Base Language - Bug #2130

fix propath assignment

04/25/2013 09:36 AM - Greg Shah

Status:	Closed	**Start date:**	05/18/2013
Priority:	Normal	**Due date:**	05/20/2013
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	12.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #1648: review all filesystem support (runt...	**Closed**	**11/20/2012**

History

#1 - 04/25/2013 09:37 AM - Greg Shah

From note 78 in #1648 by EVL:

I have clarified this point. Both Windows and Linux 4GL systems shows the same behavior:

```
propath = "NewPropath".
display propath.
```

shows the sum of the new propath value with the old one. And new entries become in the head of the resulting string:
NewPropath,OldPropath

The update as of January 17, 2013 does not work this way. Currently we just substitute the old value with the new one.

#2 - 04/25/2013 04:16 PM - Eugenie Lyzenko

- File evl_upd20130425a.zip added

PROPATH behaviour fix uploaded.

While I'm thinking about one strange issue in 64-bit Windows I have prepared the fix for PROPATH behaviour mismatch. This update includes:
1. Adding new string constant PROPATH_SEPARATOR to simplify code management.
2. Modifications for fixupPropath and splitPath methods to take into account the PROPATH_SEPARATOR in the path string handling.
3. The method setSearchPath was reworked to simulate Progress behaviour. The new PROPATH entries will be added in the head of the new PROPATH and only in the case when we do not already have these entries in the current PROPATH value.

#3 - 05/22/2013 10:44 AM - Greg Shah

Code review feedback:

1. In setSearchPath() the work.obtain() is called many times (inline). This is not a good idea because it is very inefficient. Please call it just once and save the return value to be used multiple times.

2. In setSearchPath() the null test in the for loop is backwards:

```
for (int i = 0; i < pathNew.length && pathNew != null; i++)
```

It should be this way (to avoid an NPE):

```
for (int i = 0; pathNew != null && i < pathNew.length; i++)
```

3. In setSearchPath(), if the original propath was empty, then this code will leave a "dangling" separator at the end. If the 4GL does that too, then it is OK (but document it if so).

```
                if (work.obtain().propath.indexOf(pathNew[i]) == -1)
                {
                   // Propath separator value is not the same as OS dependent file separator
                   sb.append(pathNew[i]);
                   sb.append(PROPATH_SEPARATOR);
                }
```

This problem is also present in fixupPropath(). If that is how the 4GL does it then that is OK (but document it).

4. Does the checking for already existing propath entries need to honor the file system's case sensitivity? How does the 4GL work if two paths only differ by case? Does this 4GL behavior differ between Windows and Linux?

#4 - 05/22/2013 12:45 PM - Eugenie Lyzenko

- File evl_upd20130522a.zip added

> 1. In setSearchPath() the work.obtain() is called many times (inline). This is not a good idea because it is very inefficient. Please call it just once and save the return value to be used multiple times.

Fixed.

> 2. In setSearchPath() the null test in the for loop is backwards:
>
> ```
> for (int i = 0; i < pathNew.length && pathNew != null; i++)
> ```
>
> It should be this way (to avoid an NPE):
>
> ```
> for (int i = 0; pathNew != null && i < pathNew.length; i++)
> ```

OK. I have added one more check for null pointer:

```
for (int i = 0; pathNew != null && i < pathNew.length && pathNew[i] != null; i++)
```

> 3. In setSearchPath(), if the original propath was empty, then this code will leave a "dangling" separator at the end. If the 4GL does that too, then it is OK (but document it if so).
> ...
> This problem is also present in fixupPropath(). If that is how the 4GL does it then that is OK (but document it).

Now for sure this code will not be called when old PROPATH is empty. The Method fixupPropath() adds the separator only when it founds the old path separator, if the string is empty or has only one entry - the separator is not added. I have tested this in Linux. However it is possible the case when we have the final sting like: "NewPropath,,OldPropath" when for example the OldPropath starting with comma. This is how it works in native 4GL, so I have duplicated this in P2J.

