SwingScaViewer

V1.3.X

Installation Guide

User Guide

REF : SwingScaViewer/JLP/2012/10

Author : Jean-Louis PASTUREL

Description of Successive Versions

VERSION	DATES	State	Author
V1.0	30/May/2012	Initialisation Version	JL PASTUREL
V1.1.0	15/Aug/2012	Version 1.1.0	JL PASTUREL
V1.1.2	27/Aug/2012	Version 1.1.2	JL PASTUREL
V1.1.8	03/Sep/2012	Version 1.1.8	JL PASTUREL
V1.2.0	09/Sep/2012	Version 1.2.0	JL PASTUREL
V1.3.3	01/Oct/12	Version 1.3.3	JL PASTUREL
V1.3.9	26/Oct/2012	Update for Version 1.3.9	JL PASTUREL
V1.3.18	11/Feb/2013	Update for Version 1.3.18	JL PASTUREL

Table des matières

Advertisement :
English is not my native language, so in the document, you will find a lot of syntax and grammar mistakes, awkwardness, difficulties to understand some sentences ...
Please let me know at jean-louis.pasturel-wrong-reply@orange.fr
(Remove -wrong-reply to the mail address)

1 Introduction

1.1 Context

Important :
To use this tool, you must know basics on Pattern Matching with regular expression (Perl regex for example). At the end of the document, I give some examples of regex that are currently used in **swingScaViewer**.
The general mechanism used in this tool is to parse in 2 phases :

- First phase : first regex extracts from a source that contains the interesting information in a result
- Second phase, **if necessary** : second regex extracts, from precedent result, the final information

This mechanism can handle almost all cases

The product **SwingScaViewer** is a kind of workbench that groups several tools :
– parsing dated logs (system logs, Web servers, WAS, application ...) and converting them into csv files
– visualisation of csv files, or direct visualisation with certain types of logs (JVM GC logs for example).
– Integration of others tools like AspectPerf(packaging AspectJ LTW Weaving for profiling Java application), JDBC Requests, statFilesAdvanced …
– upload and download of files
– Utilities tools => date <=> dateInMillis, aggregation of files, regex expression tester ...

The tool SwingScaViewer is developed in Scala (2.9.2) and uses several Java API:
SwingScaViewer Core : Apache 2 License http://www.apache.org/licenses/LICENSE-2.0.html
Scala License : http://www.scala-lang.org/print/146
JFreeChart License : LGPL V2.1 http://www.gnu.org/licenses/lgpl-2.1.html
jlpApis.jar (My own Apis) : LGPL V2.1 http://www.gnu.org/licenses/lgpl-2.1.html
Jsch (BSD License) : http://www.jcraft.com/jsch/LICENSE.txt
akka Actors Apache 2 License : http://www.apache.org/licenses/LICENSE-2.0.html
Jcommon : LGPL V2.1 http://www.gnu.org/licenses/lgpl-2.1.html
commons-math;1.2 : Apache 2 License : http://commons.apache.org/math/license.html
joda-time Apache 2 : http://joda-time.sourceforge.net/license.html
jtds Licencse :http://www.kxcad.net/esteco/modeFRONTIER320/legal/jtds/license.html
AspectJ (Eclipse Project) : Common Public License http://eclipse.org/legal/cpl-v10.html
mysql-connector-java GPL License : http://www.gnu.org/licenses/gpl.html
ojdbc14.jar : OTN License http://www.oracle.com/technetwork/licenses/distribution-license-

152002.html
postgresql JDBC Driver : BSD License http://jdbc.postgresql.org/license.html

Important :
The available packaging contains only the libraries that have the following licenses : LGPL, CPL, Apache, BSD
The libraries mysql-connector-java and ojdbc14.jar are not included in the free packaging. They have to be downloaded from the Web (only useful for JDBC Requests tool)
ojbcxx.jar => http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-10201-088211.html
mysql-connector-java => http://dev.mysql.com/downloads/connector/j/
Put them in the libExt directory and adapt the classpath in file scripts/swingScaViewer.cmd with the version of this JDBC drivers.

2 Installation

2.1 Packaging

Le packaging is done in a form of zip file **SwingScaViewer.zip** which the root is **swingScaViewer** .

2.2 Installation of SwingScaViewer

2.2.1 Requirements

A **Sun JDK 1.6.0+** must be installed on your desktop.

2.2.2 Create a deployment directory

For all the document, we suppose that you use a Windows System (there is no difficulty to adapt for a Linux box) and that the installation directory is **c:\opt** .
It **is not mandatory** to create a directory **swingScaViewer**.

2.2.3 De-compaction

After having downloaded **SwingScaViewer.zip** in **c:\opt** le de-compact-it in this directory.

2.2.4 Configuration

The configuration is set in the start script of **SwingScaViewer** :
File **c:\opt\swingScaViewer\scripts\swingScaViewer.cmd**

```
set root=C:\opt\swingScaViewer
set workspace=C:\opt\workspaceLP
set CLASSPATH=%root%\scaViewer.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\jcommon-1.0.17.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\jfreechart-1.0.14.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\scala-dbc.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\scala-library.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\scala-swing.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\commons-math-1.2.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\jlpApis.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\jsch-0.1.46.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\jtds-1.2.2.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\myaspectjweaver.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\mysql-connector-java-5.1.7-bin.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\ojdbc14.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\postgresql-8.3-604.jdbc3.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\springmyaspectjweaver.jar
set CLASSPATH=%CLASSPATH%;%root%\libExt\akka-actor-2.0.3.jar
set CLASSPATH=%CLASSPATH%;%root%\config
set CLASSPATH=%CLASSPATH%;%root%\myPlugins
set CLASSPATH=%CLASSPATH%;%root%\myPlugins\myPlugins.jar
Set GC_OPTS=-server -Xms1024M -Xmx1024M -XX:NewRatio=2 -XX:+UseParallelGC -XX:MaxPermSize=512M
-XX:PermSize=512M -XX:-UseAdaptiveSizePolicy
REM Set GC_LOGS=-verbose:gc  -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:
+PrintGCTimeStamps -Xloggc:%root%\logs\GC.logs

java %GC_OPTS% %GC_LOGS% -Dworkspace=%workspace% -Droot=%root%  -Dconfig.file=%root
%\config\scaViewer.properties -Xms1024M -Xmx1024M com.jlp.scaviewer.ui.SwingScaViewer
```

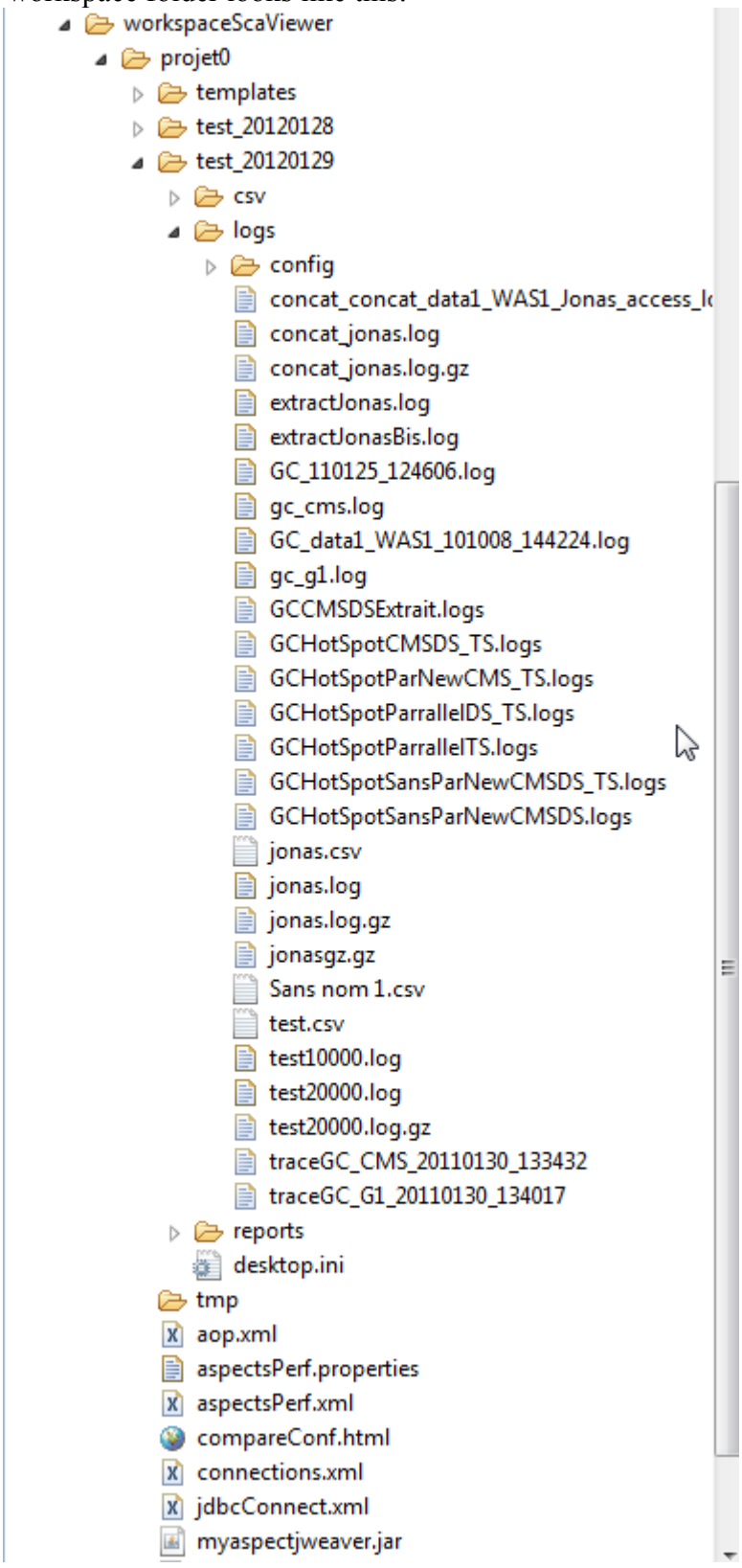
At the beginning of the file (in bold characters), 2 environment variables must be set, according to your installation.
The Window Environment Variable PATH has to make reachable the executable **java** of the JVM.
Important : The **manual creation** of the directory pointed by **%workspace%** is **mandatory.**
We can after create a link **Windows** on the desktop to launch **swingScaViewer** by a click on the mouse**.**

3 User Guide

For every feature of **swingScaViewer**, we are going to describe the threads of the screens and the parameters to set in the corresponding screens.
The screen-shots shown in this document correspond with the Version 1.x of **swingScaViewer** .
The general principle is to organize the **%workspace%** directory by project and by sub-folders.
These last sub-folders may be created by date, by type of scenario or other sorting keys.
When the scenario sub-folders is created, it is automatically prefixed by the variable (if not yet correctly prefixed) : `scaviewer.prefixscenario` initialised in the file :
config/scaViewer.properties

For the rest of the document, these sub-folders will be named “scenario folder”.

A typical tree in the workspace folder looks like this:

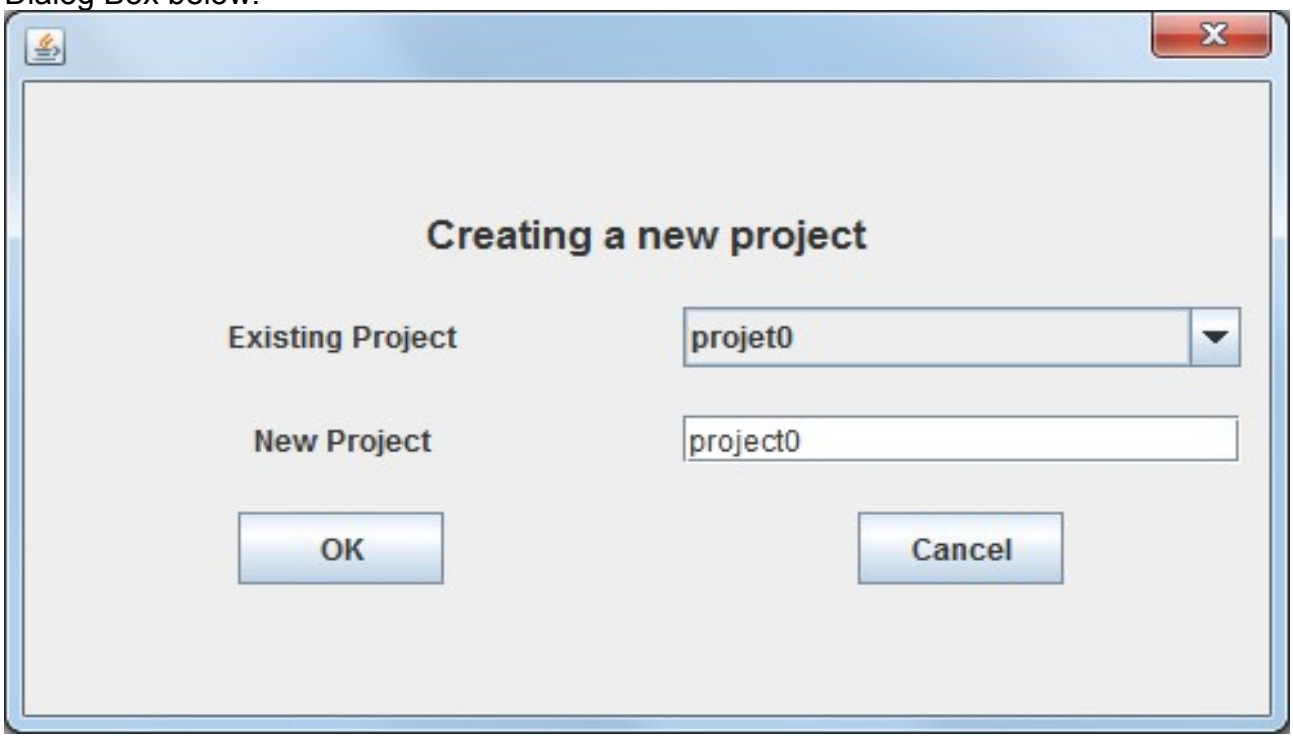


In the **logs** folder, there are the original access-logs or application logs files and in the **csv** folder, the csv files generated after treatment of logs files.

3.1 Menu "Files"

3.1.1 New Project

The sub-menu **New Project** permits to create a new project, which name is set in the Dialog Box below:

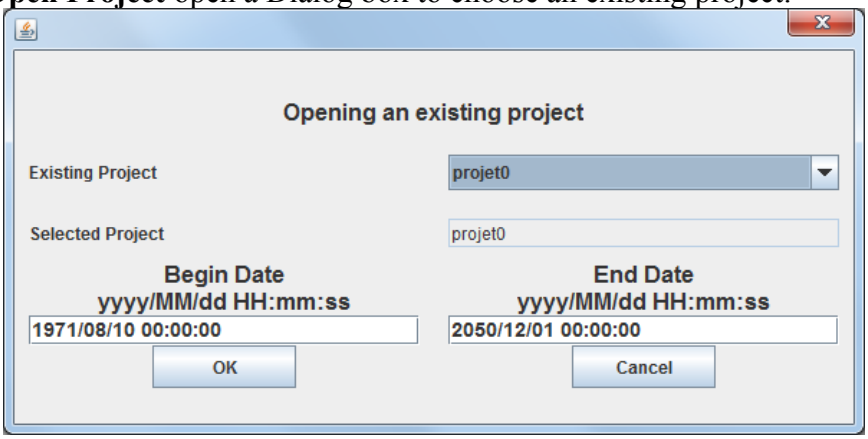


Click OK and a new directory **c:\opt\workspaceScaViewer\project0** is created if **%workspace%**

is set to **c:\opt\workspaceScaViewer**
This operation must be the first operation for a new project.

