Conversion Tools - Feature #1520

Feature # 1511 (New): Reporting v3

design gap analysis approach

09/10/2012 12:24 PM - Greg Shah

Status:	Closed		**Start date:**	09/10/2012
Priority:	Normal		**Due date:**	
Assignee:	Greg Shah		**% Done:**	100%
Category:			**Estimated time:**	8.00 hours
Target version:				
billable:	No		**version:**	
vendor_id:	GCD			

Description

History

#1 - 09/26/2012 08:03 AM - Greg Shah

- Target version set to Milestone 2

- Estimated time set to 8.00

#2 - 10/16/2012 02:40 PM - Greg Shah

- Status changed from New to WIP

- Assignee set to Greg Shah

#3 - 10/16/2012 05:01 PM - Greg Shah

1. Should this be inclusive or exclusive?

- inclusive means that we define exactly what is supported and by default everything is not supported
- exclusive means that we define exactly what is not supported and by default everything is supported
- inclusive is safer (e.g. parser changes don't require changes to the logic for what is supported) BUT there will likely be more logic required to identify everything affirmatively
- exclusive is less work but potentially more error prone

We will be INCLUSIVE.

2. Should we use "marking" or should we just implement reports that calculate the gaps?

- marking is the idea that we walk all nodes of every tree and leave behind an annotation to mark each node as supported or not (based on being inclusive or exclusive above)
- marking could be implemented in the parser, which in some ways makes things easy because one can directly map marking rules to all possible syntax that is supported BUT this approach would make the parser MUCH bigger, more complicated and more fragile (some of our AST rewriting tricks might be a problem)
- the other marking approach would be to implement a set of TRPL rules to mark things, this is safer but is probably more verbose
- using TRPL gives the full power and flexibility to the rules
- a big advantage of marking is that the reporting processing can be very generic later (at the time we have found a match, just calculate the % supported metric for the associated subtree, where the % can be 0% - totally unsupported, 100% - totally supported, or something in between - partially supported)

- the reporting approach is the alternative to marking, the idea is to create custom reports to make lists of things that are supported or not
- one might try to implement this with as much data driven logic as possible (lists or static database tables that encode functions, attributes... that are supported); note that this can still be done even with marking
- these reports would probably be very complex and would not be able to calculate metrics like % supported since they would simply be lists of unsupported things
- this might be less work but it is a poorer result and is potentially error prone

We will use TRPL based MARKING.

#4 - 10/16/2012 05:01 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

#5 - 11/21/2012 09:04 AM - Greg Shah

- Target version changed from Milestone 2 to 25

#6 - 03/20/2014 08:50 AM - Greg Shah

We want to make the marking as data driven as possible. This will allow rapid enhancement of the facilities without writing code.

Instead of a traditional static database approach, we will generate our database when needed by scanning the P2J code for Java annotations. We already provide annotations for identifying attributes and methods. We would extend this idea to include annotations for the level of support:

- UNSUPPORTED
- PARTIAL (includes text to describe what is missing)
- FULL (includes optional text describing any permanent limitations)

These levels can be independently reported for both conversion and runtime. The trick will be ensuring that the runtime code always has entries for everything in the conversion, even if just stubs to hold the annotations.

One question: how do we deal with things that are options or items that don't have a methods in the runtime that are natural backing features that can be annotated. For example, for language features that just convert to parameters of a method, how do we encode the data to mark these?

#7 - 11/16/2016 01:35 PM - Greg Shah

- Target version deleted (25)