GL loads mouse cursor from the file and the extension is not specified it actually looks for .bmp, .ico, .cur and empty extensions - in this order. The ThinClient.loadImage() actually looks for the file name w/o extension first. Is that correct? If course I can test but may be I miss something?

For images it is correct I think. Because if no-extension image is in PNG or JPG or from the valid defaults(.bmp, .ico, .cur) but still no extension - it is loading first. Because extension scan is time consuming as you can see.

> 2. The logic based on CommonImageWorker.getExtensionListIterator() for case-sensitive file system looks like the search which is case-sensitive with respect of the name part and case-insensitive with respect of the extension which look a little bit strange. I cannot check it on windev01. Is it really how 4GL works on Linux? And of course this approach is very ineffective on case-insensitive file systems.

You can not test it on Windows because the Windows is case insensitive. For Linux we have no GUI 4GL installation so I can not say for sure how it will work. The current implementation assumes maximum tolerance to possible mix of up and low extension letter cases. We can not control how the user will name the file, the maximum we can do is to check all possible combinations. And of course this has a price.

#45 - 01/11/2016 11:12 AM - Igor Skornyakov

Eugenie Lyzenko wrote:

> > I have a couple of question regarding ThinClient.loadImage() method:
> > 1. According to my tests when 4GL loads mouse cursor from the file and the extension is not specified it actually looks for .bmp, .ico, .cur and empty extensions - in this order. The ThinClient.loadImage() actually looks for the file name w/o extension first. Is that correct? If course I can test but may be I miss something?
>
> For images it is correct I think. Because if no-extension image is in PNG or JPG or from the valid defaults(.bmp, .ico, .cur) but still no extension - it is loading first. Because extension scan is time consuming as you can see.
>
> > 2. The logic based on CommonImageWorker.getExtensionListIterator() for case-sensitive file system looks like the search which is case-sensitive with respect of the name part and case-insensitive with respect of the extension which look a little bit strange. I cannot check it on windev01. Is it really how 4GL works on Linux? And of course this approach is very ineffective on case-insensitive file systems.
>
> You can not test it on Windows because the Windows is case insensitive. For Linux we have no GUI 4GL installation so I can not say for sure how it will work. The current implementation assumes maximum tolerance to possible mix of up and low extension letter cases. We can not control how the user will name the file, the maximum we can do is to check all possible combinations. And of course this has a price.

I see. Thank you Eugenie.

#46 - 01/11/2016 12:39 PM - Greg Shah

> 2. The logic based on CommonImageWorker.getExtensionListIterator() for case-sensitive file system looks like the search which is case-sensitive with respect of the name part and case-insensitive with respect of the extension which look a little bit strange. I cannot check it on windev01. Is it really how 4GL works on Linux? And of course this approach is very ineffective on case-insensitive file systems.

> You can not test it on Windows because the Windows is case insensitive. For Linux we have no GUI 4GL installation so I can not say for sure how it will work. The current implementation assumes maximum tolerance to possible mix of up and low extension letter cases. We can not control how the user will name the file, the maximum we can do is to check all possible combinations. And of course this has a price.

Both JarClassLoader.getResourceName() and FileSystemDaemon.searchPath() honor case-sensitivity in the complete name.

I'm trying to remember why we need to have the CommonImageWorker.getExtensionListIterator() provide all forms of the extensions. Won't these calling locations find the match based on the first .bmp and then processing .Bmp and all the rest is unnecessary work?

#47 - 01/11/2016 12:55 PM - Eugenie Lyzenko

Greg Shah wrote:

> Both JarClassLoader.getResourceName() and FileSystemDaemon.searchPath() honor case-sensitivity in the complete name.
>
> I'm trying to remember why we need to have the CommonImageWorker.getExtensionListIterator() provide all forms of the extensions. Won't these calling locations find the match based on the first .bmp and then processing .Bmp and all the rest is unnecessary work?

If there is no extension in the name to find - we do not initially know what file extension will match to the name requested. It can be BMP, ICO or CUR in any combination of the letter case. Example: "filename"(requested) and "filename.bMp"(actual file on the disk).

#48 - 01/11/2016 02:01 PM - Greg Shah

Yes, I understand. My point is that when we are processing on a case-sensitive file system (or searching the JAR which is always case-sensitive) AND the original 4GL system was case-INsensitive (e.g. windev01), then it seems to me that searching the list of .bmp, .Bmp, .bMp and so forth is not useful. The very first .bmp will already have case-insensitive matching applied in JarClassLoader.getResourceName() and FileSystemDaemon.searchPath(). In this case, if the .bmp doesn't match something then all the extra combinations of bmp will also always fail.

