Conversion Tools - Feature #3696

implement Progress AST anti-parser

08/23/2018 07:56 AM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Greg Shah	**% Done:**	50%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Conversion Tools - Feature #2110: improve orphan comment processing	**Closed**
Related to Conversion Tools - Feature #3695: instrument 4GL code for automate...	**New**

History

#1 - 08/23/2018 07:57 AM - Greg Shah

- Related to Feature #2110: improve orphan comment processing added

#2 - 08/23/2018 08:11 AM - Greg Shah

The objective is to output a Progress AST as a fully correct 4GL source file. The first pass at this will ignore include files. This means it will only output the equivalent of the .cache (fully preprocessed file).

This task can only be worked after the issues described in #2110 are resolved. The reason is that the Progress parser in trunk drops many tokens from the lexer which means those tokens don't appear in the AST. To keep the integrity of the original source code, we must be able to anti-parse every character of the original file back out.

If an unmodified Progress AST (the AST just after parsing/conversion front end completes) is anti-parsed, the result should be byte-for-byte identical to the .cache file.

I anticipate 2 modes of output:

- Original Source Mode. This mode simply outputs everything (including all shadow nodes with whitespace, comments and the other dropped tokens) in the proper order. These shadow nodes (when they exist) define the output **between** AST nodes.
- Formatted Tree Mode. The tree structure (and context) is used to output the AST nodes in a standardized format. Shadow nodes are ignored. This would be used for nodes in the tree (e.g. templates and other transformations) that don't have shadow nodes to define the output between nodes. Instead, heuristics must be put in place to define this. This mode will be very similar to brew.xml except the heuristics and processing will be aware of the 4GL syntax.

Both modes will ignore artificial nodes (line number 0 and column number 0).

There is at least 1 case of AST nodes being modified during parsing which needs to be reversed (if a STRING node has an annotation "unquoted-text" which is true, then the quotes must not be output).

As part of this task we will also create a ProgressTransformDriver that will:

- Optionally run the conversion front end to get ASTs.
- Run one or more TRPL pipelines for transformation of the ASTs.
- Anti-parse the ASTs.

#3 - 08/23/2018 08:11 AM - Greg Shah

- Related to Feature #3695: instrument 4GL code for automated call-graph hint generation added

#4 - 08/31/2018 11:10 AM - Greg Shah

> There is at least 1 case of AST nodes being modified during parsing which needs to be reversed (if a STRING node has an annotation "unquoted-text" which is true, then the quotes must not be output).

I've eliminated the "unquoted-text" annotation. Instead, I've taken a more generic approach. If we are changing the type and the text of a node to make it different from the token, then we store the original token type in an "oldtype" annotation and the original token text in an "original-text" annotation. This allows us to detect the fact that the type and text have changed and we can put the original "back together" for anti-parsing.

#5 - 09/16/2018 08:05 PM - Greg Shah

- Status changed from New to WIP

- Assignee set to Greg Shah

Branch 2110a was merged to trunk as revision 11284.

Branch 3696a was created from trunk revision 11284.

#6 - 10/08/2018 12:16 PM - Greg Shah

3696a has been rebased from trunk 11286. The latest revision is 11300.

#7 - 10/23/2018 01:17 PM - Greg Shah

In revision 11301, I refactored ConversionDriver to move most functionality into a common abstract base class TransformDriver.

#8 - 10/23/2018 01:20 PM - Greg Shah

Rebased 3696a from trunk 11287. The latest revision is now 11302.

#9 - 10/23/2018 04:26 PM - Greg Shah

In revision 11303, I added the ProgressTransformDriver and a sample rule-set that eliminates field-level abbreviations. It works as expected however there is no anti-parser yet. I'll add that next.

#10 - 10/23/2018 09:02 PM - Greg Shah

In revision 11304, I've got a first pass at the anti-parser. It is close but there are differences in whitesace and in some nodes not being output. It is 99% correct but that is not good enough.

Strangely, the pattern engine views are not working as expected. The result is more messy because I have to iterate manually through a sorted list.