> 4. Does the checking for already existing propath entries need to honor the file system's case sensitivity? How does the 4GL work if two paths only differ by case? Does this 4GL behavior differ between Windows and Linux?

OK. I'll check this point on both Windows and Linux 4GL systems.

#5 - 05/22/2013 02:18 PM - Eugenie Lyzenko

> 4. Does the checking for already existing propath entries need to honor the file system's case sensitivity? How does the 4GL work if two paths only differ by case? Does this 4GL behavior differ between Windows and Linux?

After testing on both 4GL systems show the strange behaviour I have not detected before. This is my fault due to the not enough testing made.

So the current status is: there is two PROPATH section in PROPATH variable, one is fixed and another - can be changed. The changeable initial string is "." in both Windows and Linux cases. So the initial PROPATH string is:
".,FixedPartOfThePropath"
Then when we change the propath via propath = "NewValue" the PROPATH become:
"NewValue,FixedPartOfThePropath"
Then when we want to change PROPATH via propath = "AnotherVeryNewValue" the PROPATH become:
"AnotherVeryNewValue,FixedPartOfThePropath"

The resulting PROPATH string become current for given 4GL session until the session terminates. This means in Procedure Editor it is required to restart the Procedure Editor to return to the system default PROPATH value.

So I need to rework setSearchPath() to reflect this handling. And I think this eliminates the case sensitivity point from consideration and support in P2J. The old changeable part replaces with the new one not depending on letter case status in strings.

#6 - 05/22/2013 05:23 PM - Eugenie Lyzenko

- File evl_upd20130522b.zip added

The update has been uploaded. The change reflects the currently detected 4GL behaviour. Within PROPATH changing we only change the variable part of PROPATH leaving the default part untouched.

The implementation idea is to store variable part in WorkArea with initial value as "." - current directory for all OS. The fixed default part is stored in common static variable and initialized once per server from ini file. Combining these two initial parts we getting the current default as in 4GL. In this approach we change only WorkArea value of the current process replacing the old value with the new one.

#7 - 05/28/2013 05:29 PM - Greg Shah

Code Review:

> The implementation idea is to store variable part in WorkArea with initial value as "." - current directory for all OS.

1. I don't think this is right. While there is a fixed part of the propath in the 4GL, I don't think it is true that the variable part is always ".". I think the variable part is whatever is defined as the PROPATH (environment variable) when the pro or mpro programs are started. In the new approach, this must be customizable (the default must be set in the directory). If I am right, we need 2 directory settings. One for the fixed part that can be saved in the propathDefault variable. The other would be for the user-defined part that can be anything and which can be changed via propath assignment.

2. The final computed "effective" propath (which is the changeable part + PROPATH_SEPARATOR + the fixed part) is not saved anywhere except for the client side. The WorkArea.propath only contains the current changeable part. This means that getLegacyPathOverride() not properly return the full effective propath. Please add a data member to WorkArea to store the current changeable part (and that can be initialized from the directory's default for the changeable part or "." if it is not found in the directory). Then the WorkArea.propath should be the fully computed propath which includes the fixed part. The getSearchPath() should also be changed so that it does not dynamically calculate the effective propath at each call. It can just return the WorkArea.propath value if it is not null.

3. Does getSearchPath() need the else branch anymore if the WorkArea.propath is no longer ever set to null?

In addition, please make the following change:

The SourceNameMapper needs to be updated to honor the current effective PROPATH instead of using the getLegacySearchPath() method. Please ask him any questions as needed.

#8 - 05/29/2013 06:19 AM - Greg Shah

Please see #1608 note 73 for two example testcases (test1.p and test2.p) which should run properly when you have fixed the SourceNameMapper issue. You can put these (and the foo.p and bar.p) into a unique subdirectory in testcases/uast/. I think that changing getLegacySearchPath() to getSearchPath() should probably be enough to solve the problem.

