Conversion Tools - Feature #7318

prevent abend during conversion in case of failures when processing TRPL rules

05/01/2023 08:28 AM - Constantin Asofiei

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Conversion Tools - Feature #7179: rework the conversion process to...	**New**

History

#1 - 05/01/2023 08:29 AM - Constantin Asofiei

This is from #7199-25:

> Greg Shah wrote:
>
> > Constantin Asofiei wrote:
> >
> > > Greg, there is a simple change in Pattern.processAst - enclose it in a try/catch(Throwable) block, and I think this will not abend the conversion in case of FWD bugs.
> >
> > I think we need to remove that AST from subsequent conversion steps otherwise it will be a real mess. And when we remove that AST, if it is a class/interface/enum there may be many other dependencies which would break. For this reason, I think the problem is bigger.

#2 - 05/01/2023 08:39 AM - Constantin Asofiei

Currently, in case of FWD bugs in TRPL rules, FWD can abend and terminate the conversion. For very large code-sets, this is time-expensive as it hides any failures in files which would be processed after this 'first abend'. We need a way to:

- prevent the conversion to abend - this is the Pattern.processAst enclosed in a try/catch proposal
- log any relevant info about the file which abended and why, and collect this file name
- on conversion finish, report the files which abended and terminate with a non-zero exit code.

Regarding this note:

> Greg Shah wrote:
>
> > I think we need to remove that AST from subsequent conversion steps otherwise it will be a real mess. And when we remove that AST, if it is a class/interface/enum there may be many other dependencies which would break. For this reason, I think the problem is bigger.

I don't necessarily agree that we should remove the AST from the conversion. I agree that if a file fails i.e. during annotation, it will almost certainly fail on core code conversion, too. And it may bring other files to fail, like when a .cls fails and dependent files fail, or shared temp-tables/frames.

We already know that the conversion is in a 'failed' state when we encounter a failed file. But, leaving it in the conversion file list, and all its dependencies, FWD will report these, which will allow us to i.e. convert all these failed files separately (to test the fixes) until we try a full run again.

This may need to work in conjunction with #6082 to automatically pull all .cls files into conversion (but this can be solved manually by running parse first, get all the pre-scanned .cls, add them to the file-cvt-list.txt and run conversion again).

#3 - 05/02/2023 08:47 AM - Greg Shah

- Related to Feature #7179: rework the conversion process to ensure that it can finish an entire run even when there are failures after parsing added

#4 - 05/02/2023 08:50 AM - Greg Shah

This is really a duplicate of #7179.

> I agree that if a file fails i.e. during annotation, it will almost certainly fail on core code conversion, too. And it may bring other files to fail, like when a .cls fails and dependent files fail, or shared temp-tables/frames.

Which will create a large amount of unnecessary noise in the output. People may look at the later problems and miss the earlier root cause. The result of that will be wasted time. Generally, I don't see any value in failures that are downstream of the root cause. They aren't real failures. Reporting them is a net negative.

> But, leaving it in the conversion file list, and all its dependencies, FWD will report these, which will allow us to i.e. convert all these failed files separately (to test the fixes) until we try a full run again.

This is not a reason to leave them in. We can just output the list of removed files at the time we remove them, so that it is clear what needs to be re-run.

#5 - 05/02/2023 08:50 AM - Greg Shah

- Subject changed from prevent abent during conversion in case of failures when processing TRPL rules to prevent abend during conversion in case of failures when processing TRPL rules