Conversion Tools - Bug #5135

replace -s, -f and -x options of ConversionDriver with a single combined mode

02/09/2021 07:45 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Conversion Tools - Feature #3927: conversion mode that takes a fil...	**Closed**
Related to Conversion Tools - Bug #5137: any text following a # character in ...	**Closed**
Related to Conversion Tools - Bug #5021: improve/fix file-ignore-list.txt ('X...	**Closed**
Related to Conversion Tools - Feature #5630: add missing classes referenced d...	**New**
Related to Conversion Tools - Feature #4105: improve flexibility of schema lo...	**Closed**
Related to Conversion Tools - Feature #6253: add file-set processing into p2j...	**Closed**

History

#1 - 02/09/2021 08:56 PM - Greg Shah

- Subject changed from replace @-s@, @-f@ and @-x@ options of @ConversionDriver@ with a single combined mode to replace -s, -f and -x options of ConversionDriver with a single combined mode

We have 3 modes for specifying the source files to process in the ConversionDriver.

The result causes issues:

1. In complex projects, one must often specify more files than needed.
   - In -f mode, it would be better if you could use a specification to set a default so that only a small number of exceptions could be written. Instead one must list **every** file in the entire project. It is a pain to create in the first place and then it is even more work to maintaining over time (as new files are added/removed in the project).
   - In -x mode, the addition of the file spec is really useful. The problem comes when one wants to include a small number of files from an otherwise fully excluded directory tree. A classic example is the case where you just want to include small number of Possenet/ADM2 files but otherwise exclude the majority of it. Look at this (from a real project):