#9 - 05/29/2013 06:51 PM - Eugenie Lyzenko

- File evl_upd20130529a.zip added

New approach update has been uploaded. The changes:
1. New member for WorkArea in EnvironmentOps.java to separate full path from the changeable area.
2. Two entries in directory to specify PROPATH, searchpath for propathDefault and searchpath-override for WorkArea.propathOverride.
3. getSearchPath returns full "effective" value, initializing WorkArea.propath if it is not yet initialized.
4. setSearchPath changes both full and changeable value inside WorkArea.
5. In SourceNameMapper.java the getLegacySearchPath() replaced with getSearchPath().getValue() to obtain the current full path value.

The problem is the test1.p and test2.p can not find the foo.p and bar.p, although the propath is changing properly to "foo,fixed_part" and "bar,fixed_part" of "foo,bar". What is missing here? Propath "foo,bar" is counting from current directory or another starting point(inside *.jar classfiles)? Also these testcases should take into account the feature of the PROPATH change (replacing the changeable part at the head of the final result).

#10 - 05/30/2013 02:58 AM - Constantin Asofiei

The problem I think is the SourceNameMapper.propath field, which gets initialized by the SourceNameMapper.initMappingData call (and this call is performed only once per P2J Server). You need to remove the SourceNameMapper.propath field and replace its usage with proper calls from EnvironmentOps.

#11 - 05/30/2013 07:00 AM - Greg Shah

Code Review (0529a)

1. Please change the name of the "propathDefault" data member to "propathFixed".

2. For the initialization of propathFixed, please read "searchpath-fixed" from the directory. This will default to "".

3. For the initialization of wa.propathOverride, please read "searchpath" from the directory. This can default to "." (except for the concern in number 4 below).

4. In the past, we have had the "searchpath" default to ".,". This means that if one doesn't have a fixed portion, then the propathOverride will always return "." now AND that will be the effective propath too. In the past we returned ".,". Which is correct from a 4GL perspective?

5. In addition to Constantin's point about the propath member of SourceNameMapper never changing when it should, please also note 2 problems:

- Use toStringMessage() (instead of getValue()) on the return from getSearchPath() because we should not ever be using the getValue() under normal circumstances.
- SourceNameMapper is splitting the value using the legacy "pathsep" but it really needs to be using EnvironmentOps.PROPATH_SEPARATOR.

#12 - 05/30/2013 10:31 AM - Eugenie Lyzenko

> The problem I think is the SourceNameMapper.propath field, which gets initialized by the SourceNameMapper.initMappingData call (and this call is performed only once per P2J Server). You need to remove the SourceNameMapper.propath field and replace its usage with proper calls from EnvironmentOps.

The propath field removing from SourceNameMapper works fine. But the file p2j/convert/NameMappingWorker.java need to be modified as well in the following places:
1. propath = Configuration.getParameter("propath", ".:").split(pathSep);
2. overwriteMappingData(p2jmap, null, propath, pkgroot, proroot, fileSep, pathSep, caseSens);

As alternative solution we can introduce something like String propathCurrent = (getting current value from EnvironmentOps) inside the convertName() method of the SourceNameMapper.

What do you think?

#13 - 05/30/2013 11:18 AM - Constantin Asofiei

The propath field removing from SourceNameMapper works fine. But the file p2j/convert/NameMappingWorker.java need to be modified as well in the following places:
1. propath = Configuration.getParameter("propath", ".:").split(pathSep);
2. overwriteMappingData(p2jmap, null, propath, pkgroot, proroot, fileSep, pathSep, caseSens);

These two usages can be removed completely also. At some point, the SourceNameMapper was used by conversion rules to disambiguate the name in a RUN statement, but this usage was removed, as the disambiguation needs to be done at runtime.

#14 - 05/30/2013 11:33 AM - Greg Shah

The SourceNameMapper.convertJavaProg() is called from NameMappingWorker.classNameExists() method which is still in use (common-progress.rules):

```
      <function name = "classNameConflict">
         <parameter name="pkgName"   type = "java.lang.String" />
         <parameter name="className" type = "java.lang.String" />
         nmap.classNameExists(pkgName, className) or p2o.isDMOInterface(className)
      </function>
```

classNameConflict is called from control_flow.rules.

