Base Language - Bug #2684

XML file processing needs to be implemented on the client side instead of on the P2J server

09/04/2015 04:50 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% 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

History

#1 - 09/04/2015 04:52 PM - Greg Shah

The X-Document handle type has two methods (LOAD() and SAVE()) that allow XML to be read from and written to files. Currently, the implementation is incorrect, in that it operates using the P2J server's filesystem directly. As with how we redirect streams processing to the client, we must also implement remote/client-side LOAD() and SAVE().

#2 - 11/19/2015 05:39 PM - Eric Faulhaber

- Subject changed from XML file processing needs to me implemented on the client side instead of on the P2J server to XML file processing needs to be implemented on the client side instead of on the P2J server

- Status changed from New to WIP

- Assignee set to Eric Faulhaber

- Target version set to Milestone 12

Greg Shah wrote:

> As with how we redirect streams processing to the client, we must also implement remote/client-side LOAD() and SAVE().

Is this meant to imply just that the architecture should be similar, or is it reasonable for me to re-use the existing P2J stream infrastructure directly (e.g., StreamDaemon, StreamBuilder, LowLevelStream, etc.) for this implementation?

I found the test case dom_xml_test.p, which seems like a good place to start. If you know of any others that might be useful (e.g., for non-obvious use cases), please let me know.

#3 - 11/19/2015 05:52 PM - Greg Shah

> Is this meant to imply just that the architecture should be similar

Yes, but maybe not "just" that.

> is it reasonable for me to re-use the existing P2J stream infrastructure directly (e.g., StreamDaemon, StreamBuilder, LowLevelStream, etc.) for this implementation?

I'm not sure. The current implementation of LOAD() is in XDocument.load() which for the "file" case uses this:

```
            xdoc = XmlHelper.parse(fname,
                                   validate.getValue(),
                                   null,
                                   isSuppressNamespaceProcessing);
```

The key here is that we must:

- handle any client-side file searching that may occur (the 4GL probably loads relative filenames from PROPATH)
- read the bytes from the client but parse the DOM on the server

For SAVE(), XDocument.save() uses XmlHelper.write(xdoc, (String)target); and we have a similar need to redirect this to the client.

I don't think we want to have 4GL compatibility dependencies in the XmlHelper.

#4 - 11/19/2015 06:09 PM - Eric Faulhaber

Greg Shah wrote:

> The key here is that we must:
>
> - handle any client-side file searching that may occur (the 4GL probably loads relative filenames from PROPATH)
> - read the bytes from the client but parse the DOM on the server

OK, so it sounds like I need to write some more test cases to figure out the file searching behavior. You note the DOM. Is that to say there aren't implications for the SAX API? If that question doesn't make sense, sorry; I'm not familiar with either API on the Progress side yet.

I'm also not familiar with the existing P2J stream classes yet, but my impression was that somewhere in that hierarchy we already handle file searching based on PROPATH and streaming bytes from the client to the server, no? I don't want to reinvent the wheel if possible, though I understand I have to determine whether the searching works in a similar way.

> For SAVE(), XDocument.save() uses XmlHelper.write(xdoc, (String)target); and we have a similar need to redirect this to the client.

Again, the existing P2J class hierarchy must handle streaming bytes in that direction already.

> I don't think we want to have 4GL compatibility dependencies in the XmlHelper.

Agreed, that doesn't make sense.

It seems that Progress would have layered this implementation on top of its older stream stuff, so I'm hopeful we can do the same. Maybe that's naïve...

#5 - 11/20/2015 08:25 AM - Greg Shah

> You note the DOM. Is that to say there aren't implications for the SAX API?

It is a good question. Where there is anything that reads from or writes to the filesystem, then it will be affected too. A quick look at the SAX API shows that SAX-READER:SET-INPUT-SOURCE(), SAX-WRITER:SET-OUTPUT-DESTINATION(), SAX-READER:*SCHEMA-LOCATION, SAX-READER:SCHEMA-PATH all have client-side implications.

There may be other items (e.g. DTD support) in the DOM that also need to be addressed.

> I'm also not familiar with the existing P2J stream classes yet, but my impression was that somewhere in that hierarchy we already handle file searching based on PROPATH and streaming bytes from the client to the server, no?

Certainly. Look at EnvironmentOps.searchPath(). You'll have to use this to turn any relative filename into an absolute filename, before you can read the file.

> > For SAVE, XDocument.save() uses XmlHelper.write(xdoc, (String)target); and we have a similar need to redirect this to the client.
>
> Again, the existing P2J class hierarchy must handle streaming bytes in that direction already.

Those stream classes all have 4GL semantics deeply ingrained. They will raise conditions and possibly do other things you don't want.

However, we do have InputStreamWrapper and OutputStreamWrapper that can wrap the Progress-compatible stream instances and expose them as an InputStream or OutputStream on the server side.

You should duplicate the way we create FileStream instances in converted code. See StreamFactory.openFileStream().

> It seems that Progress would have layered this implementation on top of its older stream stuff, so I'm hopeful we can do the same. Maybe that's naïve...

It's good to have dreams. :)

#6 - 03/30/2016 04:08 PM - Greg Shah

- Assignee changed from Eric Faulhaber to Ovidiu Maxiniuc

- Start date deleted (09/04/2015)

To be clear, I think the LOAD and SAVE implementations should be possible using InputStreamWrapper and OutputStreamWrapper on the server side. By leveraging a RemoteStream instance, these can be used to read/write files on the client. This should be reasonably easy to implement.

The SAX items in note 5 need more investigation to determine the requirements.

#7 - 04/02/2016 07:16 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> To be clear, I think the LOAD and SAVE implementations should be possible using InputStreamWrapper and OutputStreamWrapper on the server side. By leveraging a RemoteStream instance, these can be used to read/write files on the client. This should be reasonably easy to implement.

The update for moving I/O operations from server to client were committed to task branch 2181a as revno 10999.

> The SAX items in note 5 need more investigation to determine the requirements.

I checked the code for sax classes and found that:

- SAX-READER:SET-INPUT-SOURCE() - this works as expected already, the SaxReaderImpl.FileInputStreamWrapper constructs a remote stream to access client-side FS.
- SAX-WRITER:SET-OUTPUT-DESTINATION() same as previous. The wrapper class is SaxWriterImpl.FileOutputStreamWrapper.
- SAX-READER:SCHEMA-LOCATION implemented just as a property, with a getter and a setter.
- SAX-READER:SCHEMA-PATH while each file from the content of this property is checked for existence against the client/remote FS using SaxReaderImpl.FileInputStreamWrapper, but it is not used otherwise. The property can be accessed with a getter and a setter.

So my conclusion is that they are correctly implemented from the point of where the files are accessed.

#8 - 04/02/2016 07:36 AM - Greg Shah

Great!

Is there anything left to do on this task?

#9 - 04/04/2016 04:37 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> Great!
>
> Is there anything left to do on this task?

I have committed the last changes related to improvement of XML IO and parsing error handling as revision 11005 of 2181a.

#10 - 04/06/2016 10:13 AM - Ovidiu Maxiniuc

The final update for this task was committed to trunk with task branch 2181a as revision 10995.
Task can be closed.

#11 - 04/06/2016 10:13 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

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

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