     ```
     ./abl/possenet/src/adebuild/*
     ./abl/possenet/src/adecomm/_*
     ./abl/possenet/src/adecomm/*.def
     ./abl/possenet/src/adecomm/convcp.p
     ./abl/possenet/src/adecomm/xmlschema.p
     ./abl/possenet/src/adecomp/*
     ./abl/possenet/src/adedesk/*
     ./abl/possenet/src/adedict/*
     ./abl/possenet/src/adeedit/*
     ./abl/possenet/src/adeicon/*
     ./abl/possenet/src/aderes/*
     ./abl/possenet/src/aderesc/*
     ./abl/possenet/src/adeshar/*
     ./abl/possenet/src/adetran/*
     ./abl/possenet/src/adettdb/*
     ./abl/possenet/src/adeuib/*
     ./abl/possenet/src/adeweb/*
     ./abl/possenet/src/adexml/*
     ./abl/possenet/src/adm/*
     ./abl/possenet/src/adm2/a*.*
     ./abl/possenet/src/adm2/b*.*
     ./abl/possenet/src/adm2/ca*.*
     ```

```
./abl/possenet/src/adm2/cob*.*
./abl/possenet/src/adm2/com*.*
./abl/possenet/src/adm2/cons*.*
./abl/possenet/src/adm2/containr.ado
./abl/possenet/src/adm2/containr.cld
./abl/possenet/src/adm2/custom/*
./abl/possenet/src/adm2/da*.*
./abl/possenet/src/adm2/de*.*
./abl/possenet/src/adm2/dynb*.*
./abl/possenet/src/adm2/dync*.*
./abl/possenet/src/adm2/dyndata.ado
./abl/possenet/src/adm2/dynf*.*
./abl/possenet/src/adm2/dynl*.*
./abl/possenet/src/adm2/dynp*.*
./abl/possenet/src/adm2/dynr*.*
./abl/possenet/src/adm2/dyns*.*
./abl/possenet/src/adm2/dynt*.*
./abl/possenet/src/adm2/dynw*.*
./abl/possenet/src/adm2/e*.*
./abl/possenet/src/adm2/fetchc*.*
./abl/possenet/src/adm2/fetchdata.ado
./abl/possenet/src/adm2/fetchde*.*
./abl/possenet/src/adm2/fetchrows.ado
./abl/possenet/src/adm2/fi*.*
./abl/possenet/src/adm2/fo*.*
./abl/possenet/src/adm2/g*.*
./abl/possenet/src/adm2/i*.*
./abl/possenet/src/adm2/image/*
./abl/possenet/src/adm2/j*.*
./abl/possenet/src/adm2/l*.*
./abl/possenet/src/adm2/m*.*
./abl/possenet/src/adm2/p*.*
./abl/possenet/src/adm2/q*.*
./abl/possenet/src/adm2/r*.*
./abl/possenet/src/adm2/sa*.*
./abl/possenet/src/adm2/sb*.*
./abl/possenet/src/adm2/sc*.*
./abl/possenet/src/adm2/sd*.*
./abl/possenet/src/adm2/se*.*
./abl/possenet/src/adm2/smart.ado
./abl/possenet/src/adm2/smart.cld
./abl/possenet/src/adm2/smr*.*
./abl/possenet/src/adm2/support/_*.*
./abl/possenet/src/adm2/support/b*.*
./abl/possenet/src/adm2/support/c*.*
./abl/possenet/src/adm2/support/d*.*
./abl/possenet/src/adm2/support/f*.*
./abl/possenet/src/adm2/support/n*.*
./abl/possenet/src/adm2/support/p*.*
./abl/possenet/src/adm2/support/r*.*
./abl/possenet/src/adm2/support/s*.*
./abl/possenet/src/adm2/support/t*.*
./abl/possenet/src/adm2/support/u*.*
./abl/possenet/src/adm2/support/vb*.*
./abl/possenet/src/adm2/support/vc*.*
./abl/possenet/src/adm2/support/vie*.*
./abl/possenet/src/adm2/support/visuald.ado
./abl/possenet/src/adm2/template/*
./abl/possenet/src/adm2/t*.*
./abl/possenet/src/adm2/u*.*
./abl/possenet/src/adm2/view*.*
./abl/possenet/src/adm2/visp*.*
./abl/possenet/src/adm2/visual.ado
./abl/possenet/src/adm2/visual.cld
./abl/possenet/src/adm2/w*.*
./abl/possenet/src/adm2/x*.*
./abl/possenet/src/as4dict/*
./abl/possenet/src/dynamics/af/app/*
./abl/possenet/src/dynamics/af/bmp/*
./abl/possenet/src/dynamics/af/cod/*
./abl/possenet/src/dynamics/af/cod2/afa*.*
./abl/possenet/src/dynamics/af/cod2/afc*.*
./abl/possenet/src/dynamics/af/cod2/afg*.*
./abl/possenet/src/dynamics/af/cod2/afm*.*
./abl/possenet/src/dynamics/af/cod2/afp*.*
```

```
./abl/possenet/src/dynamics/af/cod2/afs*.*
./abl/possenet/src/dynamics/af/cod2/aftemc*.*
./abl/possenet/src/dynamics/af/cod2/afteml*.*
./abl/possenet/src/dynamics/af/cod2/aftemsuspd.ado
./abl/possenet/src/dynamics/af/cod2/aftemw*.*
./abl/possenet/src/dynamics/af/cod2/d*.*
./abl/possenet/src/dynamics/af/cod2/f*.*
./abl/possenet/src/dynamics/af/cod2/g*.*
./abl/possenet/src/dynamics/af/cod2/p*.*
./abl/possenet/src/dynamics/af/cod2/r*.*
./abl/possenet/src/dynamics/af/cod2/s*.*
./abl/possenet/src/dynamics/af/obj2/*
./abl/possenet/src/dynamics/af/rep/*
./abl/possenet/src/dynamics/af/sup/*
./abl/possenet/src/dynamics/af/sup2/*
./abl/possenet/src/dynamics/afr*.*
./abl/possenet/src/dynamics/as*.*
./abl/possenet/src/dynamics/db/*
./abl/possenet/src/dynamics/i*.*
./abl/possenet/src/dynamics/icf/*
./abl/possenet/src/dynamics/install/*
./abl/possenet/src/dynamics/p*.*
./abl/possenet/src/dynamics/ry/*
./abl/possenet/src/dynamics/scm/*
./abl/possenet/src/prodict/*
./abl/possenet/src/prohelp/*
./abl/possenet/src/prores/*
./abl/possenet/src/protools/*
./abl/possenet/src/template/*
./abl/possenet/src/web/*
./abl/possenet/src/web2/*
./abl/possenet/src/webedit/*
./abl/possenet/src/webspeed/*
./abl/possenet/src/webtools/*
./abl/possenet/src/webutil/*
./abl/possenet/src/workshop/*
./abl/possenet/src/wrappers/*
```

