PreProccessFilter 2

Introduction

This preprocess filter assist the users to inspect the values of a specific element in an xml file and to change the desired values in the whole file keeping the information of the old and new values. It is mostly applied in cases that the xml file has a very big size, and the specific element appears a lot of times. The preprocess filtering consists of two parts. In the first part the elements of interest and their values are extracted in a new file, and are being sorted giving the ability to the user to easily change the desired values. In the second part the changes are applied to the initial input file and the final output file is being produced.

The filter is implemented as a maven java project. It can be executed either by the command line, providing a number of arguments or by filling a configuration file and running the produced jar.

User Manual

The user can use the filter either by running it by the command line providing the required parameters or by filling a configuration file and running the jar.

Running by the config.properties file and the jar

In order to use the filter by running the jar the user has to fill a config.properties file. Specifically the user has to fille the following:

- inputFilePath : the path of the xml input file
- outputFilePath: the path of the ouput xml file that will be produced
- createNewValuesFile: the option to create the new values file. If it set to yes then the new values file is being produced. If is set to no then the changes are applied to the initial input file.
- parentNode: the node that contains values that may be changed.
- newParentNode=the new parent that will replace the initial parent node and will have as children the new value's and the old'values elements
- sameValueTag: the name of the node in the case that the value has not been changed
- oldValueTag: the name of the node that will contain the old value
- newValueTag: the name of the node that will contain the new value

After the filling of the config.properties file the user can run the jar (***preProcessFilter2-1.0.jar***) either by double-clicking on it or by running it from the command file with the command:

java –jar preProcessFilter2-1.0.jar

IMPORTANT: the config.properties file and the preProcessFilter2-1.0.jar must be in the same folder

Running by using command line arguments

To run the filter by the command line the user has to provide the following arguments:

-inputFile : the path of the xml input file
-outputFile: the path of the output xml file that will be produced
-parentNode: the parent node that contains the values that will be splitted and re-assigned
-newValuesFile: the path of the new values xml file that will contained the orders text contents to be changes
-newParentNode: the new parent node that will be created
-newValueTag: the name of the tag that will contain the new values in the output file
-oldValueTag: the name of the tag that will contain the old values in the output file
-sameValueTag: the name of the tag that will contain the values that have not changed in the output file
-createNewValues: the option (yes or no) to create the new values file or to create the output file

The command to run the filter is the following:

java -jar preProcessFilter2-1.0.one-jar.jar -inputFile "inputFilePath" -outputFile "outputFilePath" -parentNode "parentNodeName" –newValuesFile "newvaluesFilePath" -newParentNode "newParentNodeName" -newValueTag "newValueTagName" -sameValueTag "oldValueTagName" -sameValueTag "sameValueTagName" –createNewValues "yesORno"

If the command runs successfully the following message will be returned:

"Successful Preprocessing!!!"

Example

An example of using the filter is described in this section.

The input file is the events.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <other>someotherinfo</other>
  <event>Event1 Description</event>
  <event>Event2 Description</event>
  <event>Event1 Description</event>
  <other>someotherinfo2</other>
  <event>Event3 Description</event>
</root>
```

The goal is to extract all the distinct event node values, to change some of them and afterwards to apply the changes to the initial xml file. The first step is to change the configuration file as follows:

```
#  Configuration file
#  If you want to use '\' make sure to change it '\\'
#  Separate the tags by using ','

inputFilePath= events.xml

outputFilePath= output.xml

createNewValuesFile=yes

newValuesFilePath= newValuesFile.xml

parentNode=events

newParentNode=eventsNew

sameValueTag=eventsSame

oldValueTag=oldValue

newValueTag=newValue
```

By running the "***preProcessFilter1-1.0.jar***" the newValues.xml file will be produced that contains all the event nodes values sorted and the newValue elements that will contain the new values:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root>
   <events>Event1 Description</events><newValue>Event1 NEW Description</newValue>
   <events>Event2 Description</events><newValue></newValue>
   <events>Event3 Description</events><newValue>Event3 NEW Description</newValue>
</root>
```

The next step is to change the configuration file in order to apply the changes to the initial xml file. The createNewValuesFile option will be set to no. The new parent node will be named eventsNEW, for the cases that the value has not been changed the node will be named eventsSame, the old values will be included in the oldValue nodes, and the new values will be included in the newValue nodes:

```
#  Configuration file
#  If you want to use '\' make sure to change it '\\'
#  Separate the tags by using ','

inputFilePath= events.xml

outputFilePath= output.xml

createNewValuesFile=no

newValuesFilePath= newValuesFile.xml

parentNode=events

newParentNode=eventsNew
```

```
sameValueTag=eventsSame

oldValueTag=oldValue

newValueTag=newValue
```

By running the "***preProcessFilter1-1.0.jar"*** the output.xml file will be produced (the changes of the initial input xml are in green fonts):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root>
   <other>someotherinfo</other>
   <eventsNew>
      <newValue>Event1 NEW Description</newValue>
      <oldValue>Event1 Description</oldValue>
   </eventsNew>
   <eventsSame>Event2 Description</eventsSame>
   <eventsNew>
       <newValue>Event1 NEW Description</newValue>
      <oldValue>Event3 Description</oldValue>
   </eventsNew>
   <other>someotherinfo2</other>
   <eventsNew>
      <newValue>Event3 NEW Description</newValue>
      <oldValue>Event1 Description</oldValue>
   </eventsNew>
</root>
```