3.1.2 Open Project

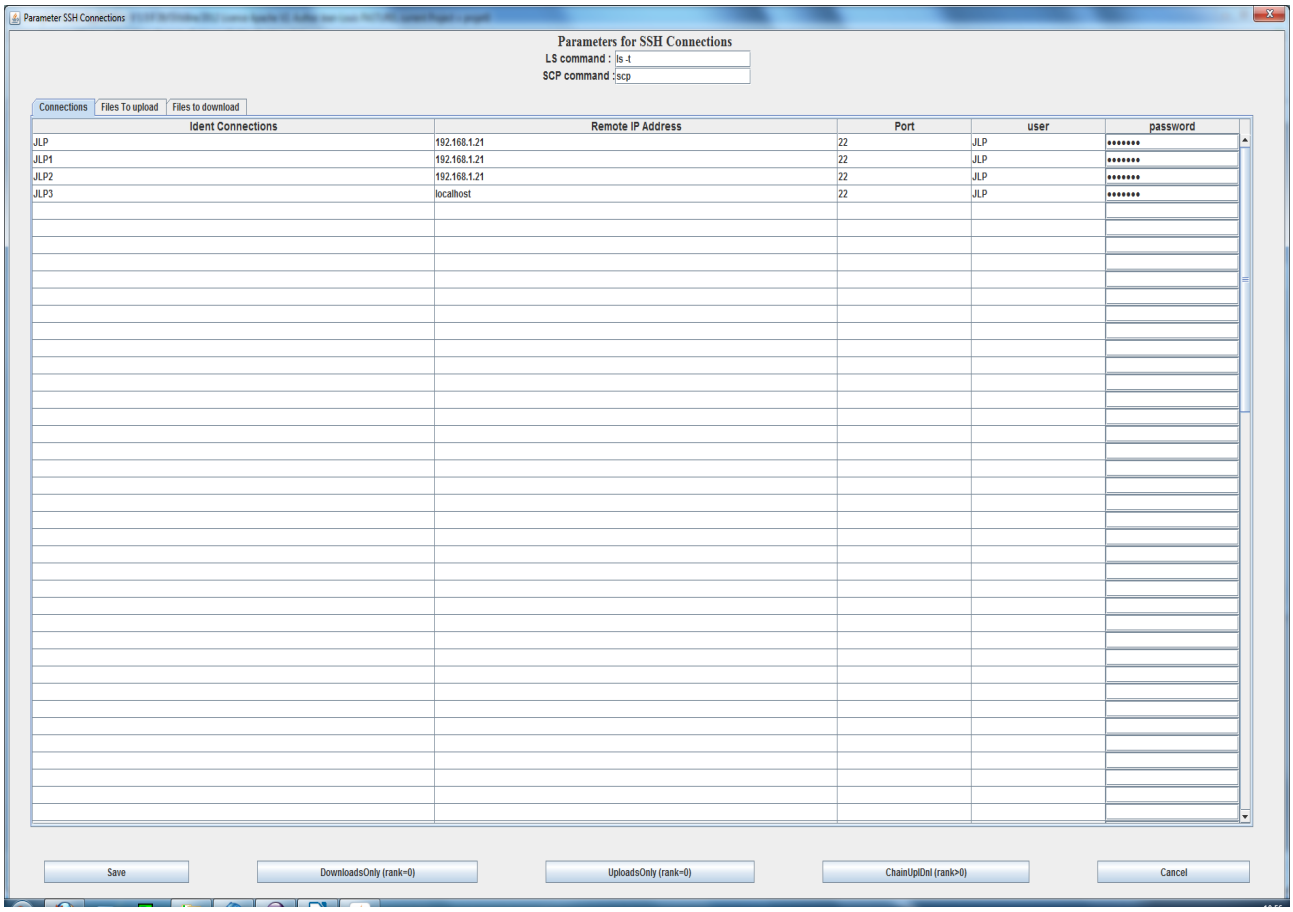
The sub-menu **Open Project** open a Dialog box to choose an existing project:



We can choose for example the existing project **projet0**
Starting with version 1.2, there is a begin / end date for the current scenario. The fields can be updated. It increases performance when parsing large files, which the the logs are not all useful.
Note : By creating or opening a project, we enable menus on the MenuBar. Closing the project disables these same menus.

3.1.3 SSH Cnx uploads/downloads

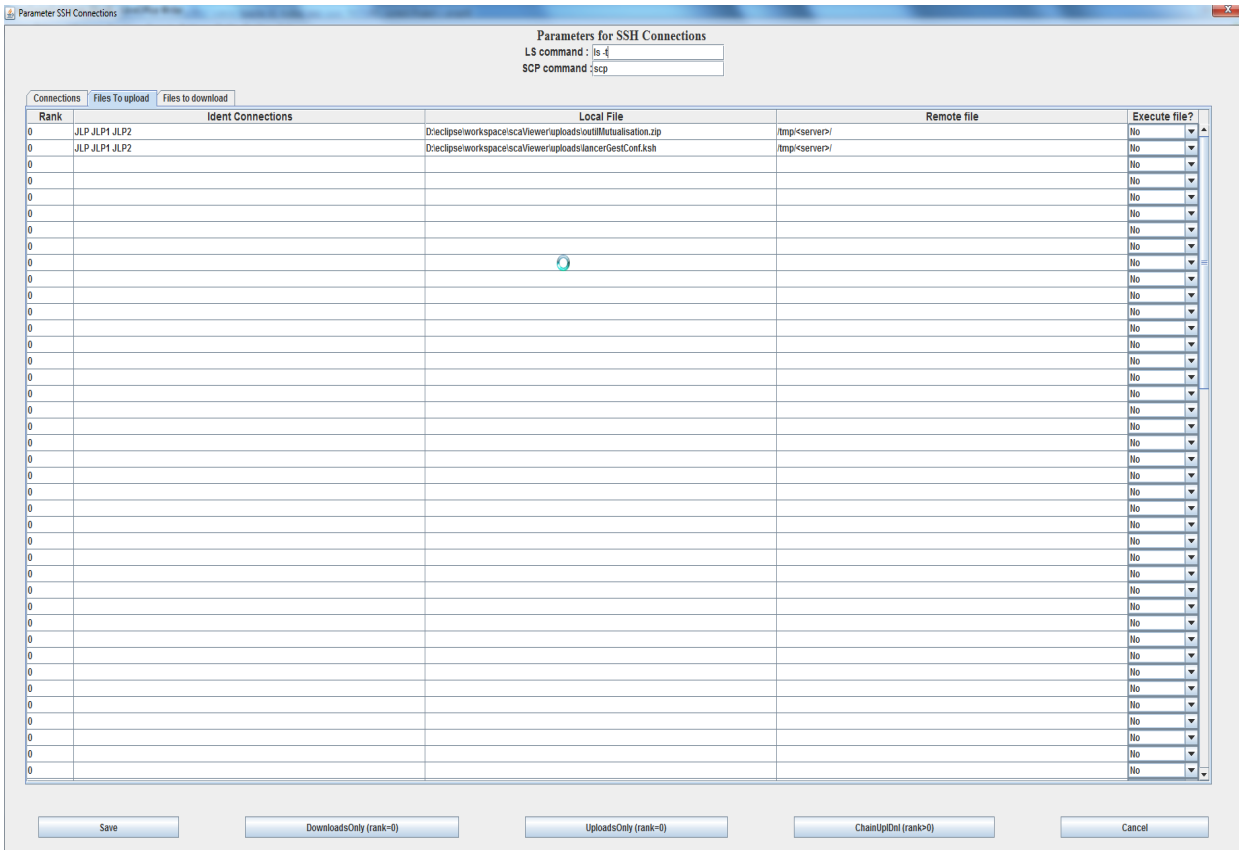
The sub-menu **SSH Cnx uploads/downloads** permits to automatize the downloads of log files or entire directories from the servers to the desktop that runs **swingScaViewer**
This mechanism is possible with direct ssh or throw ssh tunnels which are before hand opened.



The first tabPanel permits to define the connections (ident Servers, Ip Address, ssh port, login, password).

Hint : It is allowed to fill the same server with different names in the first column : it is useful when the files in a server have the same names (case of multiple WAS for example on the same server logs with the same names), the download mechanism in **swingScaViewer** prefix the file with the name of the server.

We can also adapt the command for listing files and also the command of scp. The default configuration fits well with Linux and Aix servers, no change is needed.



The second tab panel, permits to uploads files to the server (as for examples shell scripts)
The first column contains the rank of the operation :

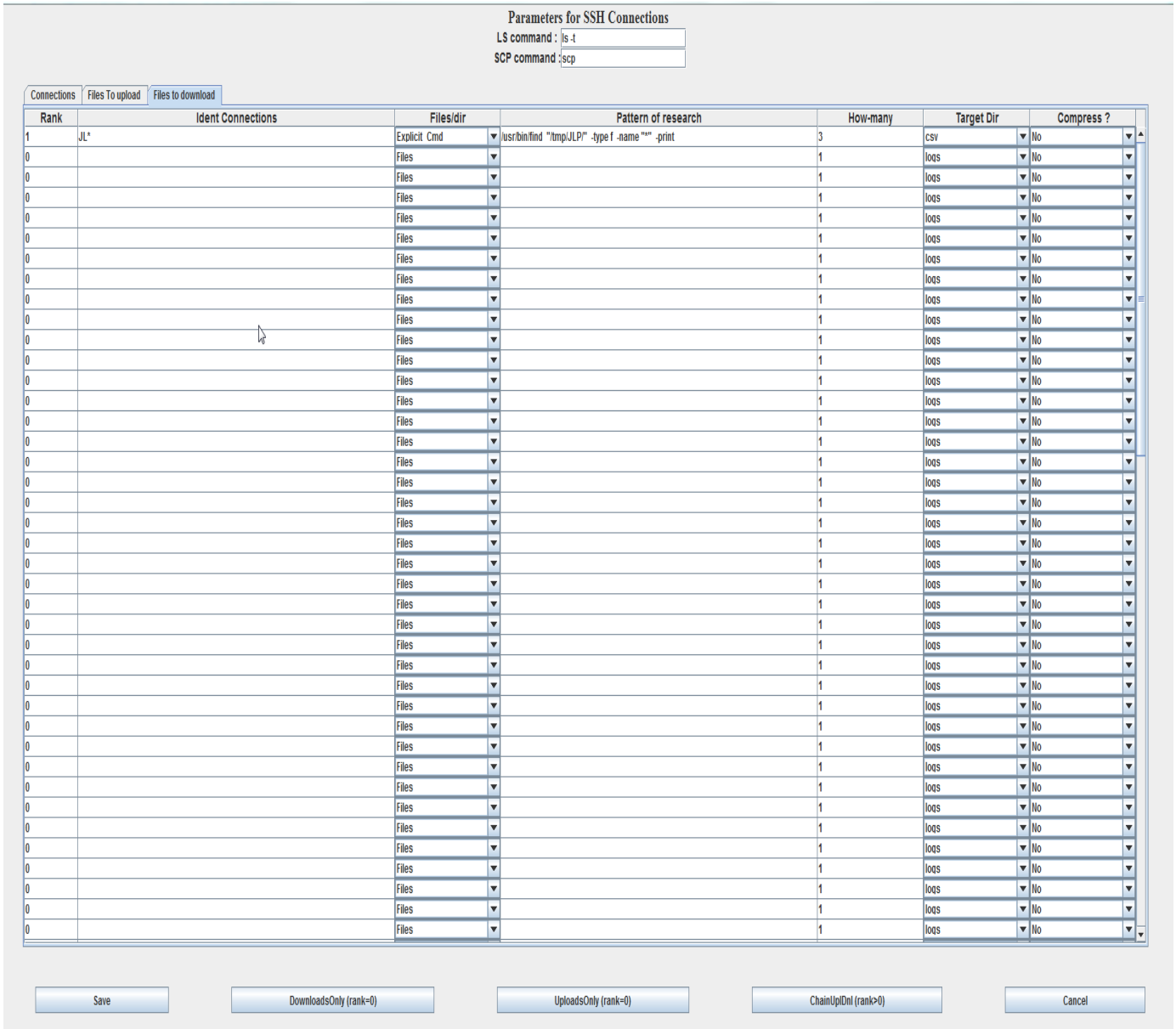
- 0 means that the operation is launched when **UploadsOnly** button is clicked
- -1 means that the line is skipped and not executed.
- > 0 means that this upload is a part of a series of upload/download with this rank (executed only when clicked on ChainUplDnl button)

The second column contains the idents of the connections.
For this second, you can use several pattern :

- allip => upload files/directories to all different ip addresses defined in the tab connexions
- * => upload files/directories to all different servers defined in the tab connexions
- server1 server2 => a list separated by a space character upload to the servers on the list
- serv* => upload to the servers which the name begins by serv

The third column contains the file to upload (by right clicking on this column, there is a File chooser that points to the uploads folders that contains pre-defined shells).
Also the pattern <server> is usable.
The fourth column contains the name of the folder of the remote server where the file will be uploaded (the last character must be the file separator as for example **/tmp/**)
The fifth column indicates if you want to execute the shell, after it was be uploaded.

Note : Before downloading files or directory, verify that you have created a new sub-folder i scenario by the Menu : **Add dir CSV/Logs** (see further in the document)



The third tab-panel permits to define the files or the directories that we want to download.
The first column (rank) has the same behaviour of the Upload tab:
The first column contains the rank of the operation :

- 0 means that the operation is launched when **DownloadsOnly** button is clicked
- -1 means that the line is skipped and not executed.
- > 0 means that this upload is a part of a series of upload/download with this rank

(executed only when clicked on ChainUplDnl button)

The second column is the key of a server (corresponding to the first column of the first tab-panel)
For this first colum, you can use several pattern :

- allip => download files/directories from all different ip addresses defined in the tab connexions
- * => download files/directories from all different servers defined in the tab connexions
- server1 server2 => a list separated by a space character dowload from the servers on the list
- serv* => download from the servers which the name begins by serv

The third column sets the type of object to download (Files ,FilesNoPrefix,Directory, Explicit_Cmd).

- Explicit_Cmd is used when you have to choose files that can't be reached by ls (example with command find to get files with a minimum size or located in different branches of a directory)
- FilesNoPrefix doen't prefix the downloaded file by the name of the server

The fourth column set the name of files or directories to download. The joker character * is now allowed everywhere in the string (beginning with version 1.1.2). You can use also the pattern <server> in the path of the file on the path of the directory, it will be replaced by the corresponding name of the server.

The firth column indicates how-many most young objects we want to download.

The sixth column is the choice of the target folder where the download will occur (logs in directory logs of the scenario, csv logs in the csv directory of the scenario, reports in the reports directory)

The seventh column indicates if you want to gzip the file before downloading

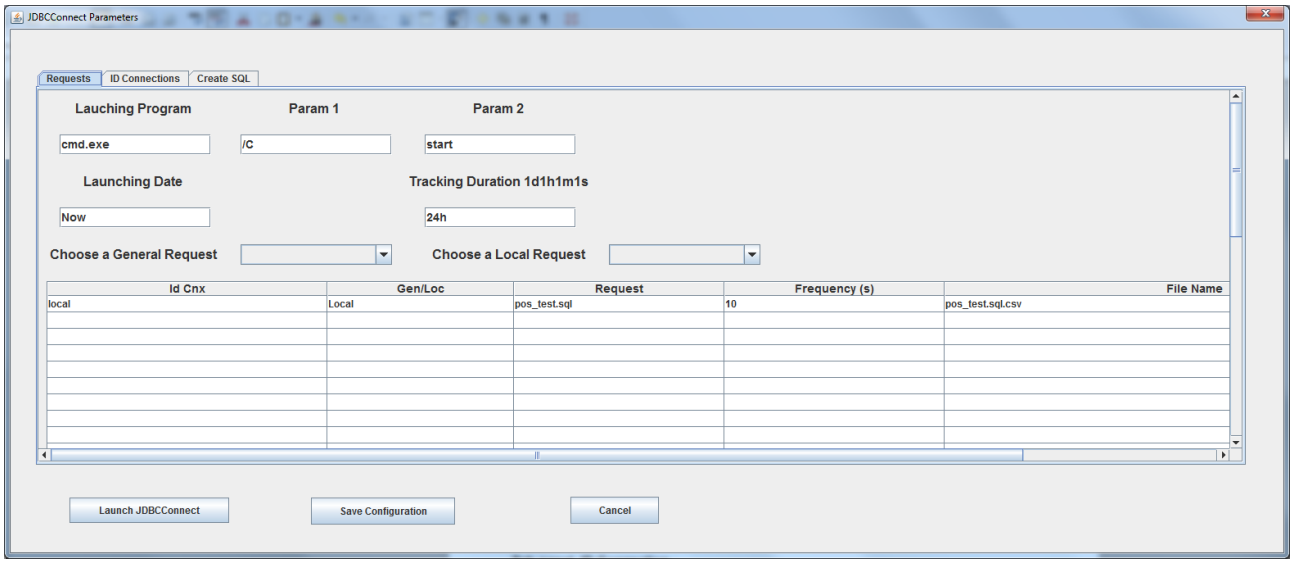
3.1.4 JDBC Requests

The sub-menu **JDBC Requests** allows to periodically request databases of an application through a JDBC connections (When it is possible, Firewall or database listener can forbid the access)
The supported databases are :

- Oracle
- Sybase
- SQL Server
- MySql
- PostgreSQL

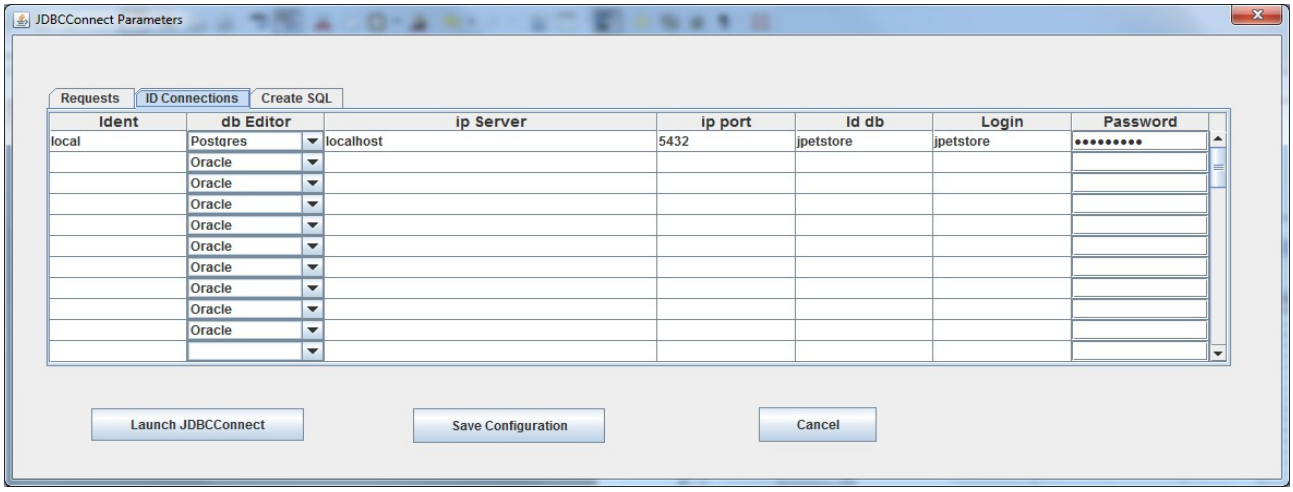
As said, more above, the Oracle JDBC Driver and MySql JDBC Driver must be downloaded from the Web and put in the libExt directory.