But all conversion usage of SourceNameMapper.convertName() through NameMappingWorker.resolveJavaClass() seems to be gone. But there still seems to be 1 remaining dependency.

For now, how about this:

1. Remove the propath member of NameMappingWorker.

2. In the constructor of NameMappingWorker, you could call EnvironmentOps.setSearchPath(Configuration.getParameter("propath", ".:"), true).

3. Remove the propath parameter to overwriteMappingData().

Constantin, what do you think?

#15 - 05/30/2013 11:44 AM - Constantin Asofiei

The NameMappingWorker should have no use of the propath, during conversion. Even if NameMappingWorker.resolveJavaClass() depends on the SourceNameMapper.convertName, the NameMappingWorker.resolveJavaClass() is not used at all in the conversion rules; so we can drop resolveJavaClass() too, together with the NameMappingWorker.propath member and the propath parameter from overwriteMappingData().

> But there still seems to be 1 remaining dependency.

What dependency do you refer too? The nmap.classNameExists call doesn't use the propath, and all the remaining usage of nmap in the conversion rules is related to building the name_map.xml: uses namp.restoreMappings, addMapping, isMapped, addSignature calls.

#16 - 05/30/2013 12:37 PM - Greg Shah

> > But there still seems to be 1 remaining dependency.
>
> What dependency do you refer too?

SourceNameMapper.convertJavaProg() is called from NameMappingWorker.classNameExists() method which is still in use.

For this reason, we must still call overrideMappingData() and allow the SourceNameMapper to be setup. I guess we can just pass a propath of null to the overrideMappingData() and we won't worry about it. The constructor won't have to call EnvironmentOps.setSearchPath().

#17 - 05/30/2013 12:42 PM - Constantin Asofiei

> SourceNameMapper.convertJavaProg() is called from NameMappingWorker.classNameExists() method which is still in use.

Yes, but the propath is not used by this call.

> For this reason, we must still call overrideMappingData() and allow the SourceNameMapper to be setup. I guess we can just pass a propath of null to the overrideMappingData() and we won't worry about it. The constructor won't have to call EnvironmentOps.setSearchPath().

I agree, but is best to just drop the propath parameter of overrideMappingData(), because overrideMappingData() would have no use for it, as SourceNameMapper.propath field is removed.

#18 - 05/30/2013 12:54 PM - Greg Shah

OK. Eugenie, please go forward as described.

#19 - 05/30/2013 01:12 PM - Eugenie Lyzenko

- File evl_upd20130530a.zip added

The new PROPATH fix has been uploaded:

1. Please change the name of the "propathDefault" data member to "propathFixed".

Done.

2. For the initialization of propathFixed, please read "searchpath-fixed" from the directory. This will default to "".

Done.

3. For the initialization of wa.propathOverride, please read "searchpath" from the directory. This can default to "." (except for the concern in number 4 below).

Done.

4. As far as I can see in both available 4GL systems we always have some fixed part of the PROPATH, it never == "". And in either case there is no PROPATH system var defined. This can mean the PROPATH system defaults are defined when the Progress application is started. If in result we have just "." it is valid. Even "." can be valid despite of it is not very "correct".

5. In addition to Constantin's point about the propath member of SourceNameMapper never changing when it should, please also note 2 problems:...

Done. Also I have changed to toStringMessage() in one additional place in EnvironmentOps instead of getValue().

6. SourceNameMapper() has been changed to dynamically get PROPATH, the member was removed.
7. convert/NameMappingWorker.java was cleaned from propath member too.

#20 - 05/30/2013 01:41 PM - Greg Shah

It looks good. The only thing missing is that the resolveJavaClass() method should be removed from NameMappingWorker, since it is no longer used AND it will not work anymore.

