Base Language - Feature #2385

implement environment access on GUI interfaces.

09/04/2014 02:06 AM - Marius Gligor

Status:	Closed	**Start date:**	09/04/2014
Priority:	Normal	**Due date:**	
Assignee:	Marius Gligor	**% Done:**	100%
Category:		**Estimated time:**	80.00 hours
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #1649: implement Windows environment/regis...	**Closed**	**01/15/2013**	**05/17/2013**
Related to User Interface - Feature #1793: improve color support	**Closed**		

History

#1 - 09/04/2014 02:47 AM - Marius Gligor

GUI clients are using environment statements LOAD USE and UNLOAD to load colors, fonts and potentially other resources.
LOAD USE and UNLOAD statements are working on Windows only using Windows registry or INI files to store data according to P4GL (OE) manual.
Current P2J implementation is P4GL (OE) compliant. It works only on Windows and use Windows registry or INI files located on client side to store data.
Unlike P4GL (OE) GUI release which is designed to work on Windows OS only the P2J GUI implementation is intended to be a cross platform implementation.
In order to have a cross platform implementation only INI files can be used because they are text files and could be accessed from any platform.

#2 - 09/04/2014 03:05 AM - Marius Gligor

- File progress.ini added

- Status changed from New to WIP

Current implementation for COLOR-TABLE and FONT-TABLE statements load the default environment from server directory.
Color or font changes in default environment should be saved also on server directory.
The attached progress.ini file is the default environment used by OE 10.2b when start.
Custom environments should follow the same structure and format.

#3 - 09/04/2014 04:02 AM - Marius Gligor

Ovidiu,

So far I did no changes on current implementation for Windows environment access.
I only added getLegacyPlatform() method which is used to select color schema Windows or Linux/Unix for ChUI clients.
I created a new Redmine task #2385 and I added you on the watchers list. Please follow conversions history for this task.

Thanks
Marius

On 04.09.2014 00:43, Greg Shah wrote:

> Ovidiu,

2. At this moment all EnvironmentOps are restricted to Win32 OS types. I saw in your mag_upd20140903a.zip that you already implemented a getLegacyPlatform API in EnvironmentOps that should allow accessing stanza-based INI files on Linux. Do you intend to extend support here ?

My thinking was that we can just eliminate the WIN32 checking in the case of stanza-based INI files. We really shouldn't care what platform we are on in that case, since they are simple text files. I prefer not to depend on configuration (getLegacyPlatform()) unless it is really needed.

Thanks,
Greg

On 09/03/2014 04:52 PM, Ovidiu Maxiniuc wrote:

Marius,

I am working on a support task that needs a feature that would allow P2J to retrieve some environment values (as you manage colors and fonts). I am interested in 2 issues:

1. To not duplicate the implementation, please let me know if you already added support for loading an INI file at the startup (emulate the -ininame command line parameter, see the OpenEdge® Deployment: Startup Command and Parameter Reference).

2. At this moment all EnvironmentOps are restricted to Win32 OS types. I saw in your mag_upd20140903a.zip that you already implemented a getLegacyPlatform API in EnvironmentOps that should allow accessing stanza-based INI files on Linux. Do you intend to extend support here ?

Regards,
Ovidiu

#4 - 09/05/2014 09:36 AM - Marius Gligor

I did some tests using LOAD USE and UNLOAD statements to manage user environment on both OE and P2J.
Current P2J implementation works well as I expected. I found only 2 minor differences:

1. The P2J implementation is case sensitive while OE (Windows OS) is non case sensitive!
For example color section in INI files could be [Color] or [color]. In OE works on both case but on P4J depends on string case specified on Java call:
EnvironmentOps.getKeyValue("Colors", "color0");
or
EnvironmentOps.getKeyValue("colors", "color0");
Since we are looking to use INI files in both Linux OS and Windows OS the name of the INI files should be case sensitive but the section and key

parameters should be no case sensitive.

2. In OE when a statement like LOAD environment is executed if no registry key is found the environment parameter is interpreted as an INI file. If file exists is loaded.
If no INI file was found an error message The LOAD of <environment> failed (4450) is displayed.
P2J lookup only in registry for key and if not found return an unknown (?) value.