The screen that configure JDBCConnect are given below :



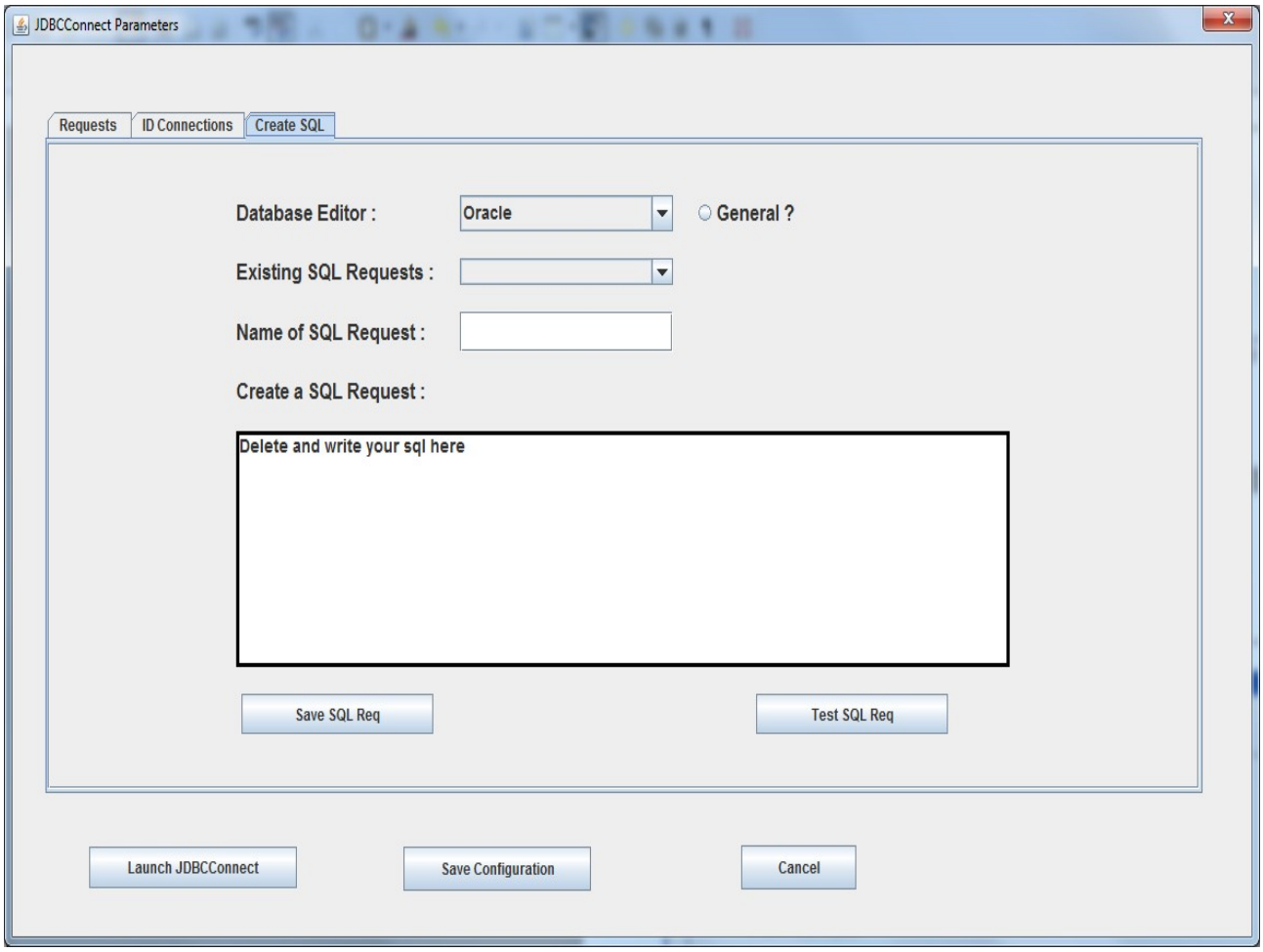
In this tab-panel, we set the parameters to launch JDBC Requests, that can also run after stopping **swingScaViewer.**
The two combo-boxes permit to chose existing requests (see also how to create a SQL request below). The requests are send to the IDConnexion (see the corresponding tab-panel).

Tab-panel ID Connection :

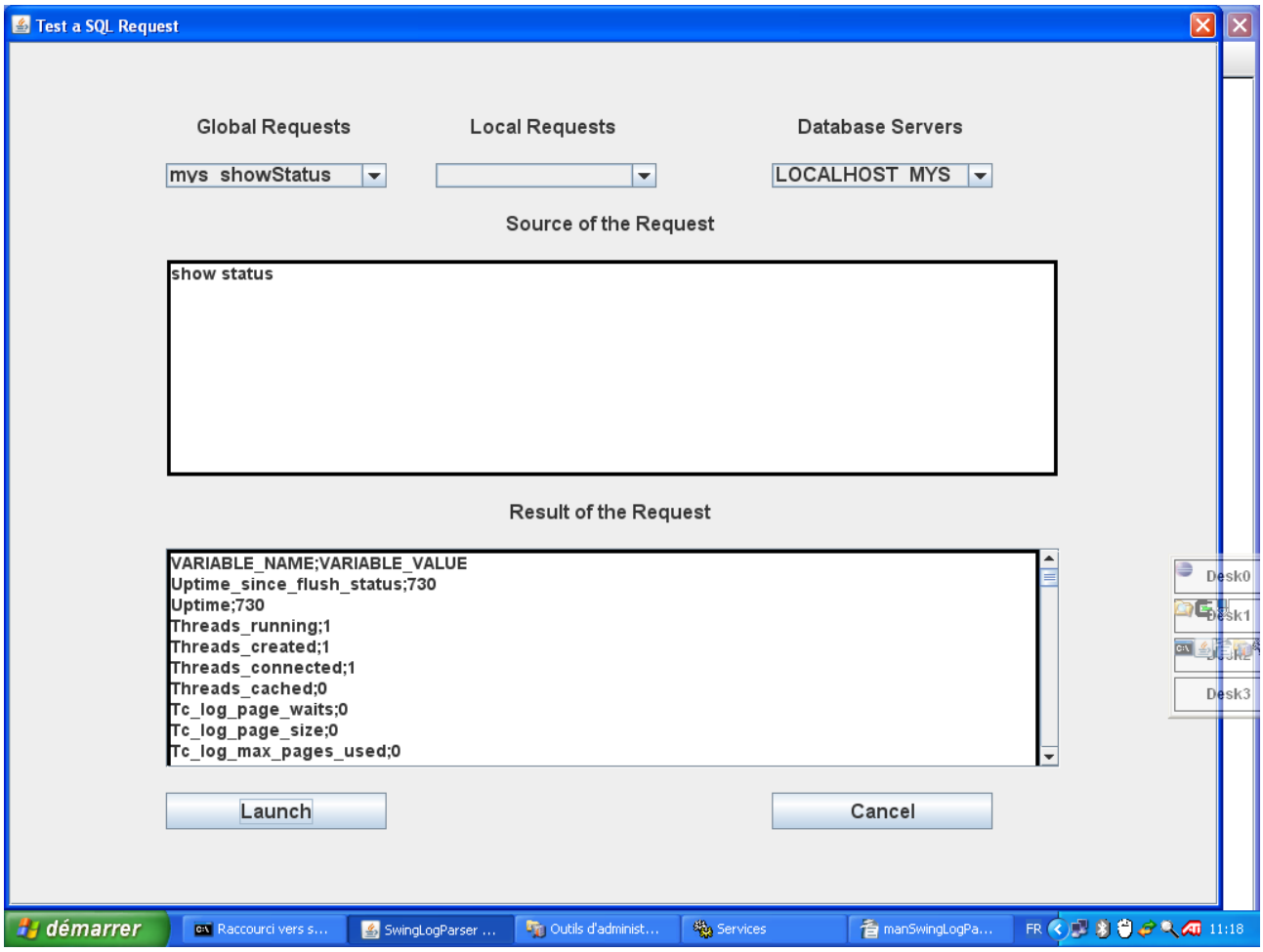


In this tab-panel, the parameters needed by a JDBC connection must be set

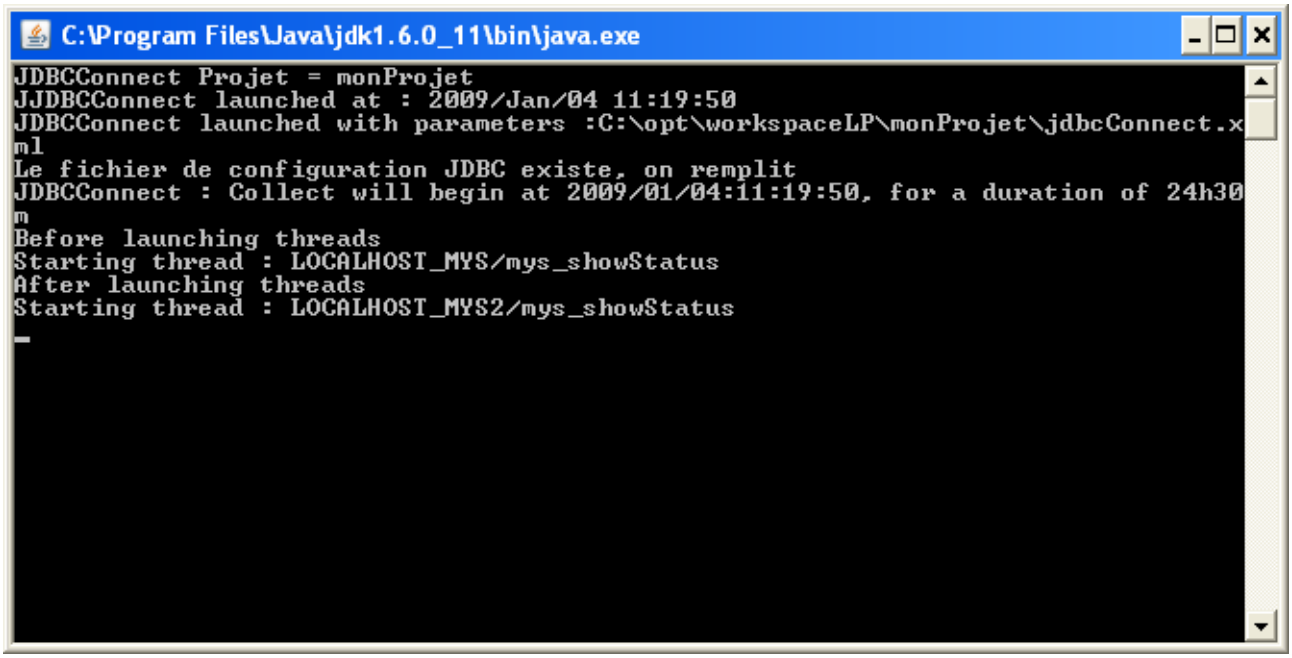
Tab-Panel Create SQL



This tab-Panel permit to create and test SQL requests.
In the above example, the request **show status** toward a MySQL database gives a complete state of the database.
If we do a unit test, for this request, we obtain the result shown below :



The launch of JDBC Requests with the parameters above gives the Windows Console command below :



The result files are created in the most recent logs directory. The files are generated in gz format. So you have to kill the Windows cmd console to generate a correct gz final file.

Example with the parameters set above :

```
Time;VARIABLE_NAME;VARIABLE_VALUE
2009/01/04:11:19:51;Uptime_since_flush_status;847
2009/01/04:11:19:51;Uptime;847
2009/01/04:11:19:51;Threads_running;2
2009/01/04:11:19:51;Threads_created;2
2009/01/04:11:19:51;Threads_connected;2
2009/01/04:11:19:51;Threads_cached;0
2009/01/04:11:19:51;Tc_log_page_waits;0
2009/01/04:11:19:51;Tc_log_page_size;0
2009/01/04:11:19:51;Tc_log_max_pages_used;0
2009/01/04:11:19:51;Table_locks_waited;0
2009/01/04:11:19:51;Table_locks_immediate;18
2009/01/04:11:19:51;Ssl_version;
2009/01/04:11:19:51;Ssl_verify_mode;0
2009/01/04:11:19:51;Ssl_verify_depth;0
2009/01/04:11:19:51;Ssl_used_session_cache_entries;0
2009/01/04:11:19:51;Ssl_sessions_reused;0
2009/01/04:11:19:51;Ssl_session_cache_timeouts;0
2009/01/04:11:19:51;Ssl_session_cache_size;0
2009/01/04:11:19:51;Ssl_session_cache_overflows;0
2009/01/04:11:19:51;Ssl_session_cache_mode;NONE
2009/01/04:11:19:51;Ssl_session_cache_misses;0
2009/01/04:11:19:51;Ssl_session_cache_hits;0
2009/01/04:11:19:51;Ssl_finished_connects;0
```

The dates in the file permit to directly chart it with the Menu Viewers (see further in the document)

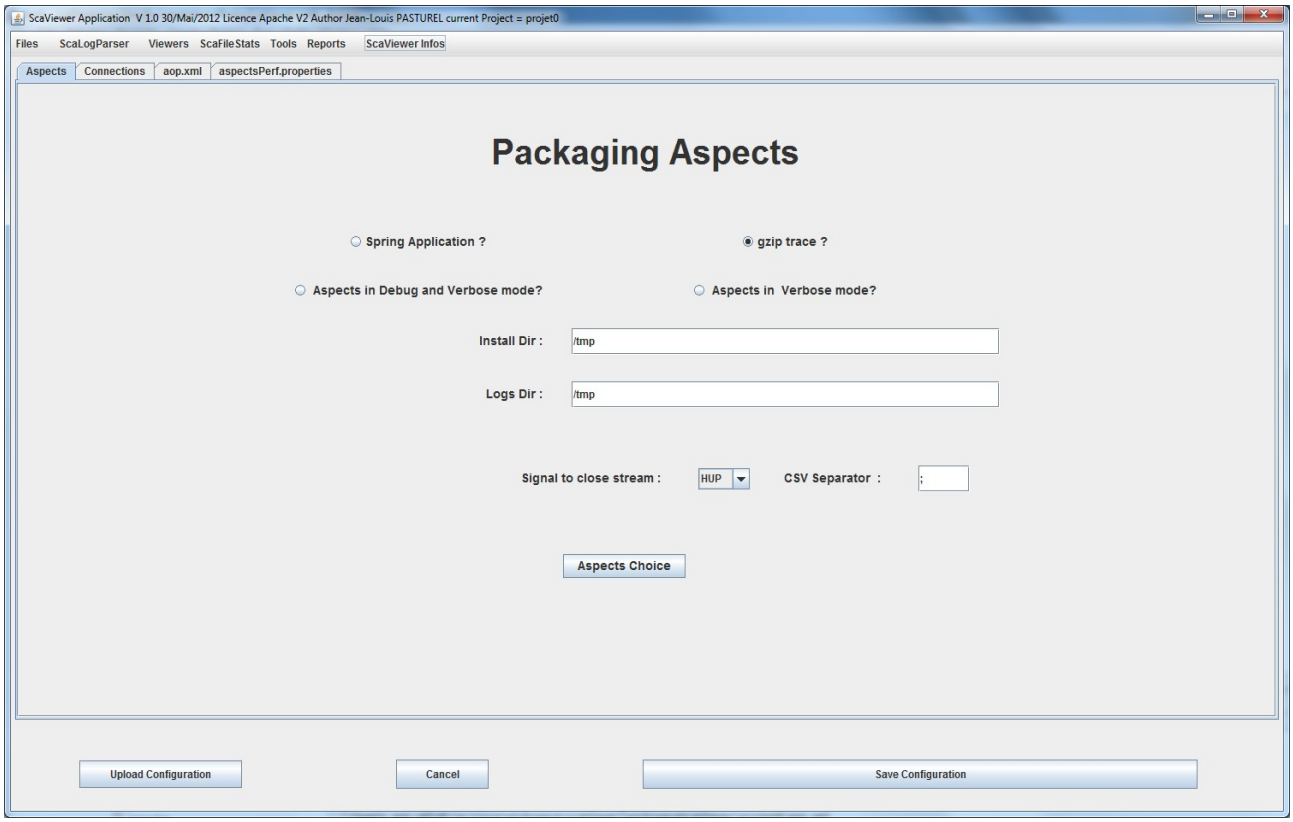
3.1.5 Perf with AspectJ/Java Agent

Advanced use for profiling JAVA application

The sub-menu **Perf with AspectJ/Java Agent** permit to package in a single jar file an Aspectj java agent (**myaspectjweaver.jar**) that contains:

- the Load Time Weaver **aspectjweaver** (LTW) standard of Eclipse project AspectJ
- The Spring-agent **spring-agent.jar** (LTW) used for certain forms of weaving with Spring (it is recommended to read the Spring AOP docs before)
- The collection of Aspectjs of the tool **AspectsPref**
- The standard configuration file **aop.xml** of AspectJ
- the configuration file **aspectsPerf.properties** of the **AspectsPerf** tool.