#21 - 05/30/2013 01:51 PM - Eugenie Lyzenko

- File evl_upd20130530b.zip added

> The only thing missing is that the resolveJavaClass() method should be removed from NameMappingWorker, since it is no longer used AND it will not work anymore.

OK. Done. New update has been uploaded.

#22 - 05/30/2013 01:56 PM - Greg Shah

OK, this is ready for testing. When the #1650 changes pass testing, this will be next. It will need both conversion and runtime regression testing.

#23 - 05/30/2013 06:51 PM - Eugenie Lyzenko

> OK, this is ready for testing. When the #1650 changes pass testing, this will be next. It will need both conversion and runtime regression testing.

The file evl_upd20130530b.zip is in staging/p2j directory. It is ready to be applied and reconvert(Not done yet).

#24 - 06/03/2013 01:11 PM - Eugenie Lyzenko

The update evl_upd20130530b.zip has passed the regression testing and committed in bzr as revision 10357.

#25 - 06/03/2013 01:58 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

#26 - 06/04/2013 09:19 AM - Constantin Asofiei

Greg, is it me or SourceNameMapper.java is missing the H019 changes, from 20130412?

#27 - 06/04/2013 10:34 AM - Greg Shah

No, you are right. That code is missing:

```
bzr diff -r 10356..10357 SourceNameMapper.java
=== modified file 'src/com/goldencode/p2j/util/SourceNameMapper.java'
--- src/com/goldencode/p2j/util/SourceNameMapper.java   2013-05-30 22:32:16 +0000
```

```
+++ src/com/goldencode/p2j/util/SourceNameMapper.java   2013-06-03 17:00:37 +0000
@@ -46,8 +46,9 @@
 ** 017 CA  20130209          Added dynamic EXTENT and EXTENT parameter support.
 ** 018 CS  20130304          Added support for checking if a class exists in a
 **                           certain package as result of conversion from file
-** 019 EVL 20130412          Fixing the regular expressions for windows runtime in
-**                           initMappingData() method to properly handle windows file separator.
+** 019 EVL 20130529          Replacing getLegacySearchPath() with getSearchPath() to obtain
+**                           full effective PROPATH variable value to reflect the changes in
+**                           PROPATH approach implementation.
 */

 package com.goldencode.p2j.util;
@@ -140,9 +141,6 @@
    /** Progress Java class name to procedure file name map table. */
    private static Map<String, ExternalProgram> j2pMap = null;

-   /** List of the directories in the PROPATH of the legacy system. */
-   private static String[] propath = null;
-
    /** Base path for all Java packages. */
    private static String pkgroot = null;

@@ -237,7 +235,17 @@
    public static String convertName(String progName)
    {
       initMappingData();
+
+      // get the original propath and split it into pieces
+      String searchpath = EnvironmentOps.getSearchPath().toStringMessage();
+
+      if (!caseSens)
+      {
+         searchpath = searchpath.toLowerCase();
+      }

+      String propath[] = searchpath.split(EnvironmentOps.PROPATH_SEPARATOR);
+
       // check the simple form of the name
       ExternalProgram extProg = p2jMap.get(canonicalize(progName));

@@ -415,7 +423,6 @@
     */
    public synchronized static void overwriteMappingData(Map      p2jMap,
                                                         Map      j2pMap,
-                                                        String[] propath,
                                                         String   pkgroot,
                                                         String   proroot,
                                                         String   fileSep,
@@ -424,7 +431,6 @@
    {
       SourceNameMapper.p2jMap    = p2jMap;
       SourceNameMapper.j2pMap    = j2pMap;
-      SourceNameMapper.propath   = propath;
       SourceNameMapper.pkgroot   = pkgroot;
       SourceNameMapper.proroot   = proroot;
       SourceNameMapper.fileSep   = fileSep;
@@ -819,19 +825,9 @@
          // this is only a conversion-time configuration value ("basepath")
          proroot   = "";

-         // get the original propath and split it into pieces
-         String searchpath = EnvironmentOps.getLegacySearchPath();
-
-         if (!caseSens)
-         {
-            searchpath = searchpath.toLowerCase();
-         }
-
-         propath = searchpath.split(pathSep);
-
          // build the fully qualified name of the name map XML file
          StringBuilder sb = new StringBuilder();
-         sb.append(pkgroot.replaceAll("\\.", StringHelper.fixupRegex(fileSep)));
+         sb.append(pkgroot.replaceAll("\\.", fileSep));
```

```
          if (!fileSep.equals(sb.substring(sb.length() - 1)))
          {
```