3. In OE manuals I found something like this: INI files shared by multiple user should be stored on a location where each interested users have access.
We could follow the same behaviour in P2J using the existing implementation and do only minimal changes to allow INI files to work also on Linux (GUI ?).
If we have to move INI files location somewhere on the sever side we have to do deep changes in current implementation.

Also when LOAD USE and UNLOAD statements are executed we have to notify both COLOR-TABLE and FONT-TABLE implementations in order to do the appropriate actions:
- LOAD environment - load colors and fonts.
- USE environment - select current environment (colors and fonts.
- UNLOAD environment - discard current colors and fonts if it's possible (no window references) and select the default environment for colors and fonts.

#5 - 09/05/2014 10:06 AM - Greg Shah

Interesting findings.

> Since we are looking to use INI files in both Linux OS and Windows OS the name of the INI files should be case sensitive

Actually, for stanza INIs files I think we should implement case-insensitive filename support. The reason: we are migrating from a system which maintains case in the file system name but which accepts filename specifications that are case-insensitive. This means that the actual file in the file system may be ProGResS.iNi but progress.ini is specified in the code. In order to reduce the friction of making this work, we can honor the behavior of Windows in this regard.

We already do this today in our general file system support. Please look at FileSystemDaemon.search() which shows an example.

> but the section and key parameters should be no case sensitive.

Absolutely correct.

Does this only apply to stanza INIs or does this behavior extend to the registry too?

> If no INI file was found an error message The LOAD of <environment> failed (4450) is displayed.
> P2J lookup only in registry for key and if not found return an unknown (?) value.

Yes, please fix this.

> We could follow the same behaviour in P2J using the existing implementation and do only minimal changes to allow INI files to work also on Linux (GUI ?).
> If we have to move INI files location somewhere on the sever side we have to do deep changes in current implementation.

If I understand correctly, the place where we get into the most trouble is if the application uses PUT-KEY-VALUE (which translates into EnvironmentAccessor.setKeyValue() in P2J), right? Otherwise the INI is read-only and the only "trick" is to read the INI from the server instead of the client. Preferably in this case, we would read it from the application jar file and by enabling this we eliminate some deployment complexity.

I can accept that this approach would only be allowed for read-only usage. Any writable access would have to be like normal from the client. I understand this is still a bit tricky, but I hope we might find a less intrusive way to support this idea.

What do you think?

> Also when LOAD USE and UNLOAD statements are executed we have to notify both COLOR-TABLE and FONT-TABLE implementations in order to do the appropriate actions:

This sounds like we need an abstraction interface for notifications/callbacks that is implemented by our *-TABLE classes.

#6 - 09/05/2014 10:56 AM - Marius Gligor

Indeed the big problem with INI files on P2J is the deployment location. Reading or writing INI files on server side from clients side is not a blocked issue.

1. On read the LOAD is the single statement which require client access for reading, USE and UNLOAD doesn't. A good program should load a custom environment only once.
As a result LOAD statement should be less used than normal statements which implements the business.

2, On write using PUT-KEY-VALUE should be used only at least once. I don't see at this moment any reasons for an application to save repeatedly colors or fonts.
PUT-KEY-VALUE statement looks likely an "administrative" command used on small scripts to create and update environments.

#7 - 09/05/2014 11:05 AM - Greg Shah

I agree with your thoughts. If this read-only approach is still going to take a significant amount of time/changes, then we can defer it until much later. I will let you make the decision on that. If we defer it, we will need to add a task to the "Deployment and Management Improvements" milestone.

#8 - 09/10/2014 11:09 AM - Marius Gligor

- File mag_upd20140910a.zip added

Here are my changes that I did so far on this task. Please do a code review on these changes and let me know the results.
A short description of changes that I did:

1. Section name and key name are now case insensitive for INI files.

2. I added a new client parameter environment:read:only. When this parameter is true allow the load of INI files as resources from jar files. Default value is false and INI files are read from file system.

3. I did also other changes which allow GUI clients to use INI files as environments on both Windows and Linux OS. My changes should not affect the initial behaviour of environment API designed to work only on Windows OS.

4. I added color table notification on execution of USE environment statement. The current environment name is stored and used later to load colors from loaded environment. Color table is lazy loaded when is first used. Font table does not require this notification because he manage the font table on client side and always call EnvironmentDeamon.getEnvironmentName() in order to find the current environment name.