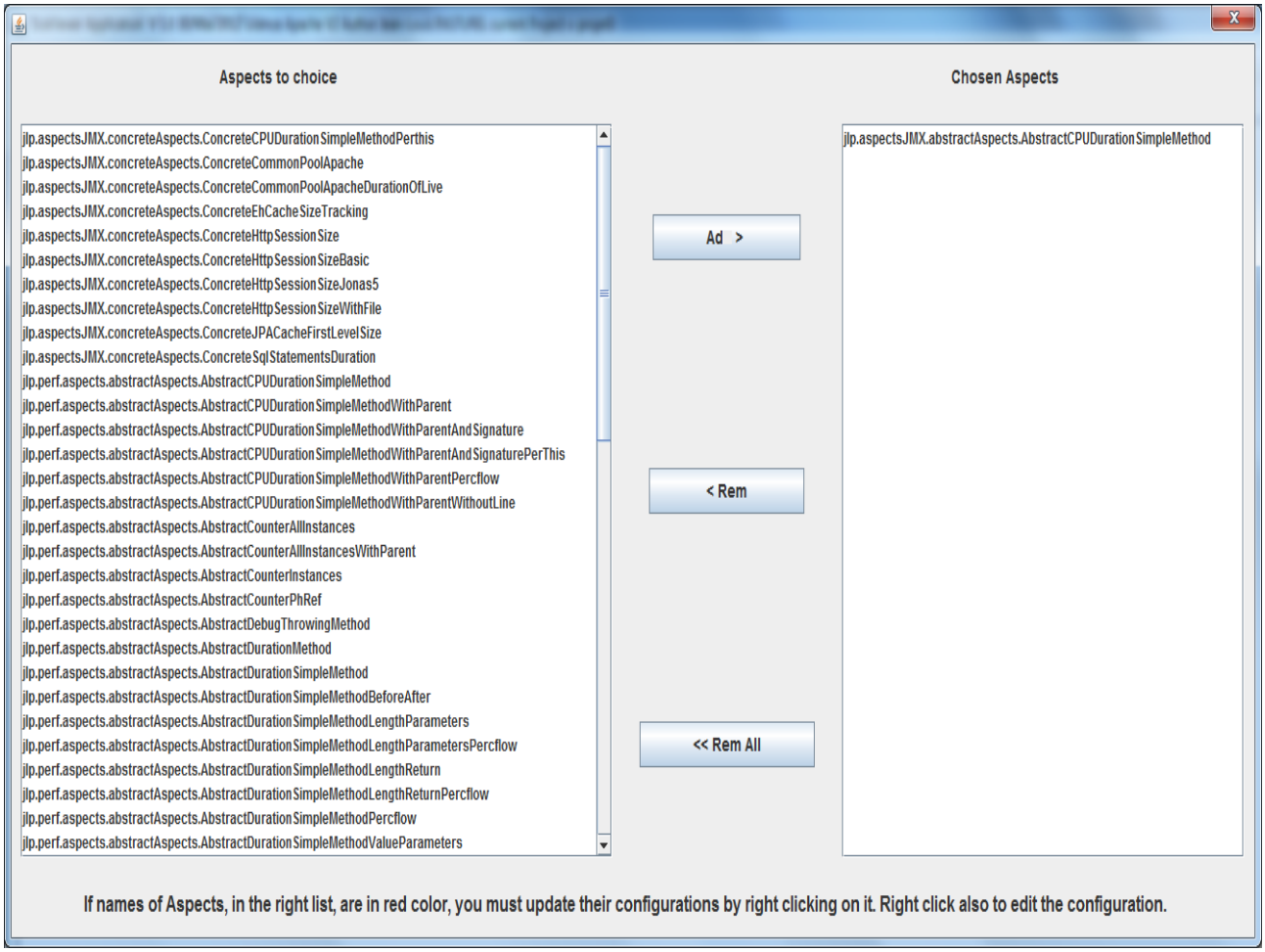
The interest to make a single jar is that the aspectj are in in the classpath at the start of the JVM. Some Aspects may need to add some other jar at the bootclasspath. For example to profile Jolt, Axis, CXF ...
The tool permits, when the packaging is ready to upload it to the remote server.
The screen involved in this packager are given below :



The tab-panel Aspects defines :

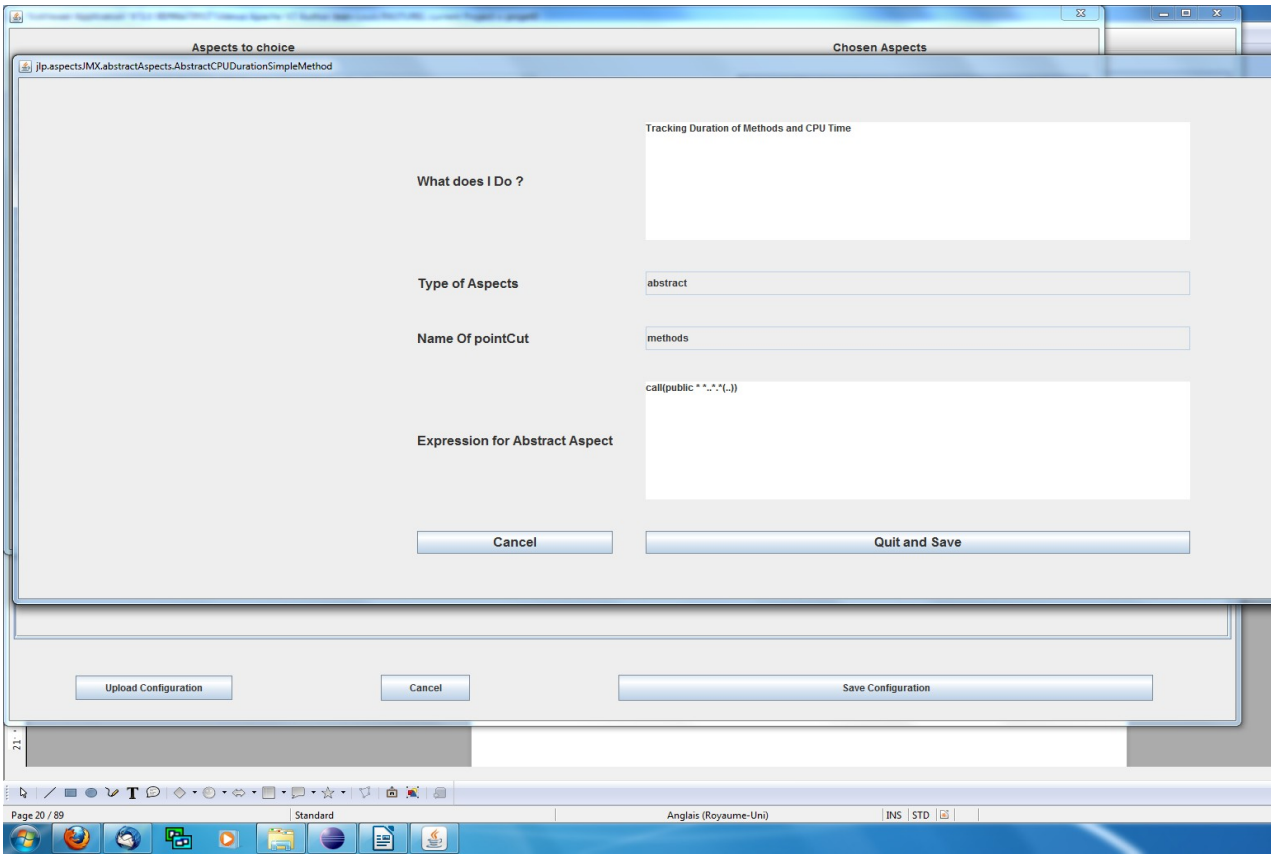
- what kind of agent weaver to use (Standard AspectJ or Spring, standard advised)
- the gzipped or not of the logs generated
- The debug/verbose mode of Aspectj
- the directory of the remote server where to upload myaspectjweaver.jar
- the directory of remote server where the logs will be generated
- the system signal (Unix only, HUP advised) to close correctly the gzip file and rotate the logs

- A Button that permits to choose an Aspect between a collection that can be extended
- Below different parameters linked at the aspect :
- The definition of the pointcut for abstract Aspectj, that define the scope (package/classes) on which the Aspect is woven. The docs of AspectJ explains all the syntax of the pointcut.(http://www.eclipse.org/aspectj/doc/released/progguide/starting-aspectj.html#pointcuts)



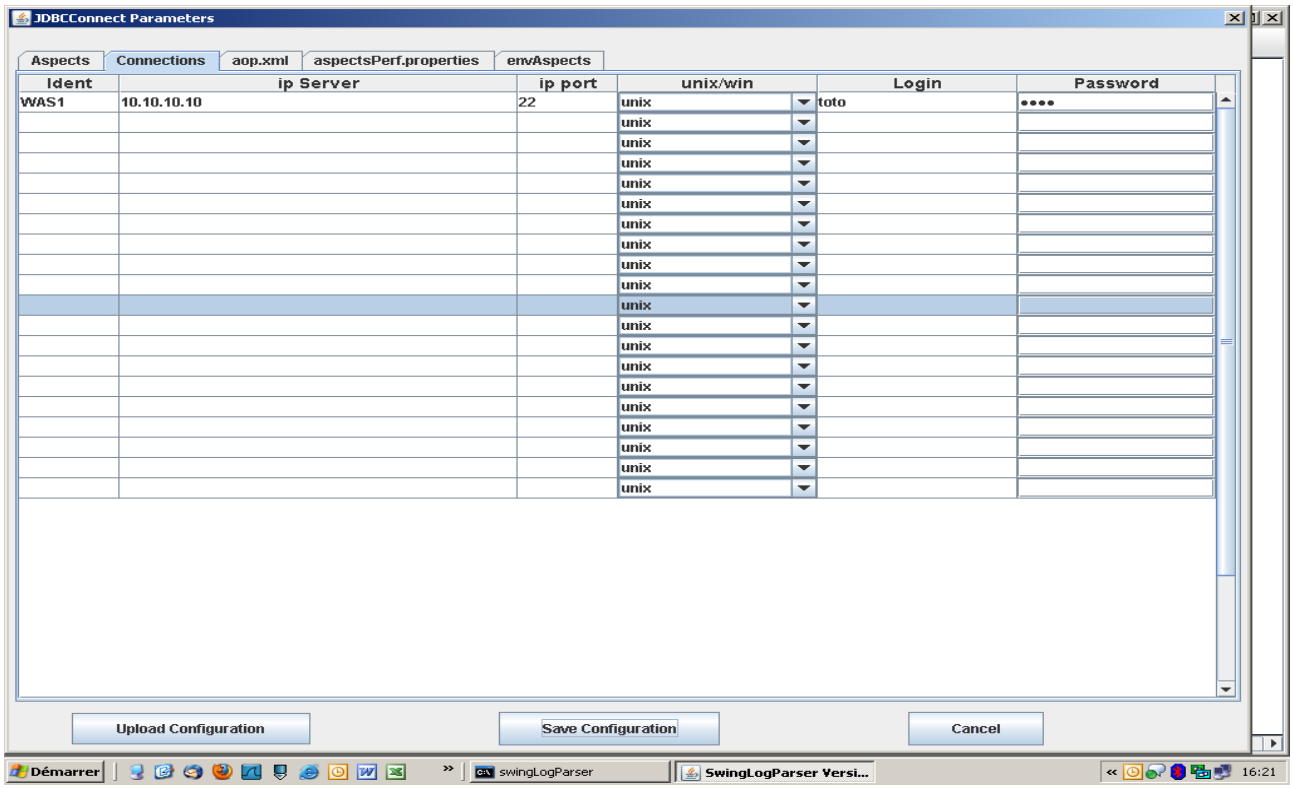
The Panel before permit to choose one or more Aspects.

For each aspect, by right clicking in a selected aspect, you can configure it as shown below :



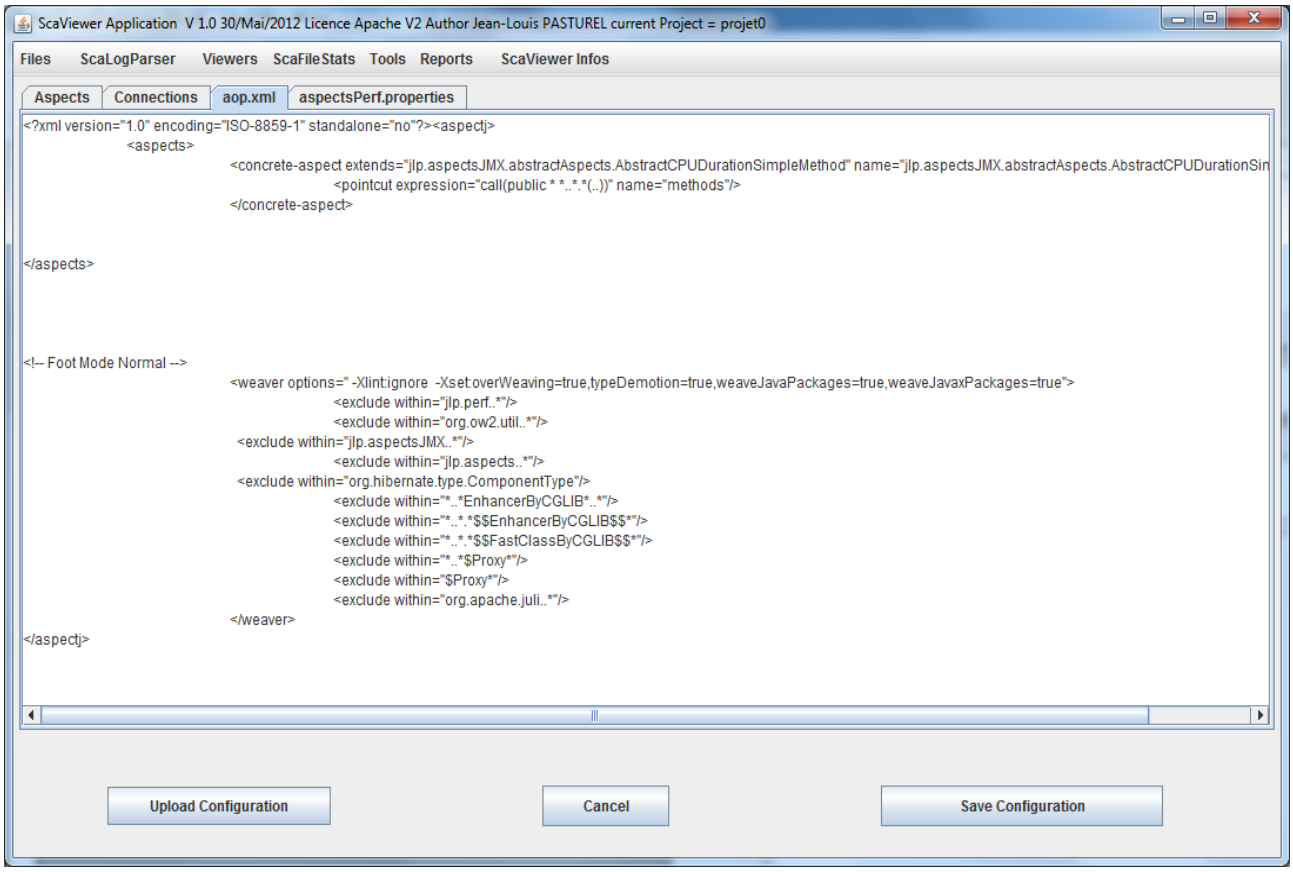
Every event on the button Quit and Save modifies the file **aop.xml**, located on the third tab-pan*el.*