2. The scripting for projects is very complex and often broken in some build/conversion targets. The standard project template has 3 front end targets, 3 main conversion targets etc... (one for -s, one for -f and one for -x).
3. Very confusing.
   - It is all very confusing for users, especially in any scenario where either or both of the file/ignore lists exists.
   - Notice that the possenet exclude list above is very long **and has none of the actual included files in it**. This means that it is error prone and it is hard to reason about.

We only need a single mode   By implementing a single approach all of the above problems will be gone. The idea:

1. Same 3 parameter syntax as -x:
   - top_level_src_dir
   - file_spec
   - list_filename
2. We should default the file_spec to (*.[pPwW]|*.cls). The user should be able to optionally override this with their own spec.
3. The only required parameter would be top_level_src_dir. I don't think it is reasonable to default this (yet).
4. If the optional list_filename is specified, then each line would be in the format <directive> <file_or_path_spec>.
   - directive would be I (include the spec) or X (exclude the spec), case-insensitive.
   - file_or_path_spec would be the same syntax used by -x mode today, except it would be interpreted based on the directive.
   - The is an implicit I <command_line_or_default_file_spec at the top of the file.
   - The file is processed top to bottom, with each directive potentially editing the list (adding or removing files).

The above example of Possenet files would be re-written as this condensed version:

```
X ./abl/possenet/*
I ./abl/possenet/src/adecomm/as-utils.w
I ./abl/possenet/src/adecomm/get-user.p
I ./abl/possenet/src/adm2/containr.p
I ./abl/possenet/src/adm2/dyndata.w
I ./abl/possenet/src/adm2/fetchdata.p
I ./abl/possenet/src/adm2/fetchrows.p
I ./abl/possenet/src/adm2/smart.p
I ./abl/possenet/src/adm2/support/visuald.w
I ./abl/possenet/src/adm2/visual.p
I ./abl/possenet/src/dynamics/af/cod2/aftemsuspd.w
```

#2 - 02/10/2021 07:51 AM - Greg Shah

- Related to Feature #3927: conversion mode that takes a file spec and a file blacklist added

#3 - 02/10/2021 08:00 AM - Greg Shah

- Related to Bug #5137: any text following a # character in the include or exclude file list is ignored added

#4 - 07/08/2021 06:07 AM - Greg Shah

- Related to Bug #5021: improve/fix file-ignore-list.txt ('X' conversion mode) under Windows added

#5 - 08/31/2021 07:24 AM - Greg Shah

- Related to Feature #5630: add missing classes referenced during parsing to the conversion list in file list mode added

#6 - 10/25/2021 10:49 AM - Ovidiu Maxiniuc

- Related to Feature #4105: improve flexibility of schema loading added

#7 - 04/06/2022 05:04 PM - Greg Shah

I've been thinking about the syntax for this approach. Using a list of X (exclude) and I (include) filters is a powerful tool, but it isn't enough to handle all cases. The original idea was to use a single top_level_src_dir + file_spec to create the maximum list of inputs. Then the file that contains filters (X and I) would modify that list. When I wrote #5135-1, I was confusing the addition of files with the I filter. The two should be separate.