5. I added notification on execution of UNLOAD environment for both color and font tables. The notification is used to discard cached environments. A P4G application might load and environment named foo, then unload and later load another environment named also foo from another location that's why environments must be discarded on unload.

6. I changed the implementation of EnvironmentDeamon.unlod() method because the initial implementation always discard current environment instead the specified one.

7. Notification on execution of LOAD environment statement is not necessary as long as environments are lazy loaded on both color and font tables.

8. Another question. Is any event or call on server side when a P4 converted application is started? If yes this will be a good point to initialize color and font tables.
Let me explain using a simple converted program like:

```

   LOAD "test" "INI".
   USE "test".
   COLOR-TABLE:NUM-ENTRIES.
   UNLOAD "test".
```

converted as:

```
EnvironmentOps.load("test", "INI");
EnvironmentOps.use("test");
ColorTable.getNumEntries();
EnvironmentOps.unload("test");
```

Color table is initialized when ColorTable.getNumEntries(); statement is executed. At this moment the current environment is "test" as a result of EnvironmentOps.use("test"); statement execution.
On my implementation I always force a read of default color table from server directory followed by the loading of custom environment whenever the WorkArea is created (see WorkArea constructor on ColorTable class).
We could simplify the initialization code if it's possible to do that before any converted statement is executed.

#9 - 09/10/2014 05:20 PM - Greg Shah

Code Review 0910a

1. Doesn't StanzaIni always get used on the client side? If so, then the jar file reading is going to fail since the application jar files don't ever exist there. The idea of reading from the jar is a server-side idea to reduce deployment problems.

2. Do we need the extra call to ClientExports.unload() when we are already calling down to the client for EnvironmentDaemon.unload() processing?

3. I think EnvironmentOps.isGui() should return !LogicalTerminal.isChui() instead of LogicalTerminal.isChui().

4. ColorRgb is missing a history entry and the javadoc for the new toString() method is incomplete.

5. ColorTable.ColorPair.toString() is missing javadoc.

6. In regard to your question Is any event or call on server side when a P4 converted application is started?:

I'm not sure whether this will actually simplify things. Please see the "P2J Developer Guide" in the "Runtime Hooks and Plug-Ins" chapter. We have both server-level hooks ("Server Initialization and Termination Hooks") and session-level hooks ("Session Initialization and Termination Hooks"). Although the examples shown there show how to configure these in the directory, we would not want to require extra configuration for this, but we would just register these at server startup in the bootstrap code. But as I say, I don't think this actually is a good idea. Take a look and see what you think.

#10 - 09/11/2014 09:26 AM - Marius Gligor

- File mag_upd20140911a.zip added

1. I fixed code review issues from 0910a.

2. I integrated the changes from task #2366 related to override default environment (-ininame' command-line parameter).

3. Regarding the loading of INI files as resources from jar files. I changed the code to load the INI file from server using a server exported API similar with loading images from server. INI files are small files. For example progress.ini file is only 5K in size and are usually loaded only once so no big impacts on performances are expected.

Here I have 2 questions:

a) The read only flag which change the loading rule from client file system to server jar as resource should be a global setting on server directory or a client parameter like currently is already implemented?

b) Should apply the same loading rule when override the default environment too (task #2366)?
In other words when -ininame' command-line parameter is used the INI file should be loaded always from client file system or could be loaded also from server jar file if read only option is set?.

#11 - 09/11/2014 09:51 AM - Marius Gligor

- Status changed from WIP to Review

#12 - 09/11/2014 10:49 AM - Greg Shah

Code Review 0911a

The changes are very good.

> a) The read only flag which change the loading rule from client file system to server jar as resource should be a global setting on server directory or a client parameter like currently is already implemented?

The directory is best. Since it is only needed in the EnvironmentDaemon and StanzaIni classes, perhaps it is best to read it inside the EnvironmentDaemon constructor. The flag can then be passed to the StanzaIni constructor whenever that is invoked. ThinClient shouldn't be involved.

The lookup should allow for per-account, per-group, per-server or default settings in the directory.

> b) Should apply the same loading rule when override the default environment too (task #2366)?
> In other words when -ininame' command-line parameter is used the INI file should be loaded always from client file system or could be loaded also from server jar file if read only option is set?.