#11 - 10/24/2018 02:58 PM - Greg Shah

Revision 11305 has a fully working 'simple' anti-parser plus fixes for the example rule-set. The parser and AST shadow node creation had latent bugs that I fixed.

#12 - 10/31/2018 03:39 PM - Greg Shah

Rebased 3696a from trunk 11288. The latest revision is now 11307.

#13 - 11/23/2018 11:28 AM - Greg Shah

I want to rebase 3696a from the latest trunk. I will rebase in 15 minutes unless there are objections.

#14 - 11/23/2018 12:42 PM - Constantin Asofiei

n/a

#15 - 11/23/2018 12:55 PM - Greg Shah

3696a was rebased from trunk 11291. The latest revision is 11338.

#16 - 12/06/2018 05:41 AM - Constantin Asofiei

3696a was rebased from trunk 11293 and passed testing.

Revision 11351 contains some changes related to GET-DB-CLIENT conflicts.

Will release shortly.

#17 - 12/06/2018 07:22 AM - Constantin Asofiei

3696a was merged to trunk rev 11294 and archived.
Passed conversion and runtime ChUI application testing.
Passed conversion and/or report_server for all major projects.

Contains changes related to tasks #3696, #3750, #3772, #3773, #3793, #3818 by ECF, GES and CA. Following list contains most important changes:
1. Fully working 'simple' anti-parser (#3696)
2. Callgraph improvements (#3818):
- Basic support for multiple classes with the same qualified names
- Fixed some callgraph processing issues related to OO.
- Fixed rootlist file calculation when folders are used.
3. OO fixes and improvements (at parser level)
- Complete rework of pre_scan_class to use a cut down parser approach instead of a full parser with fake symbol resolution. This is believed to be a closer representation of how the 4GL does things and it resolves a very large number of pre_scan_class failures as found in a customer app that is mostly OO.
- Removed class_map.xml from SymbolResolver.java - now PROPATH is used. Interfaces have access to Progress.Lang.Object methods. Refs #3773
- Fixed ClassDefinition.lookupHierarchy - must look both runtime and static members, if searching in non-static mode. Allow SymbolResolver to try the current class when looking for a member, before falling back to a mock class definition.
4. performance improvements (#3793 and others)
- stream JSON responses from server in chunks to reduce heap spikes (for Analytics)
- reduced schema dictionary heap use
5. Analytics
- Added CSV export for the detail report in FWD Analytics code/schema reports
- Added CSV export for the code/source reports in FWD Analytics
6. lots of other misc fixes and improvements, related to parsing and report generation for large customer application(s).

#18 - 12/06/2018 08:20 AM - Greg Shah

- % Done changed from 0 to 50

The items that remain in this task are the following:

- Provide more examples and documentation.
- Phase 2
    - Need to automatically calculate the line/column numbers of changes to the tree. Some of our processing depends upon this to work. For example, artificial nodes don't get anti-parsed AND some processing depends on calculating which nodes are left/right of each other so the relative line numbers must be consistent.
    - Provide a default formatting/whitespace output for nodes that were created via program (TRPL rules) instead of being read from the input source code. In the current parser, in order to have formatted output, hidden nodes with the whitespace would have to be inserted in between nodes and linked properly. The core issue here is that simple heuristics may not be enough since there is so much different syntax in the language.
    - Add shadow node support to the TemplateWorker for loading and grafting. This will also require changes in XmlFilePlugin.
    - Add more tools (in ProgressPatternWorker) for insert/delete/move of nodes and shadow nodes (keeping all the shadow node linkages intact).
    - Provide tools for the TRPL user to control/edit/specify the formatting, without hard coding the whitespace as hidden nodes.
    - Create a tool to help apply changes to the original files by writing the changes as patches that could be applied via diff. In combination with the current output, this could make it unnecessary to implement phase 3.
- Phase 3
    - Implement an option to flow edits back to the original source files (even include files). This is not guaranteed to work in all cases since changes may stretch across the boundaries of preprocessor expansions, conditional preprocessor directives and nested includes. However, it is also possible that many or most changes could be calculated safely.