Tab-Panel Connections :



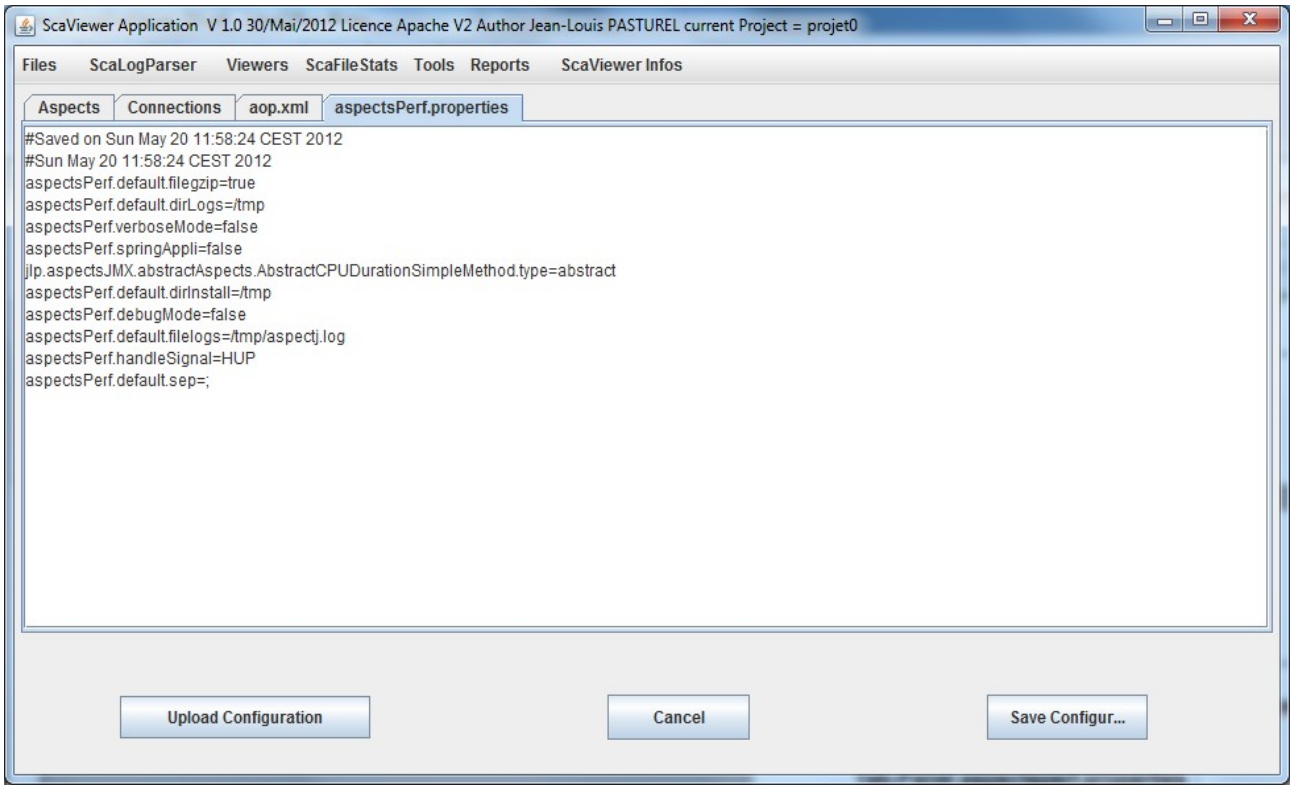
This tab-panel defines on which servers the agent must be uploaded

Tab-Panel aop.xml :



This tab-panel shows the contents of the file **aop.xml**. (read-Only access, modifications are not saved)

Tab-Panel aspectsperf.properties :

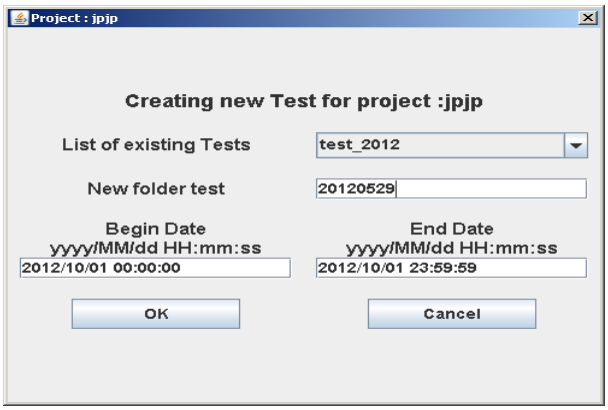


This tab-panel shows the contents of the file **aspectsperf.properties**. (read-Only access, modifications are not saved)

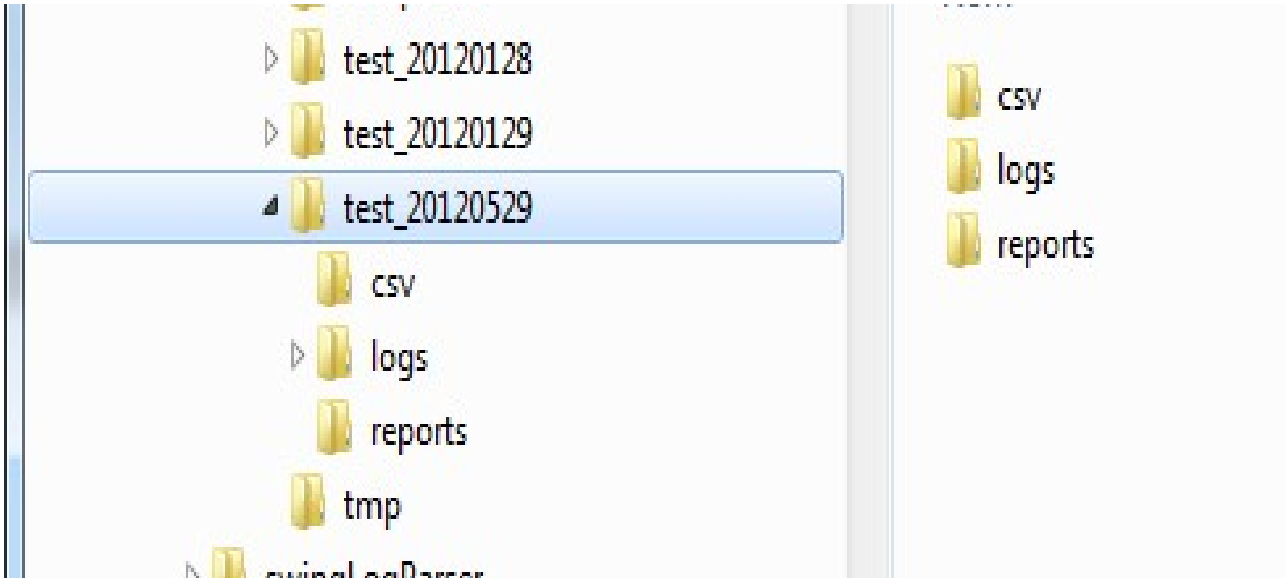
Note :
The library of Aspects is extensible, but I have not an available documentation. If somebody is interested by the source code of AspectPerf, I can send it by mail. My mail address is given at the beginning of this documentation

3.1.6 Add Directory Logs/CSV

The sub-menu **Add Directory Logs/CSV** permits to structure the directory of the project in scenarios or date of tests.
The scenario or test name is what you type in the Dialog Box below.
Important : When you create a New Project, you must also create a first scenario.



the new directory is created as shown below :
%workspace%\projet0\test_20120529
Starting with version 1.2, there is a begin / end date for the current scenario. The fields can be updated. It increases performance when parsing large files, which the the logs are not all useful.



In the sub-directory **logs** of the new folder (**test_20120529**), we put all logs files corresponding to the new scenario . As see more above, the automatic download help you to handle the logs files from the remote server.

Note : to remove a project, remove the directory of the project .

3.2 Menu "ScaLogParser"

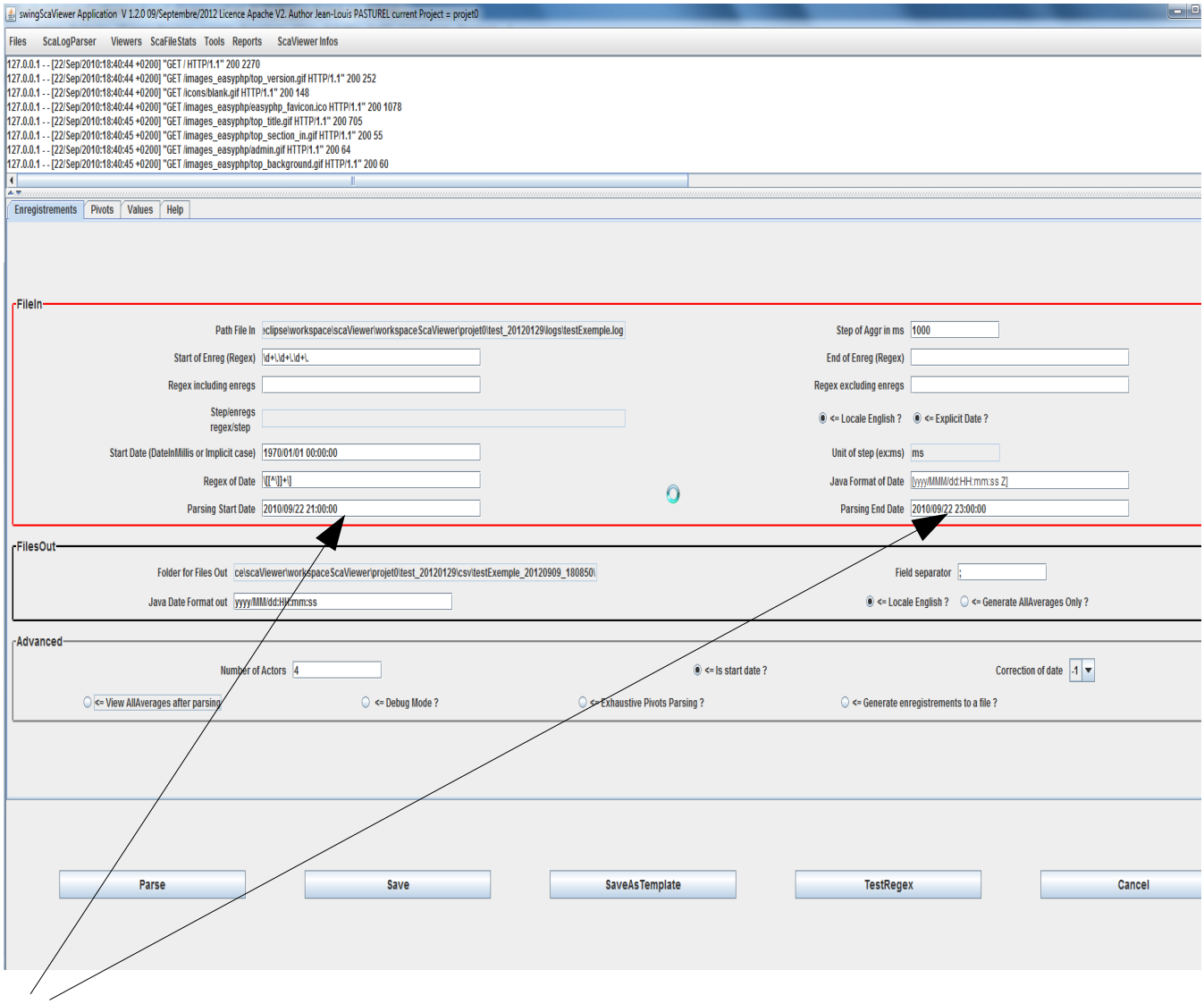
With this menu, we transform a log file to a csv file that can be graphed with JFreeChart (Menu

Viewers)
The Menu ScaLogParser (in fact a Button) open a customized FileChooser in the directory %workspace/<Project><last_scenario>/logs
You have to choose a file.

There are 3 possibilities to configure before parsing :

- with no template, (ex-nilho) => ComboBox empty
- with a local template, only valid for the current project General Template unselected and a template chosen in the ComboBox
- with a general template, shared by all projects General Template selected and a template chosen in the ComboBox

The screens below, show the more general configuration with no template
After clicking on OK Button, the screen below appears :



Parsing Start Date / Parsing End Date : new in version 1.2.0 to set a beginning and an end date for parsing. It is useful when log files are large. See the table below for the format.

In the TextArea at the top of the Windows, there are the first 20 lines of the log file to parse

In the bottom part, there is a Tabbed Pane with 4 tabs

The first tab has the functions :

- to define what is a record in the source file (Beginning/End of record, Locale …)
- to retrieve the date of the record (explicitly or implicitly).
- to define the format of the output CSV file(Date format, separator)
- Automatic graph or not after parsing

File In parameters :

Parameters	Signification	Example
Path File In	Full path of the source file**.** Automatically filled by swingScaViewer when chosen	
Step of Aggr in ms	Measurement period of result aggregation in milliseconds.	1000
Start of Enreg (Regex)	Regular expression (Perl Regex) to find the beginning of the record. **Mandatory.**	**\d{4}-\d{2}-\d{2}**
End of Enreg (Regex)	Regular perl expression to catch the end of a record. If this field is empty, it signifies that each line is a record.	If there is no evident end of record pattern, you can set it the same pattern as the pattern of the beginning of the record. swingScaViewer takes care not to skip any record.
Regex including enregs	Regular perl expression that permits to include all records that are interesting.	Exclude feature has priority over Include feature.
Regex excluding enregs	Regular perl expression that permits to exclude some records that are not interesting.	
Step within enregs or regex for step	Used when **Explicit Date** is unselected. Define the increment relative to the parameter **fileIn.startDate** under 2 ways. => **regexp** a perl regex **val** , a value in between 2 records (unit is given by **Unit of Step)**	**(?![^=]=)\d+** **val=1000** **(*)** See more explanation after this table.
Locale English	Choice of the Locale for File In (for Date and representation of Double)	Selected (as possible)
Explicit Date	If selected, means that every record of the file contains an explicit date. If not (Implicit) the date must be computed.	Selected (as possible)
Start Date (DateInMillis or Implicit case)	Definition of a beginning date with the Java DataFormat defined by the field **Java Format of Date**	**1970/01/01 00:00:00** (0 of TimeStamp) Used when **Explicit Date**

Parameters	Signification	Example
		is unchecked, if so an increment must be defined. or when **Regex of Date** is set **to dateInMillis** with explicit dates.
Unit Of Step	Unit of the increment. Possible values are **s,ms,micros,nanos**	**Ms**
Regex of Date	Regular expression (perl regex) to catch the date. The date must be in the first selected group. In case of date in timestamp in millis the pattern to set is **dateInMillis**.	**^([^:]+[^]+)** **if dateInMillis** in **java date format,** you can put here two regex separated by a blank to extract the date in millis (the 2 regex must not contain a blank)
Java Format of Date	Select the Java DateFormat of the records of the log file.	**[dd/MMM/yyyy:HH:mm:ss Z]** or **dateInMillis or dateInMillis,<mult>** mult=1000 if date is in seconds The dateInMillis format, compute date, starting with the beginning date filled in the **Start Date (DateInMillis or Implicit case)** parameter
Parsing Start Date	The date for the beginning of parsing	Format must be : **yyyy/MM/dd HH:mm:ss**
Parsing End Date	The date for the end of parsing	Format must be : **yyyy/MM/dd HH:mm:ss** If beginning date >= end date, there is no control for the date. So with the two dates set by default to 1970/01/01 00:00:00, there is no control of date.