Cases that won't work using this approach:

- Input lists that can't easily be represented by top_level_src_dir + file_spec.
  - The current default for ConversionDriver allows for specifying an explicit list of source files on the command line.
  - The -f mode is the same case, but stored inside an explicit file list.
  - These approaches can add any list of sources, even if it does not match the limited pattern provided by top_level_src_dir + file_spec.
- The case where more than one top_level_src_dir + file_spec would most simply represent the maximum input list.
  - The alternative is to use an enclosing directory that contains all needed locations and an overbroad spec while then adding many more filtering expressions to cut the unwanted parts out.
  - But this still might not support all cases. If a single file_spec cannot match all possible source files, then this breaks down.

I think we can implement a single approach to solve all of these cases. We need 5 directives instead of 2 directives.

- D <recursive_top_level_dir> <file_spec> adds a top level dir and file spec which will be recursively processed to add 0 or more files into the list
- N <non_recursive_top_level_dir> <file_spec> adds a top level dir and file spec which will be non-recursively processed to add 0 or more files into the list
- F <absolute_or_relative_file_name> - explicitly adds a single/individual file into the list
- X <exclusion_filter_expression_including_wildcards> - filter to remove from the list
- I <inclusion_filter_expression_including_wildcards> - filter to add something removed back into the list

The X and I are the same as above, but now we can add files into the maximal file list using D, N and F. The input file will be processed in 2 passes:

1. Process all D, N and F directives to make the maximal file list of all matches. These can be processed in any order because they only ever add to the list. In practice there is no reason to process them in anything other than top to bottom. If there are no D, N and F directives, then we will create a maximal file list using D . (*.[pPwW]|*.cls) which will gather all .p, .P, .w, .W and .cls files under the current directory, recursively.
2. Process all X and I filters, in order top to bottom, to modify the maximal list into the actual list.

With this approach, I think we don't use 3 command line parameters. There is just a single parameter for the configuration file.

#8 - 04/06/2022 06:03 PM - Greg Shah

- Assignee set to Constantin Asofiei

#9 - 04/06/2022 06:14 PM - Greg Shah

- Related to Feature #6253: add file-set processing into p2j.cfg.xml added

#10 - 04/06/2022 06:17 PM - Greg Shah

This new file processing approach can be activated with the -Z<input_file> command line option. I know this is different from the other existing -f, -s and -x options but we are getting rid of those eventually so I don't think we need to follow the same command line approach there.

Please implement the backing support for this in common code in the FileListFactory and FileList class hierarchy. That will allow us to reuse that support from other places (maybe first in #6253).

#11 - 04/07/2022 03:04 AM - Constantin Asofiei

- Status changed from New to WIP

Greg, to make something clear: the X and I filters will work only on the set of files determined from the D, N, F operations, right? So, if an I filter references an exact file which is not part of the D + N + F file set, then it will not be included.

The 'dry run' (-l) mode will be used to test compatibility between a -Z mode and an existing whitelist (-f) or blacklist (-x) mode, so we know the same files are converted, for existing projects.

#12 - 04/07/2022 06:55 AM - Greg Shah

> Greg, to make something clear: the X and I filters will work only on the set of files determined from the D, N, F operations, right? So, if an I filter references an exact file which is not part of the D + N + F file set, then it will not be included.

Correct.

> The 'dry run' (-l) mode will be used to test compatibility between a -Z mode and an existing whitelist (-f) or blacklist (-x) mode, so we know the same files are converted, for existing projects.

Yes, that makes sense.

#13 - 04/07/2022 06:55 AM - Greg Shah

As with other flags -z and -Z should be equivalent.

#14 - 04/07/2022 10:25 AM - Constantin Asofiei

Greg, in the case where we use a file, there will be problems if there are space characters in either the file spec or the actual folder name, when the D or N modes are used, as the space is used as a separator. I'll add protection against this (i.e. a single space must exist beside the second character).

