Last edited by **Ondřej Vašíček** 2 weeks ago

3. Plugin Output Filters

The Analysis adapter executes analysis tools which produce varying outputs. To allow users to control what outputs are saved as the result of running an analysis, there is a plug-in system in place which allows users to create their own custom filters to process analysis outputs.

Concept

Automation Results contain Contribution resources which hold outputs of the automation execution. The Analysis adapter by default produces these Contributions:

- **executionTime** - total duration of the analysis in milliseconds
- **statusMessage** - progress messages from the adapter
- **returnCode** - exit code of the compilation
- **stdout** - stdout of the analysis
- **stderr** - stderr of the analysis
- ***filename*** - any number of contributions representing files produced or modified during analysis

Contribution resources have these properties:

- **title** - name of the contribution (e.g. stdout, stderr, returnCode, ...)
- **value** - value of the contribution according to the valueType (usually a string)
- **valueType** - value type of the contribution's value (e.g. http://www.w3.org/2001/XMLSchema#string)
- **description** - any description of the contribution
- **filePath** - Path to the actual file in the adapter's file system.

The Analysis adapter runs all contributions through a filter selected by the *outputFilter* parameter of the Automation Request. The filter can process all contributions in any way including deleting them entirely, creating new ones, modifying their values, parsing their contents and deriving new infos, etc.

How To Create a Filter

There are two steps needed to create an output filter.

1) create a configuration file

A .properties file describing the filter needs to be created in the *conf/analysis_advanced/PluginFilters/* directory. Use the *ExamplePluginFilter.properties* file as a template.

Properties that need to be defined are:

- **implements** - class path of the IFilter interface of the adapter -- do not change this value
- **class** - class path of your own filter (pluginFilters.customPluginFilters.YourFilterClass)
- **tool** - identifier of the AutomationPlan which this filter is meant for

2) create a .java file implementing the filter

Create .java class file implementing the *cz.vutbr.fit.group.verifit.oslc.analysis.outputFilters.IFilter* and *cz.vutbr.fit.group.verifit.jsem.IExtension* interfaces in the *conf/analysis_advanced/PluginFilters/* directory. Use the ExamplePluginFilter.java as a template.

3) rebuild the adapter

Run build.[sh|ps1] and then run the adapter

Filter interface

The filter needs to implement the parse method *"void parse(List<Map<String,String>> inoutContributions)*". The in/out parameter contains a list of maps where each list item represents one Contribution represented by a key/value map. Keys are names of Contribution properties (id, title, value, valueType, description, filePath) and values hold their corresponding values.

Contribution IDs need to be unique per Automation Request (only one of each ID for the same Automation Request). The ID set by the filter gets a prefix holding the Automation Request ID added to it to ensure uniqueness among all Contributions of all Automation Requests.

The value of the filePath property can be used to access the file represented by the Contribution and, for example, load its contents.

The simplest filter does not do anything leaving all Contributions untouched. The default loads the contents of all file contributions that represent non-binary files (such as the stdout and stderr contributions) and leaves all other contributions untouched.