File Out parameters :

Parameters	Signification	Example
Folder for Files Out	Full path of the target directory files**.** Automatically filled by swingScaViewer when chosen	
Field separator	Separator for the output CSV file	;
Java Date Format out	Format DateFormat Java of the csv generated file	**yyyy/MM/dd HH:mm:ss**
Locale English	Choice of the Locale for File In (for Date and representation of Double)	Selected (as possible)

Parameters	Signification	Example
Generate AllAverages Only	When selected, only the average of the values parsed are generated on a single file.	The name of the file is **allAverages.csv**

Advanced parameters :

Parameters	Signification	Example
Numbers of Actors	Number of actors (akka Actors) to treat the log file	**2** **or nb cores * 2 (if hyperthreaded)**
Is start date	If selected, means that the date is the date of the beginning of the request	Selected depends of file logs.
Correction to date	When the first value is expressed as a duration the 3 strategies are : if 0 => no modification of the date if 1 => add the duration to the date if -1 => subtract the duration to the date	Select 0 or 1 or -1
View AllAverages after parsing	If checked, at the end of the parsing, the file allAverages.csv is graphed automatically.	**Checked/Unchecked**
Debug Mode	For debugging with a little extract of the log file. There is a log trace in the directory <swingScaViewer_Home>/logs	**Checked/Unchecked**
Exhaustive Pivots Parsing	Useful when 2 pivots have inclusive regex for example GET\s/MyHome/ GET\s/ If not checked, the URLs matching the first regex are not parsed by the second regex	**Checked/Unchecked** If you are sure that regex are all not inclusive, unchecking is faster for parsing.
Generate Enregistrements to a File	Help also for debugging to see if you correctly parse the record of the log file. Useful when the record are spanned in several lines of the file.	**Checked/Unchecked**

The analysis of access-logs is based on the utilisation of regular expressions that are described in the JDK Javadoc. Look at Pattern class :

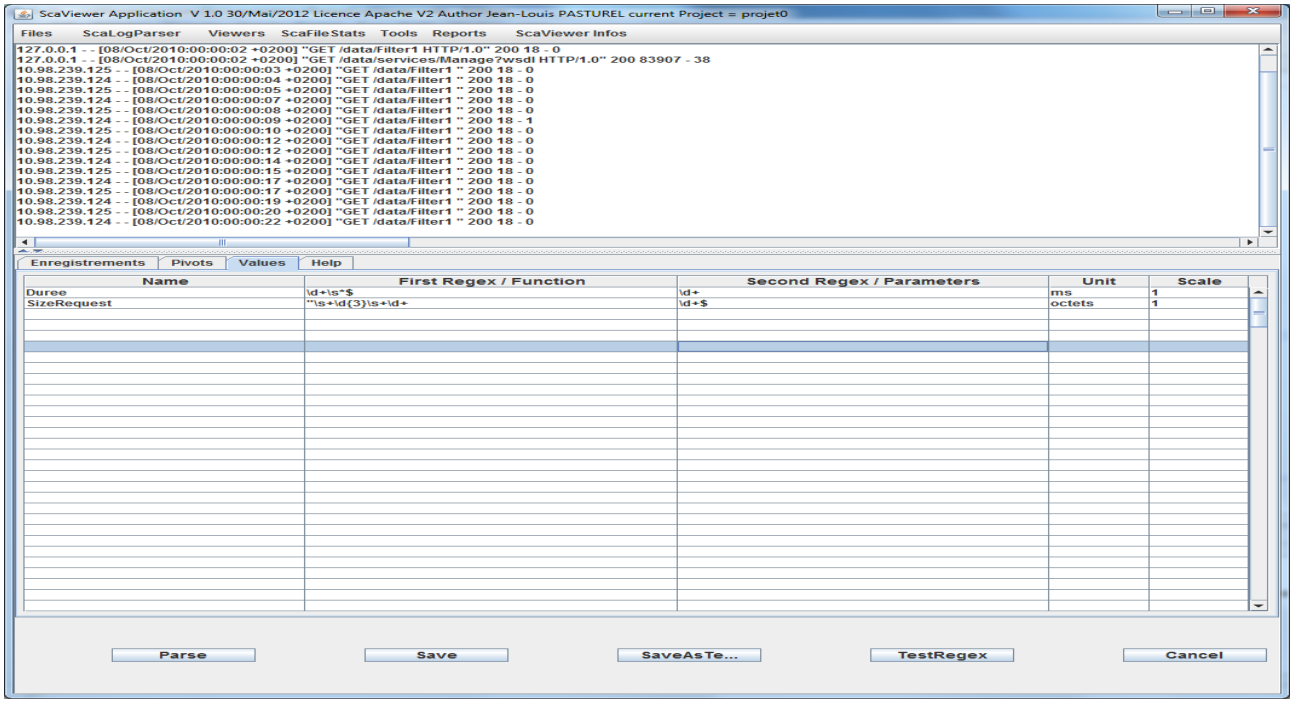
http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html

(*) The different forms of increment are :

- **<regular expression>**
  - Enter here a regular expression to match the increment. The tool test regexp can help you (see later in this document).

- **val=<Constant value of the increment>**
  Specify here the constant value of the increment, the unit is given by the parameter **Unit of Step .** A To be be used when there is no explicit date in the access logs and you know the period of generating every record.

Value tab :



The Value tab is divided in 5 columns :

- **Name Value** : in this column you set the name of the value that you want to follow (the name is free)
- **First Regex / Function** : in this column, you set the mean for extracting the value, there are two ways to extract a value :
  - **regex** (the default shown on the screen-shot above). You put a regex to extract the value. If you are not able to extract the value with this first regexp, you can apply a second regexp(**Second Regex / Parameters**) to extract the final value from the first matching.
  - **function=<nameOfClass>**
    - It is an advanced feature, that permits to write a kind of plug-in to swingScaViewer (useful when the value can't be extracted directly example : sum or difference of 2 values in a record , computing GC throughput...).**(**)** See below the whole explanation of this feature.
- **Second Regex / Parameters**

- A regex when needed as explained above or parameters when using plugins/fuction

- **Unit :**
  - the unit of the value extracted.
- **Change Scale** :
  - permits to change the scale of the value to match the unit chosen.

(**) key word function :
It is an advanced topic and it needs to know programming in Scala language.

We must write a S**cala class** with the default package, which has 2 methods with the given prototypes below :

def metInit(tab:Array[String]=null)
def retour(params:Array[String]):Double

There are examples in the directory **<swingScaViewer_Home>/myPlugins** . The classes (byte-code) must be put in the archive jar **: <swingScaViewer_Home>/myPlugins/myPlugins.jar.** This archive jar is also set in the classpath of the script **swingScaViewer.cmd** (.ksh).

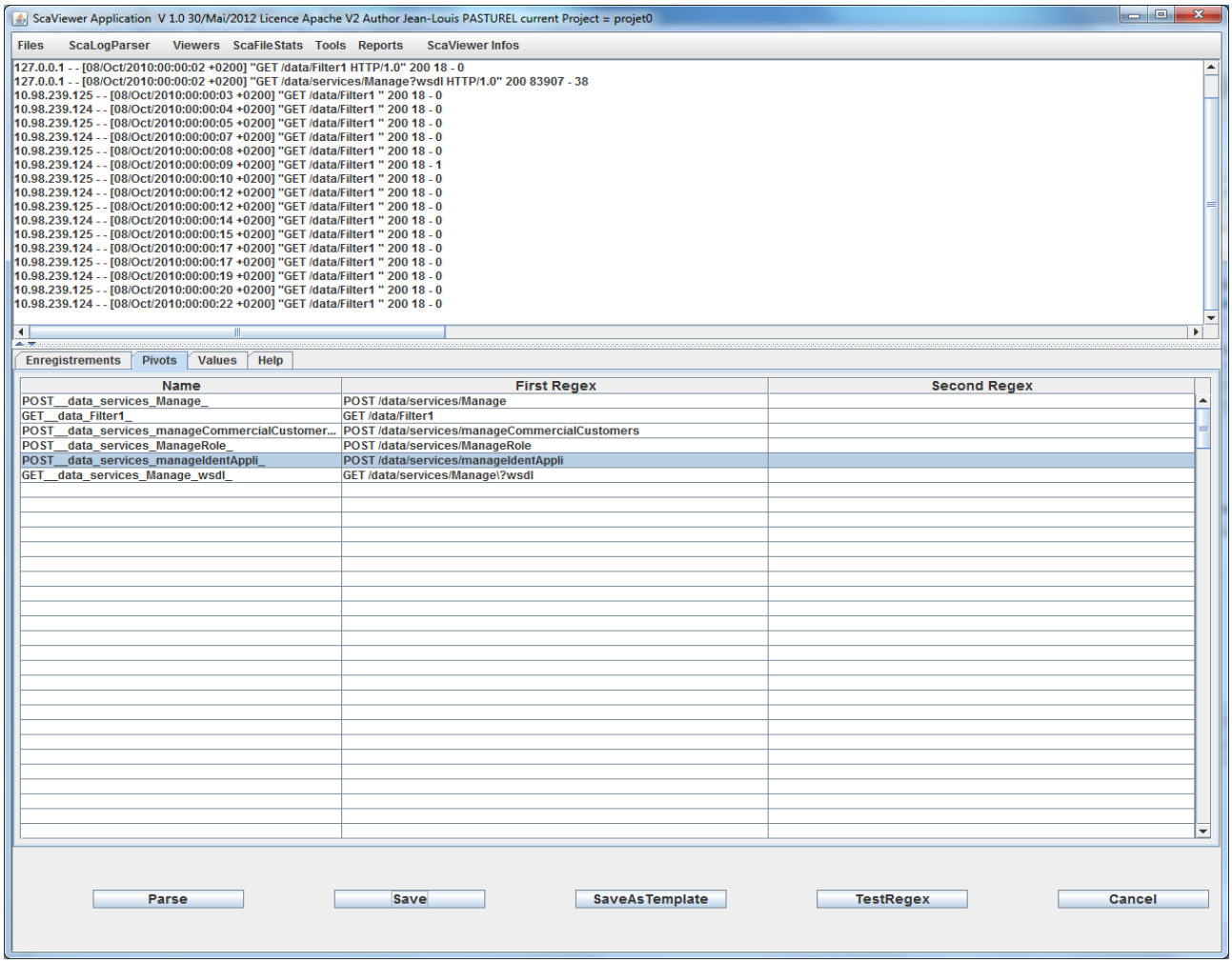
The return value of the method retour is a **Double** (it can be also Double.NaN, this case is treated).
In the column **First Regex / Function ,** the name of the class is set (without the suffix .class) **: function=Add2Values**
In the field **Second Regex / Parameters**, we can set all the necessary parameters to execute the function. The parameters are separated by a separator defined in the first character of the field**.** Be care when there is blank in a parameter to choose another separator than a blank space**.**
Example :

,param1,parma2,param3

, is the separator, param1 to param3 the parameters.

The record is passed by default to the class, ant it is the params(0) of the parameters of the function **retour**. The others parameters are added at params(1) and so on...
Tip :
When the name of a class begins by Conc, the function can be used in multi-threaded parsing. If one class does not begin by Conc in the Value Tab, the parsing is automatically mono-threaded (The field Nb of Actors is set automatically to 1)
Look at the Annexe part an example of function.

Pivots Tab :



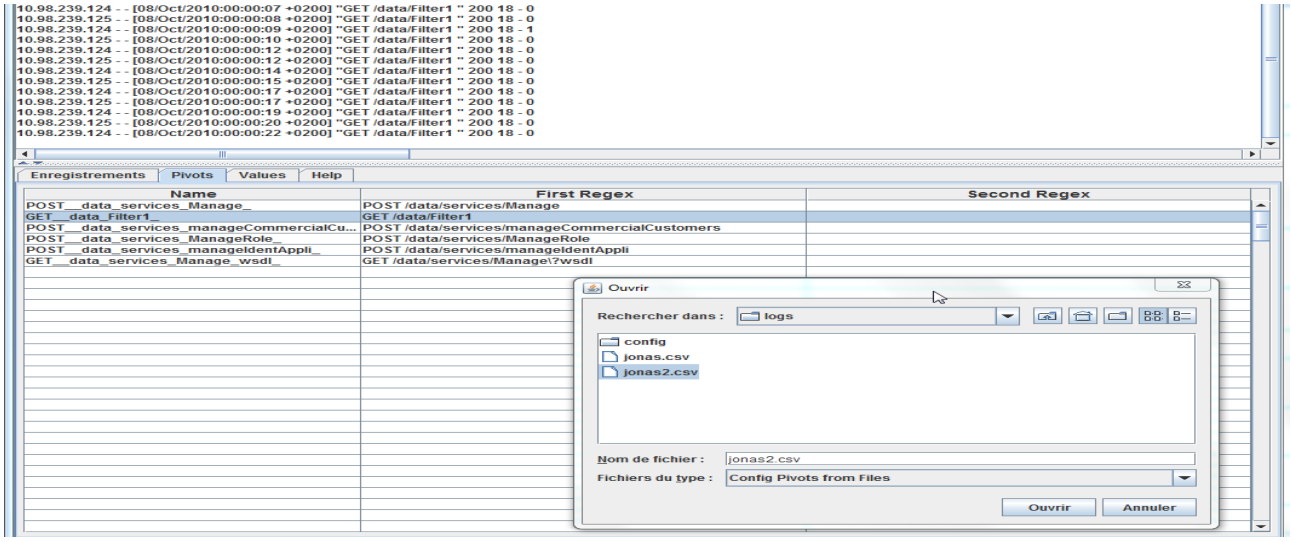
What is a "pivot" ?
In a record, there are other information that are not numeric but permit to sort the values. Example URLs in Apache Access Logs, or other WAS access logs , a **Pivot** can be a specific pattern of URL as shown above.

This tab has 3 columns :

- **Name :**
  - The free name of the Pivot
- **First Regex** : You put a full string value if you can or a regexp to extract the value. If you are not able to extract the value with this first regexp, you can apply a second regex **(Second Regex)**
- **Second Regex** to extract the final Pivot from the first matching.

Hint 1 : If beforehand you have parsed this file with **scaFileStats** (see farther in the document) and

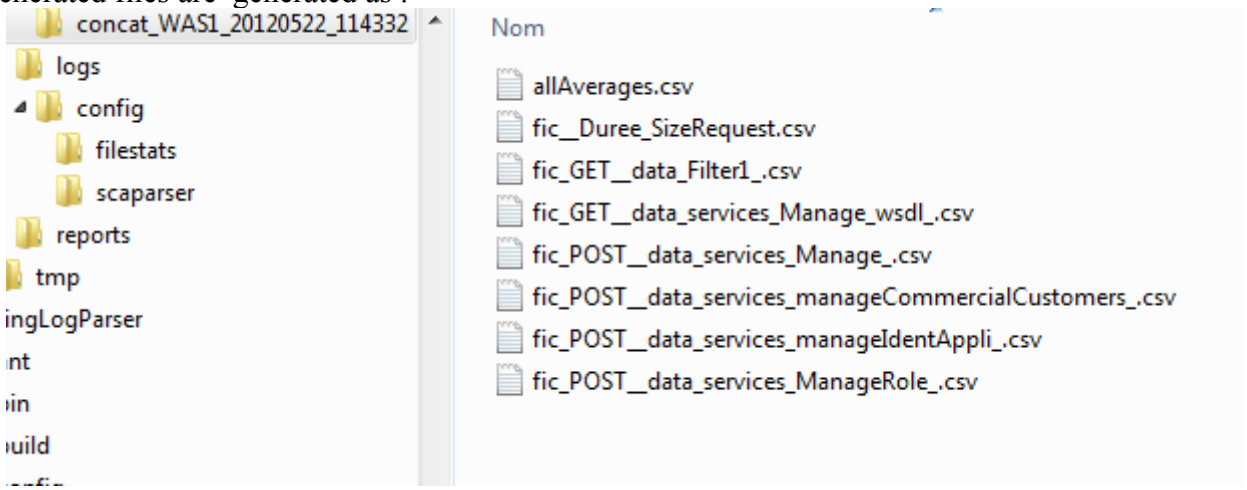
saved the result table in a csv file in the logs directory, by right clicking in the table Pivot and choosing the csv file saved, the rows of the Pivot tab are automatically filled. After you can suppress, modify, insert other rows.



Hint 2 : For the column **First Regex,** if you use full String as possible (without pattern regex as \s\? ...) , the treatment is faster.

Principle of generated files in the csv directory

If you have 2 values in the Tab Value (Duree and SizeRequest) and 6 Pivots in the tab Pivot, the 8 generated files are generated as :



- **fic_Duree_SizeRequest.csv** => global, without Pivot
- The last 6 files are the csv files where each file is bound to a Pivot .
- **allAverages.csv (** Contains the average column of the 7 files above**)**

Help Tab:
To be filled in a future version. It will describe, like above, how to fill the different tabs and fields.

Buttons :

TestRegex:
The button TestRegex permits to test regex in a Dialog box. See the Menu tool for more explanation.