Here was the previous change that was removed:

```
bzr diff -r 10243..10356 SourceNameMapper.java
=== modified file 'src/com/goldencode/p2j/util/SourceNameMapper.java'
--- src/com/goldencode/p2j/util/SourceNameMapper.java   2013-03-05 19:42:29 +0000
+++ src/com/goldencode/p2j/util/SourceNameMapper.java   2013-05-30 22:32:16 +0000
@@ -46,6 +46,8 @@
 ** 017 CA  20130209          Added dynamic EXTENT and EXTENT parameter support.
 ** 018 CS  20130304          Added support for checking if a class exists in a
 **                           certain package as result of conversion from file
+** 019 EVL 20130412          Fixing the regular expressions for windows runtime in
+**                           initMappingData() method to properly handle windows file separator.
 */

 package com.goldencode.p2j.util;
@@ -829,7 +831,7 @@

          // build the fully qualified name of the name map XML file
          StringBuilder sb = new StringBuilder();
-         sb.append(pkgroot.replaceAll("\\.", fileSep));
+         sb.append(pkgroot.replaceAll("\\.", StringHelper.fixupRegex(fileSep)));

          if (!fileSep.equals(sb.substring(sb.length() - 1)))
          {
```

Eugenie: please fix this and let's review the final SourceNameMapper code here (upload your update zip into Redmine with the fixed SourceNameMapper).

#28 - 06/04/2013 11:02 AM - Eugenie Lyzenko

- File evl_upd20130604a.zip added

The updated file has been uploaded for you to review.

#29 - 06/04/2013 11:06 AM - Greg Shah

It is OK. I think we can avoid testing this one. Check it in and distribute it.

#30 - 06/04/2013 11:29 AM - Eugenie Lyzenko

> It is OK. I think we can avoid testing this one. Check it in and distribute it.

OK, agreed. Moreover this change did pass testing with the process launching update. Committed in bzr as 10358.

#31 - 06/26/2013 08:00 PM - Eugenie Lyzenko

There is one javadoc issue found here which does not touch the functionality and does not require re-testing. The propath parameter should be removed from javadoc comments to the overwriteMappingData method line 413 of the SourceNameMapper.java. Because it is not used anymore in method and generates the javadoc Warning.

I've noted this working on another task that generates javadocs.

#32 - 06/27/2013 09:13 AM - Greg Shah

You can go ahead and fix the javadoc issue in that one file. Do NOT add a history entry for that change. Then you may check it into bazaar (and send it out as an update). No testing is needed.

#33 - 06/27/2013 10:15 AM - Eugenie Lyzenko

- File evl_upd20130627a.zip added

Fixed in appended update. Committed in bzr as 10360.

#34 - 11/16/2016 11:43 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

evl_upd20130425a.zip	10.6 KB	04/25/2013	Eugenie Lyzenko
evl_upd20130522a.zip	10.7 KB	05/22/2013	Eugenie Lyzenko
evl_upd20130522b.zip	10.7 KB	05/22/2013	Eugenie Lyzenko
evl_upd20130529a.zip	23 KB	05/29/2013	Eugenie Lyzenko
evl_upd20130530a.zip	26.4 KB	05/30/2013	Eugenie Lyzenko
evl_upd20130530b.zip	26.1 KB	05/30/2013	Eugenie Lyzenko
evl_upd20130604a.zip	12.8 KB	06/04/2013	Eugenie Lyzenko
evl_upd20130627a.zip	12.8 KB	06/27/2013	Eugenie Lyzenko