The only reason to use this "full list" is if we are processing on a case-sensitive file system (or searching the JAR which is always case-sensitive) AND the original 4GL system was case-sensitive. In that case, we need the full set of possible extensions.

Am I misunderstanding something?

If not, then there is an optimization here for the case-sensitive target system + case-INsensitive original system.

#49 - 01/11/2016 02:35 PM - Igor Skornyakov

Greg Shah wrote:

> Yes, I understand. My point is that when we are processing on a case-sensitive file system (or searching the JAR which is always case-sensitive) AND the original 4GL system was case-INsensitive (e.g. windev01), then it seems to me that searching the list of .bmp, .Bmp, .bMp and so forth is not useful. The very first .bmp will already have case-insensitive matching applied in JarClassLoader.getResourceName() and FileSystemDaemon.searchPath(). In this case, if the .bmp doesn't match something then all the extra combinations of bmp will also always fail.
>
> The only reason to use this "full list" is if we are processing on a case-sensitive file system (or searching the JAR which is always case-sensitive) AND the original 4GL system was case-sensitive. In that case, we need the full set of possible extensions.
>
> Am I misunderstanding something?
>
> If not, then there is an optimization here for the case-sensitive target system + case-INsensitive original system.

Strictly speaking this is not exactly correct. For example NTFS is case insensitive file system in the sense that **a.bmp** and **a.bMp** are equivalent. However if we have **a.bmp** and **b.bMp** and create a jar file then **b.bMp** could not be retrieved from the archive as **b.bmp**

Please note however that for the current approach to be compatible with 4GL behaviour (if we believe that it really supports different cases of extensions) the order of different versions in the iterator should be **exactly** the same as in Progress.

#50 - 01/11/2016 02:44 PM - Greg Shah

> Strictly speaking this is not exactly correct. For example NTFS is case insensitive file system in the sense that a.bmp and a.bMp are equivalent. However if we have a.bmp and b.bMp and create a jar file then b.bMp could not be retrieved from the archive as b.bmp

That is true for normal jar file access. But Eugenie has customized JarClassLoader.getResourceName() to provide case-insensitive searching.

#51 - 01/11/2016 02:54 PM - Igor Skornyakov

Greg Shah wrote:

> > Strictly speaking this is not exactly correct. For example NTFS is case insensitive file system in the sense that a.bmp and a.bMp are equivalent. However if we have a.bmp and b.bMp and create a jar file then b.bMp could not be retrieved from the archive as b.bmp
>
> That is true for normal jar file access. But Eugenie has customized JarClassLoader.getResourceName() to provide case-insensitive searching.

I see. But the result of the case-insensitive search in a case-sensitive file system is not predictable. Please not also that when we will implement http(s)-based PROPATH entries the approach with extensions iterator will become even more expensive. May be it is more safe just to insist that images' extensions should be e.g. lower case only (at least when extensions can be omitted in the code)?

#52 - 01/11/2016 03:23 PM - Greg Shah

> But the result of the case-insensitive search in a case-sensitive file system is not predictable.

True. But if a filename was used on the original system and one tried to do the same thing in P2J, it will work.

And we know there can only be 1 file that matches that name on the original 4GL system, so loading the first we find is acceptable.

> Please not also that when we will implement http(s)-based PROPATH entries the approach with extensions iterator will become even more expensive.

Yes, we already support this.

Yes, it is costly. But it is required.

> May be it is more safe just to insist that images' extensions should be e.g. lower case only (at least when extensions can be omitted in the code)?

We can't do this.

#53 - 01/11/2016 03:38 PM - Igor Skornyakov

Greg Shah wrote:

> And we know there can only be 1 file that matches that name on the original 4GL system, so loading the first we find is acceptable.

I'm not sure that it is always true. Imagine that there are multiple PROPATH entries and files with extensions which are different in the extension case only inside different entries. Or just such files in different folders of the same archive. The order of different extensions in p2j should be the the same as in 4GL to ensure compatibility.

> We can't do this.

I see. Thank you.

#54 - 01/11/2016 03:43 PM - Greg Shah

> > And we know there can only be 1 file that matches that name on the original 4GL system, so loading the first we find is acceptable.
>
> I'm not sure that it is always true. Imagine that there are multiple PROPATH entries and files with extensions which are different in the extension case only inside different entries. Or just such files in different folders of the same archive. The order of different extensions in p2j should be the the same as in 4GL to ensure compatibility.

I see your point.

However, the original system could not have multiple files in the same folder which only differ by case. Since we search the propath in the same order as does the 4GL (providing that the propath has been configured for us in the same order), then given the same filesystem state, P2J and the 4GL will both pick up the same first match.

That is the best that we can do, I think.