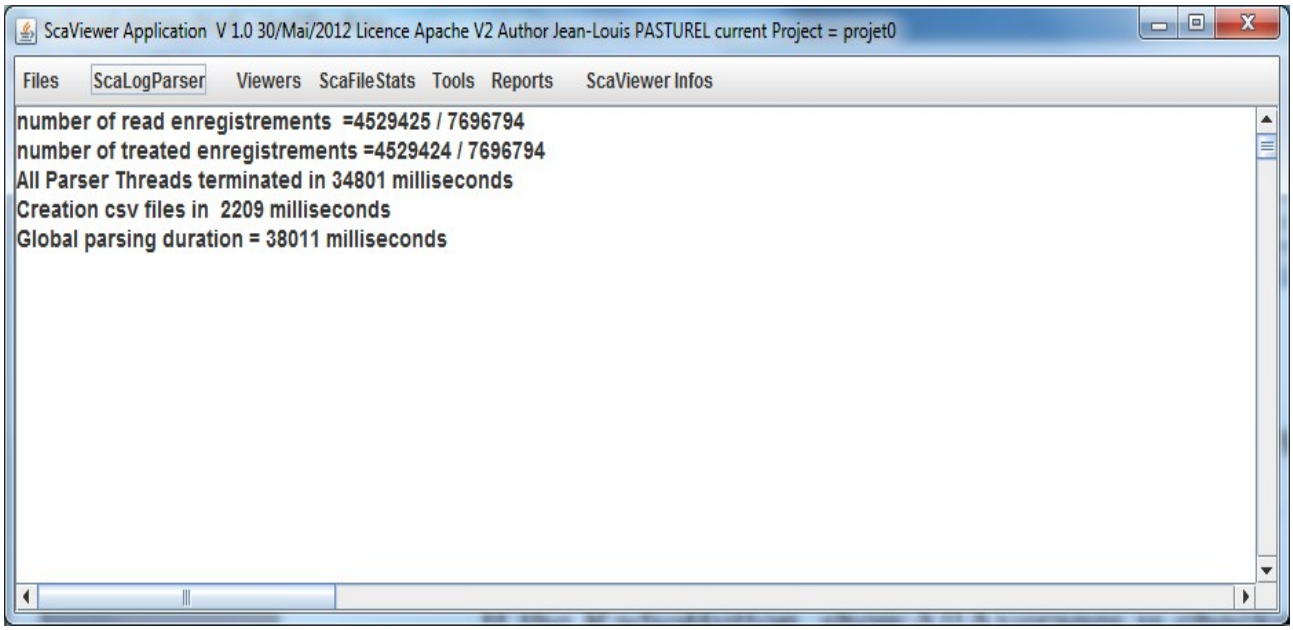
Save :
The button Save saves the configuration and can be be reused for parsing another time

SaveAsTemplate :
The button **SaveAsTemplate** permits to save a template of the configuration that will be used as pattern for the same kind of access log file**.**

We can save this template on the context of the project (**RadioButton General Template ?** Unchecked), or in a General Context (can be used by others projects) when **RadioButton General Template ?** is checked**.**

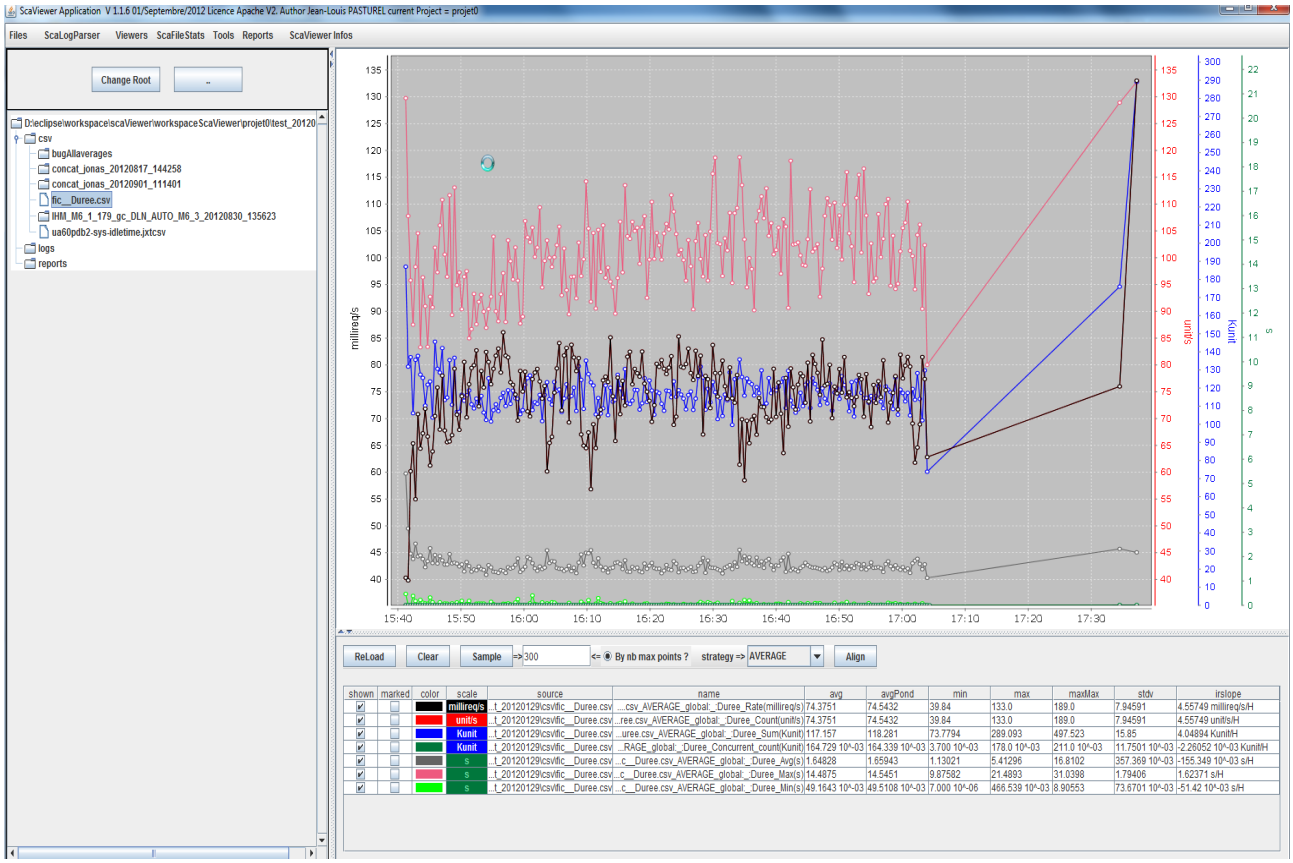
Parse Logs :

The click on Parse Logs starts the Parsing :



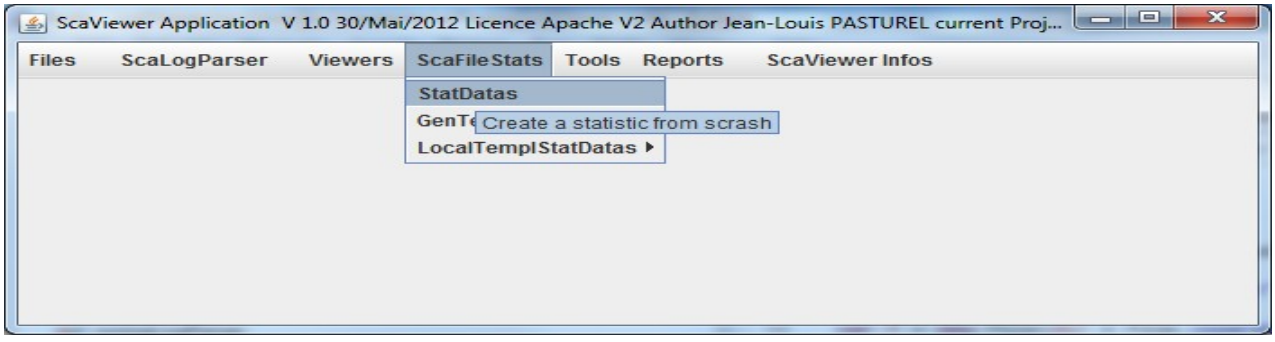
This screen is shown at the end of the parsing, without graphing the **allAverages.csv** file because the radioButton **View AllAverages after parsing** was unchecked.

If the RadioButton **View AllAverages after parsing** is checked before launching the parsing, the **allAverages.csv** is graphed as shown below :



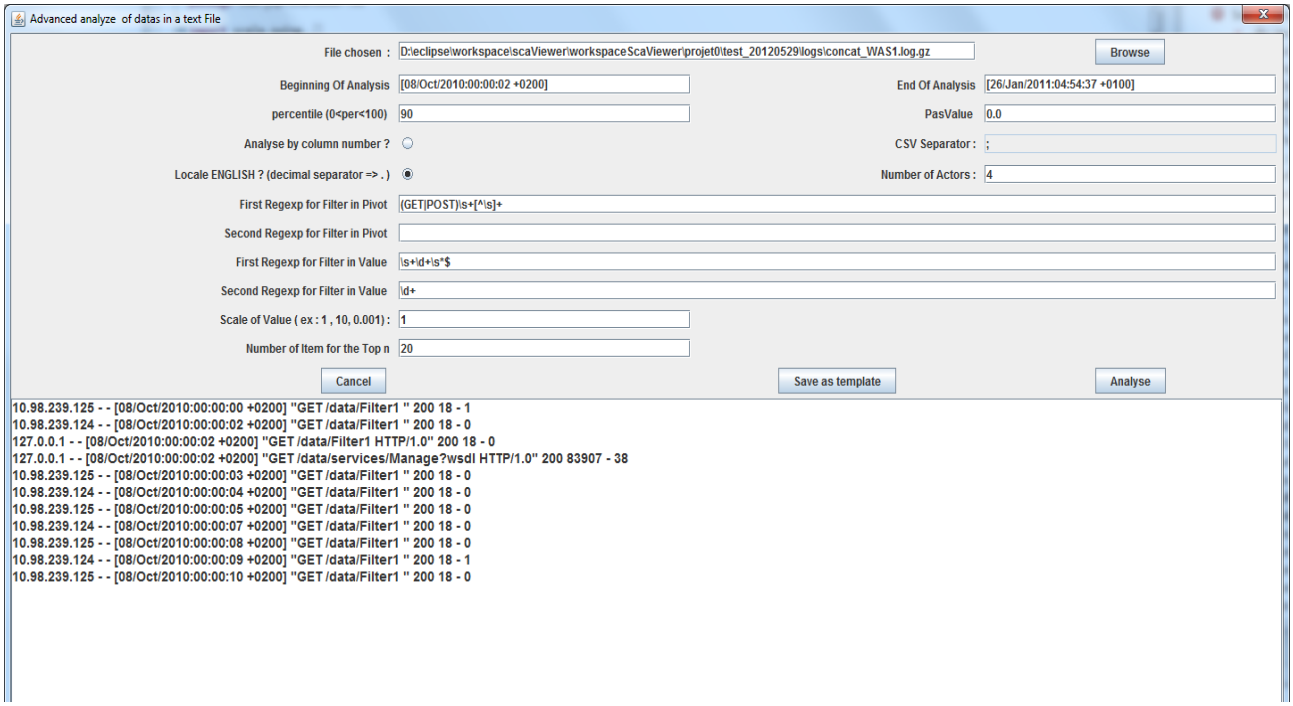
3.3 Menu "ScaFileStats"

To be used, when the records are mono-line. The feature recognize a certain numbers of date Pattern.
If a pattern is not known, this pattern must be added in the file (advanced topic) :
<swingScaViewer_Home>/config/scaViewerDates.properties



3.3.1 Sous-Menu "StatDatas"

After choosing a log file to parse :

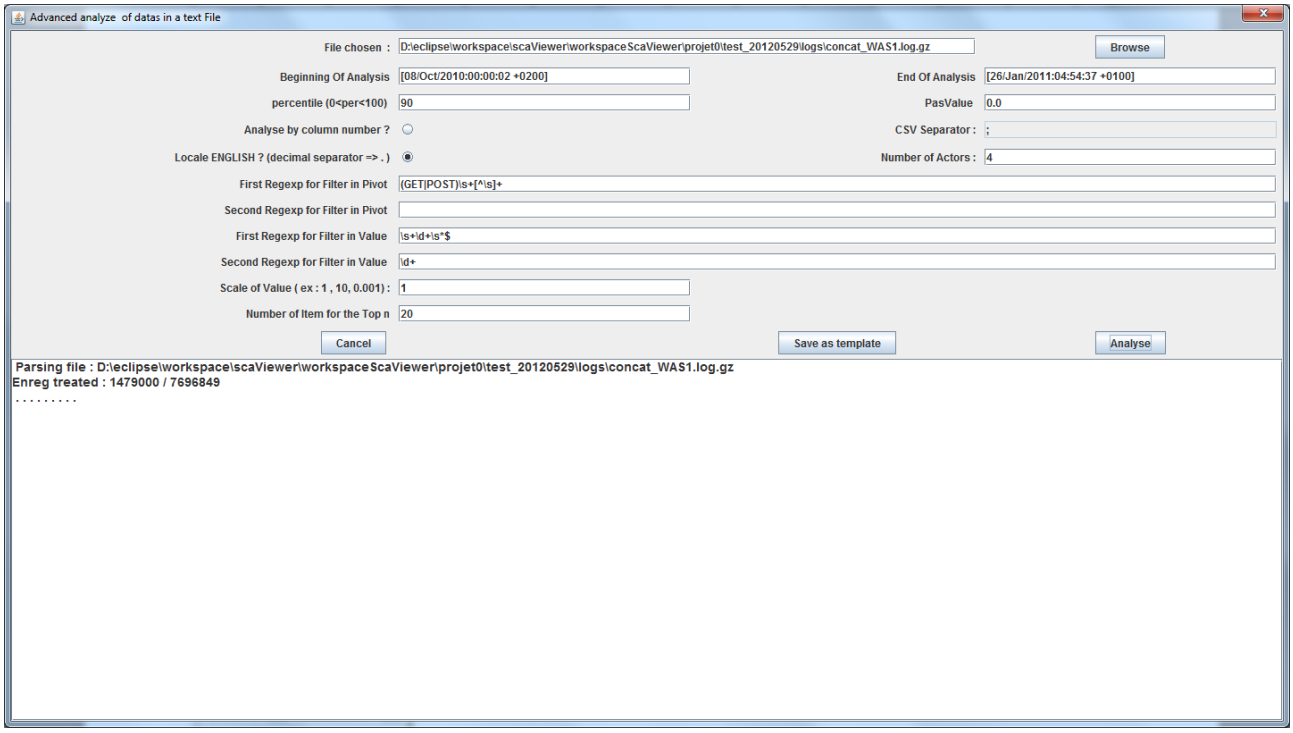


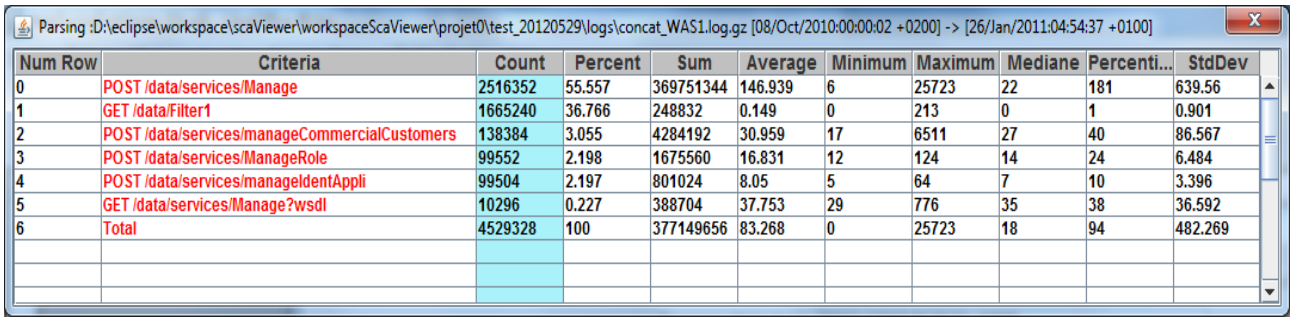
The fields to fill are explained in the table below :

Name of Parameters	Possibles Values	Comments
Beginning Of Analysis	[08/0ct/2010:00:**10:00** +0200]	By default the first date found in the file but can be modified by hand. The format of the date must be respected
End Of Analysis	[26/Jan/2011:**00:30:00** +0100]	By default the last date found in the file but can be modified by hand.The format of the date must be respected
percentile	Number between 0 and 100	The percentile X% : the value which X % of the values are lesser than this value
PasValue	Number depending on scale value	When there is a huge numbers of values in the source file, a OutOfMemory could be raised during the computing of the Mediane Value and the Percentile value. This parameter permits to aggregate values by step, and so the number of values decreases
Analyse by column number	checked/unchecked	If checked, source file is in csv format.
Locale English (decimal separator => .)?	checked/unchecked	If Checked => English => decimal separator = . If not checked => French => decimal separator = ,
Number of Actors	2	Akka Actors to treat the file. =Nb CPUs (if not hyperthreaded) =NbCpus*2 (if hyperthreaded)
Csv separator	, or ; or other character	CSV Separator, enabled when Analyse by column number is checked
First regexp for Filter in pivot	Java/perl regex	First Regexp to extract the Pivot
Second regexp for Filter in pivot	Java/perl regex	If necessary, second regexp to extract the Pivot from the first matching
First regexp for Filter in Value	Java/perl regex	First regexp to extract the value
Second regexp for Filter in Value	Java/perl regex	If necessary, second regexp to extract the Value from the first matching
Scale of a value	1, 10, 0.001 ...	To change the scale of the value
Number of items of the topn	10	Defines the Top n that will be shown after the parsing.