Yes, please apply the same server-load rule for the -ininame case.

#13 - 09/11/2014 11:35 AM - Marius Gligor

- File mag_upd20140911b.zip added

Fixed 0911a code review. The read only flag is global and read from server node readOnlyEnvironments which should have a boolean value.
The read only rule is applied also when -ininame command line option is in use.
Default value is false Would you prefer another directory node name?

#14 - 09/11/2014 12:41 PM - Greg Shah

Code Review 0911b

Instead of adding a ServerExports method to access the read-only flag, please use the Directory interface (access it like ClientCore with DirectoryManager.getInstance() and then call getString(Directory.ID_RELATIVE, ...).

> Would you prefer another directory node name?

No, your choice is good.

#15 - 09/11/2014 01:17 PM - Marius Gligor

I tried to use DirectoryManager.getInstance() but it doesn't work.
I'm using the following code:

```
Directory remoteDir = DirectoryManager.getInstance();
if (remoteDir != null)
{
   result = remoteDir.getBoolean(Directory.ID_RELATIVE, "readOnlyEnvironments", false);
}
```

and in server directory I have:

```
   <node class="container" name="server">
      <node class="container" name="default">
        <node class="boolean" name="readOnlyEnvironments">
          <node-attribute name="value" value="TRUE"/>
        </node>
```

but the result is always false (default) instead true !!!
What's wrong here?

#16 - 09/11/2014 01:21 PM - Marius Gligor

It works only for DirScope.ACCOUNTS

#17 - 09/11/2014 01:24 PM - Greg Shah

I think you need to have it located in /server/default/runtime/default/ instead of /server/default/.

#18 - 09/11/2014 01:30 PM - Marius Gligor

Yes you are right it works in /server/default/runtime/default/
But on note 12 you said: The lookup should allow for per-account, per-group, per-server or default settings in the directory.

#19 - 09/11/2014 01:33 PM - Marius Gligor

- File mag_upd20140911c.zip added

done.

#20 - 09/11/2014 01:34 PM - Constantin Asofiei

Marius Gligor wrote:

> Yes you are right it works in /server/default/runtime/default/
> But on note 12 you said: The lookup should allow for per-account, per-group, per-server or default settings in the directory.

You need to use Utils.getDirectoryNode APIs to be able to automatically search per-account/group/server/default. But the client-side doesn't have access to these APIs, only the server side can use Utils.getDirectoryNode.

#21 - 09/11/2014 01:36 PM - Greg Shah

The DirectoryServer class (which backs the DirectoryManager.getInstance() on the server side) uses Utils.getDirectoryNode*() methods to do the lookup. Those methods use the following hierarchy for lookups:

1. /server/<server_id>/runtime/<account_or_group_id>/
2. /server/<server_id>/runtime/default/
3. /server/default/runtime/<account_or_group_id>/
4. /server/default/runtime/default/

This is what I meant. The lookup can be specified for a specific server + group/user OR for a specific server OR for all servers with a group/user OR a default for all servers.

The remote directory approach will be suitable.

#22 - 09/11/2014 01:37 PM - Greg Shah

In other words, when you call DirectoryManager.getInstance() on the client, you will get a remote proxy to a server-side DirectoryServer instance that uses the Utils.getDirectoryNode*() APIs that we need to access. So your approach will be fine.

#23 - 09/11/2014 01:39 PM - Marius Gligor

OK. I understood thanks.

#24 - 09/11/2014 01:58 PM - Greg Shah

Code Review 0911c

The changes look good. Please get these runtime regression tested.

#25 - 09/12/2014 10:54 AM - Marius Gligor

- % Done changed from 0 to 100

0911c passed regression tests. Committed revision 10608.

#26 - 09/12/2014 12:54 PM - Greg Shah

- Status changed from Review to Closed

#27 - 11/16/2016 12:13 PM - Greg Shah

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

Files

progress.ini	5.62 KB	09/04/2014	Marius Gligor
mag_upd20140910a.zip	249 KB	09/10/2014	Marius Gligor
mag_upd20140911a.zip	245 KB	09/11/2014	Marius Gligor
mag_upd20140911b.zip	245 KB	09/11/2014	Marius Gligor
mag_upd20140911c.zip	245 KB	09/11/2014	Marius Gligor