#55 - 01/11/2016 03:50 PM - Igor Skornyakov

Greg Shah wrote:

> I see your point.
>
> However, the original system could not have multiple files in the same folder which only differ by case. Since we search the propath in the same order as does the 4GL (providing that the propath has been configured for us in the same order), then given the same filesystem state, P2J and the 4GL will both pick up the same first match.

Imagine that we have to PROPATH entries. The first one contains **a.bmp** and the second one **a.BMP**. p2j will pick the first one. However if 4GL will try the **.BMP** first the second one will be picked.

> That is the best that we can do, I think.

I agree. I believe that it is better to pick the wrong file in the (rare) scenario described above rather than to fail erroneously in much more general case.

#56 - 01/11/2016 04:00 PM - Greg Shah

> However, the original system could not have multiple files in the same folder which only differ by case. Since we search the propath in the same order as does the 4GL (providing that the propath has been configured for us in the same order), then given the same filesystem state, P2J and the 4GL will both pick up the same first match.

> Imagine that we have to PROPATH entries. The first one contains **a.bmp** and the second one **a.BMP**. p2j will pick the first one. However if 4GL will try the **.BMP** first the second one will be picked.

I don't understand what you are saying here. This case-insensitive searching only is used when the target (P2J runtime) system has a case-sensitive file system AND the original (4GL runtime) system had a case-INsensitive file system.

In such a case, the 4GL searching for **a.BMP** in the propath will find **a.bmp** first, just the same way as P2J would.

#57 - 01/11/2016 04:02 PM - Igor Skornyakov

Greg Shah wrote:

> I don't understand what you are saying here. This case-insensitive searching only is used when the target (P2J runtime) system has a case-sensitive file system AND the original (4GL runtime) system had a case-INsensitive file system.
>
> In such a case, the 4GL searching for **a.BMP** in the propath will find **a.bmp** first, just the same way as P2J would.

Oh, I see. Sorry.

#58 - 01/12/2016 06:11 AM - Igor Skornyakov

I've used ProcessExplorer to monitor file operations when image file w/o an extension is loaded. As far as I can see the logic is the same as for loading mouse cursor:
1. The extensions are tried in the following order: **.bmp**, **.ico**, **.cur**, **<empty>**
2. The sequence described above is applied to each PROPATH entry (the outer loop is by PROPATH entries, not extensions).

The current logic in the ThinClient.loadImage() is different. Have I missed something and should I fix it?
Thank you.

#59 - 01/12/2016 06:38 AM - Eugenie Lyzenko

> I've used ProcessExplorer to monitor file operations when image file w/o an extension is loaded. As far as I can see the logic is the same as for >loading mouse cursor:
> 1. The extensions are tried in the following order: .bmp, .ico, .cur, <empty>
> 2. The sequence described above is applied to each PROPATH entry (the external loop is by PROPATH entries, not extensions).
>
> The current logic in the ThinClient.loadImage() is different. Have I missed something and should I fix it?

ThinClient.loadImage() uses first the original name. This name can be with extension and can be without. We do not know preliminary what is the case. So yes, the initial name is looking first. In the case there is no extension - it differs from what you described in point number 2 above. But note, the image file on disk can be without extension too.

Do we need to fix the current logic? I'm not sure. Anyway it will require to add the code to detect whether the initially requested name has extension or not. Because original name search(if extension exists) should be the first.

#60 - 01/12/2016 06:45 AM - Igor Skornyakov

Eugenie Lyzenko wrote:

> Do we need to fix the current logic? I'm not sure. Anyway it will require to add the code to detect whether the initially requested name has extension or not. Because original name search(if extension exists) should be the first.

I understand this. But it is a one line of code to check if the extension is provided or not.

#61 - 01/12/2016 06:52 AM - Eugenie Lyzenko

Igor Skornyakov wrote:

> I understand this. But it is a one line of code to check if the extension is provided or not.

I guess you will not cover all possible name cases with one line for name like this:

```
abc.bmp.ext
.nam
```

#62 - 01/12/2016 09:13 AM - Greg Shah

Igor: as part of your work for #2565, please do resolve the deviations found through your testing. In addition, please optimize the search (by avoiding unnecessary extra searching in the case I described in note 48).

#63 - 01/12/2016 09:14 AM - Igor Skornyakov

Greg Shah wrote:

> Igor: as part of your work for #2565, please do resolve the deviations found through your testing. In addition, please optimize the search (by avoiding unnecessary extra searching in the case I described in note 48).

OK, thank you.

#64 - 03/01/2016 01:02 PM - Greg Shah

The recently noted issues are resolved by trunk revision 10976 (merged from 2565a).

#65 - 11/16/2016 12:12 PM - Greg Shah

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

#66 - 01/30/2017 01:28 PM - Greg Shah

- Related to Feature #2476: window icon support improvements added