#15 - 04/07/2022 11:14 AM - Greg Shah

Yes, this makes sense. We should also require double quoted strings when there are embedded space characters.

```
D "some stupid name" "*dumb\ spec*.[pPwW]"
```

#16 - 04/07/2022 11:19 AM - Constantin Asofiei

Greg Shah wrote:

> Yes, this makes sense. We should also require double quoted strings when there are embedded space characters.
>
> [...]

But double quotes can be part of a file name... I'll just force them to be escaped with '\' and 'unescape' them once the line is parsed.

#17 - 04/07/2022 02:16 PM - Constantin Asofiei

Greg, I'll remove the whitelist and blacklist modes in current work, but the 'FILESPEC' (-s mode) should remain, this is very useful for development (at least for me).

#18 - 04/07/2022 02:25 PM - Greg Shah

I was assuming that all of the modes would remain until we've migrated all projects into the -z.

#19 - 04/07/2022 02:25 PM - Greg Shah

And yes, it is OK to leave -s.

#20 - 04/08/2022 09:17 AM - Constantin Asofiei

Another issue: lets assume we are under Windows, where file names are case-insensitive; you have a F ./Build.p line and there is an operation X build.p. The file names (at OS level) match.

I can either use java.nio.Path.toRealPath (if it works) to resolve the real filename at OS level, or otherwise rely on the FileList.caseSens flag for case-sensitivity matching.

The other problem is that ./build.p and build.p both refer the same file, so these need to be standardized.

#21 - 04/08/2022 10:00 AM - Greg Shah

> I can either use java.nio.Path.toRealPath (if it works) to resolve the real filename at OS level, or otherwise rely on the FileList.caseSens flag for case-sensitivity matching.

Yes, let's handle the case-insensitive matching. We should not require the filenames to be case-sensitively matched on Linux when the project comes from Windows. We already handle this in preprocessor using FileScope.resolveFileName(). Inside there we use Utils.canonicalizePath() and FileScope.findCaseInsensitively(). I know we have runtime code that also does this. In a perfect world, we would implement common helpers used from all these places.

> The other problem is that ./build.p and build.p both refer the same file, so these need to be standardized.

Agreed. FYI, the -x mode is currently dependent upon the ./ prefix for all these files. It is a consequence of the current usage of the regex matches() in FileSpecList.listImpl(). Eric added a notice in the class javadoc to note this leading char matching weirdness.

#22 - 04/13/2022 08:19 AM - Constantin Asofiei

Z mode is stripping the ./ prefix, because of some peculiarities when resolving the path (I need the real OS filenames, to be able to apply the filters properly).

I can 'force it' back, to work the same as blacklist works, but considering that these will be removed, I don't see the point.

#23 - 04/13/2022 08:28 AM - Greg Shah

OK

#24 - 04/13/2022 08:32 AM - Constantin Asofiei

- Status changed from WIP to Review

- % Done changed from 0 to 100

The changes are in 3821c/13778 for both this task and #6253.

#25 - 04/15/2022 12:55 AM - Eric Faulhaber

I created a list file for a project. The file contained several lines using the D directive. I noticed there can be only one space between the <recursive_top_level_dir> text and the <file_spec> text. If there is more than one space, that directive is ignored silently. I had multiple spaces in lines after the first, so that the <file_spec> text was aligned in a column for readability. Since, there is no warning that the lines are not accepted/processed, it is easy to get unexpected results.

Once I removed the extra spaces, it worked great.

#26 - 04/15/2022 08:10 AM - Greg Shah

Any amount of whitespace (not just space char) should be ignored.

#27 - 04/16/2022 02:58 PM - Constantin Asofiei

Greg Shah wrote:

> Any amount of whitespace (not just space char) should be ignored.

Please see 3821c/13788.

#28 - 04/18/2022 05:45 AM - Greg Shah

- Status changed from Review to Closed