In case of csv file(**Analyse by column number** checked) :
- the field **First regexp for Filter in pivot** is named **index of Column for Pivot (starts to 0)**
- the field **First regexp to extract a Value** is named **index of Column for Value (starts to 0)**

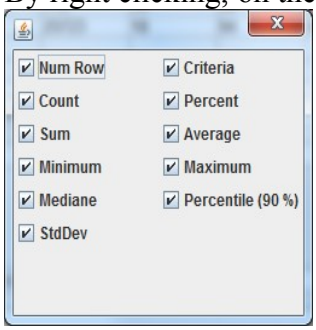
The other fields are unchanged.
After clicking on Analyse button :





Num Row	Criteria	Count	Percent	Sum	Average	Minimum	Maximum	Mediane	Percenti...	StdDev
0	POST /data/services/Manage	2516352	55.557	369751344	146.939	6	25723	22	181	639.56
1	GET /data/Filter1	1665240	36.766	248832	0.149	0	213	0	1	0.901
2	POST /data/services/manageCommercialCustomers	138384	3.055	4284192	30.959	17	6511	27	40	86.567
3	POST /data/services/ManageRole	99552	2.198	1675560	16.831	12	124	14	24	6.484
4	POST /data/services/manageIdentAppli	99504	2.197	801024	8.05	5	64	7	10	3.396
5	GET /data/services/Manage?wsdl	10296	0.227	388704	37.753	29	776	35	38	36.592
6	Total	4529328	100	377149656	83.268	0	25723	18	94	482.269

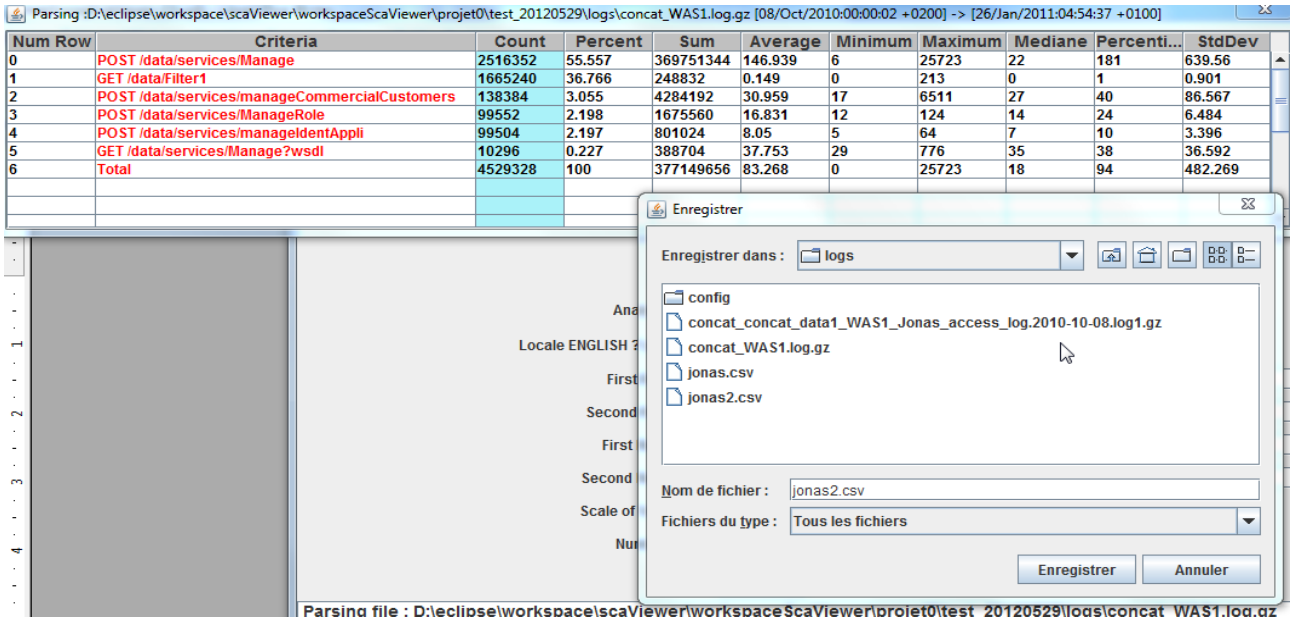
By right clicking, on the title of column (Sum for example), you can remove useless column :



By right clicking, in the contain of the table you can

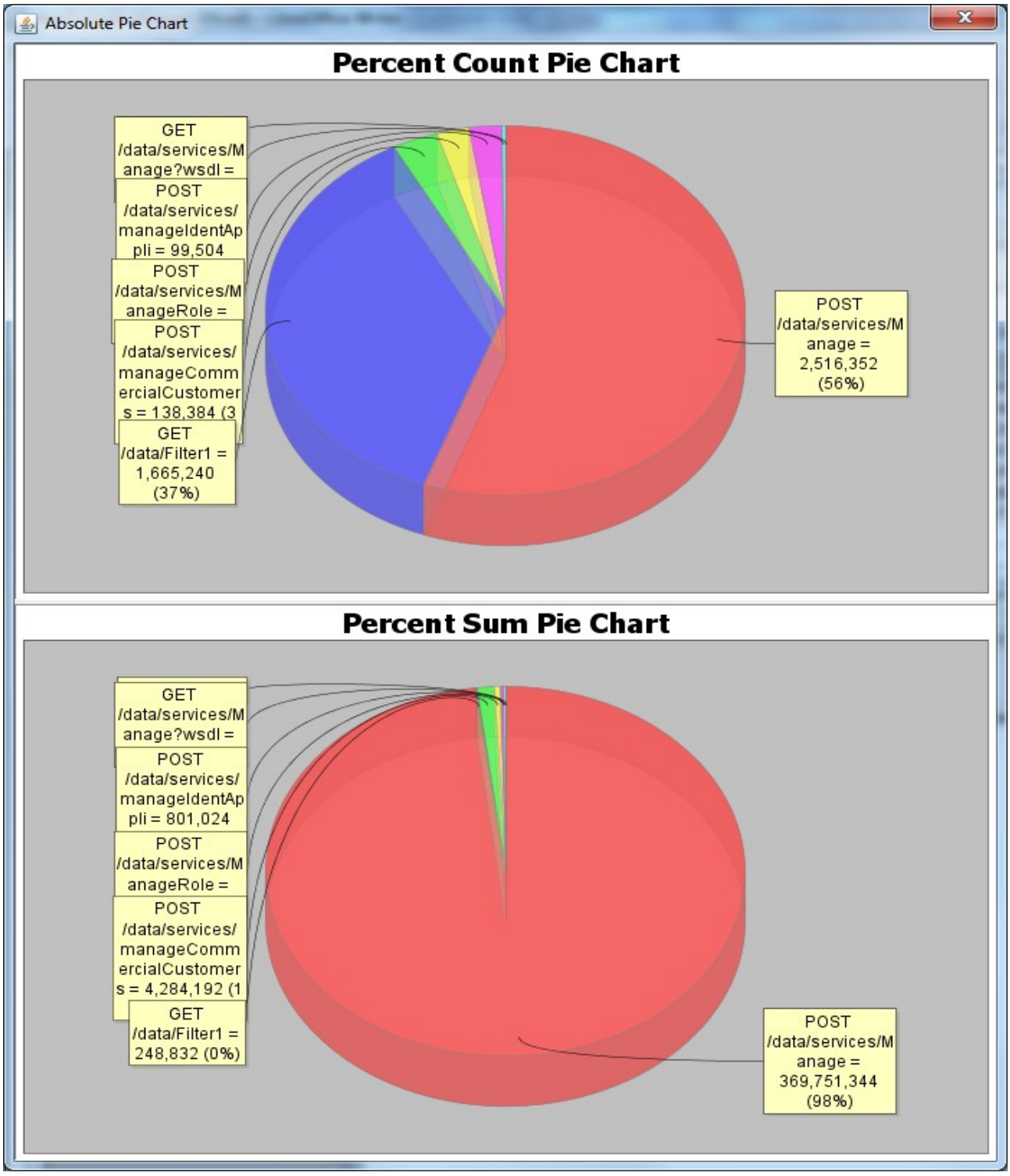
- save the table in a csv file

- graph the table as Pie Chart.



Hint: Saving this file in Csv Format, permits to fill automatically the Pivot Tab of the Menu **ScaLogParser (** see Menu **ScaLogParser** more above in the document**)**

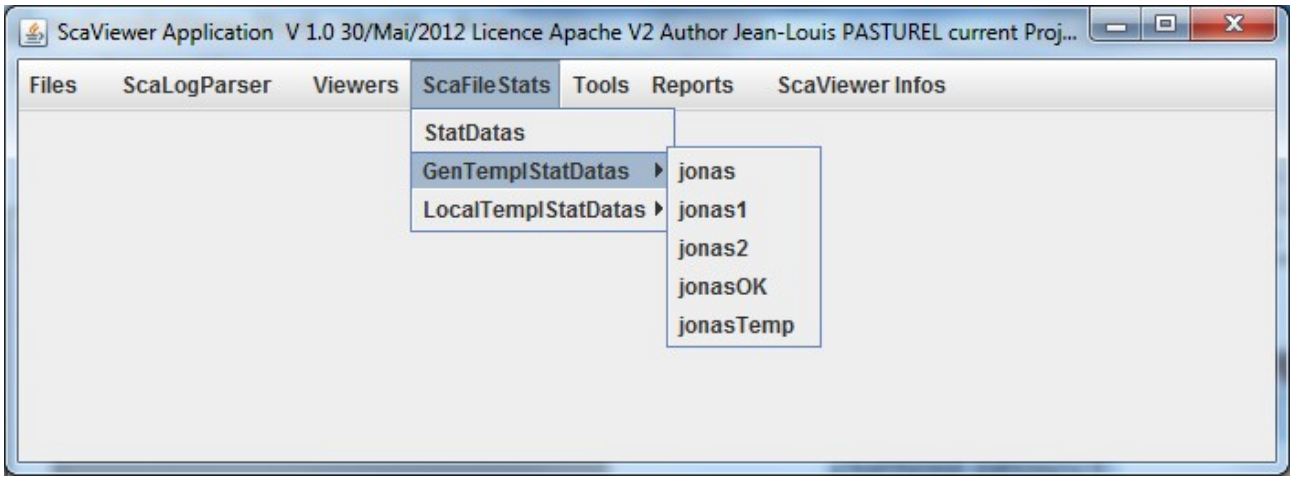
Example for the Pie Chart :



Button SaveAsTemplate

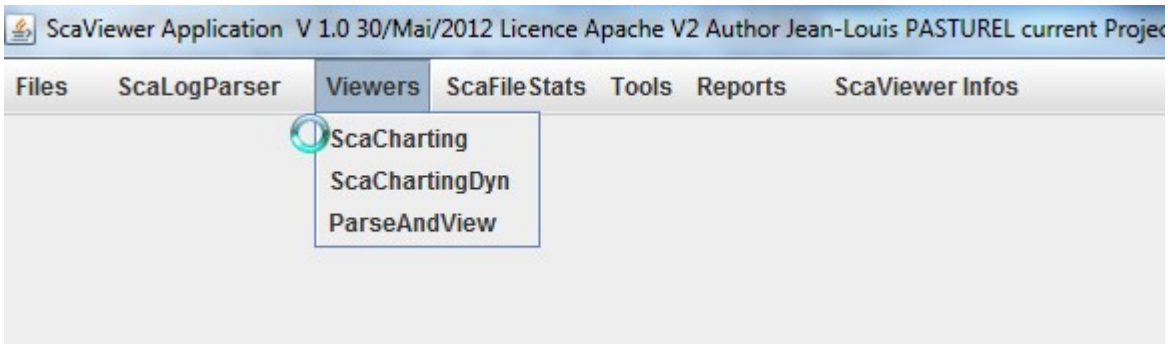
The button **SaveAsTemplate** runs as we have seen above to save a template with **ScaLogParser** You can save a General Template shared with all projects or a local template used only by the current project.

3.3.2 Sub-Menu “GenTemplStatDatas” and “ LocalTemplStatDatas ”



Choose a template before opening a log file.
The fields are filled with the values of the template **after** you have chosen the file to parse

3.4 Menu "Viewers"



Csv file restrictions :
The only CSV files that ScaCharting and ScaChartingDyn can graph must have the following configuration :

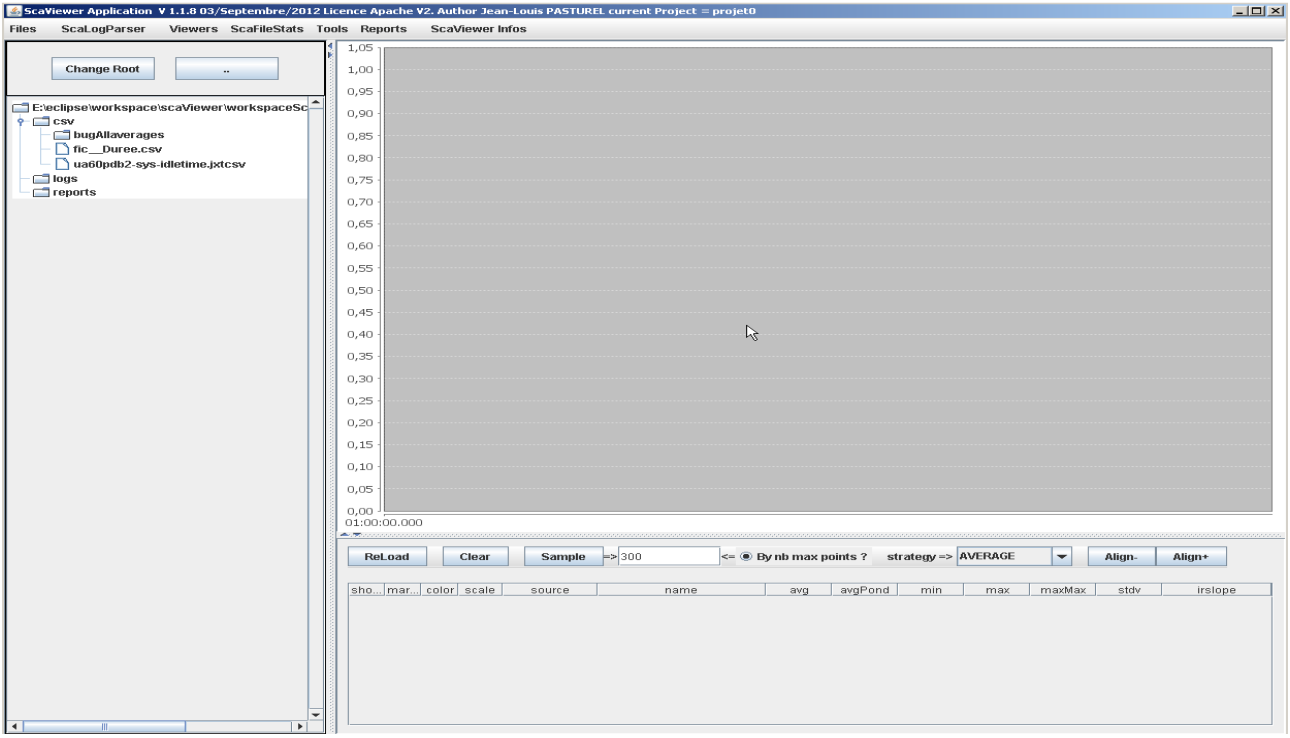
- the first line contains title of columns separated by the csv separator
- the first column must be a date respecting one format described in the file **<swingScaViewer_Home>/config/scaViewerDates.properties**
- The others columns can contains numeric values or strings (Pivots), but a column must be fill with the same type. This column can be empty (if ; is the csv separator ;; indicates an empty column)

This kind of file contains “**Time Series**”, in the framework JFreeChart concepts.

3.4.1 Sub-Menus "ScaCharting" and “ScaChartingDyn”

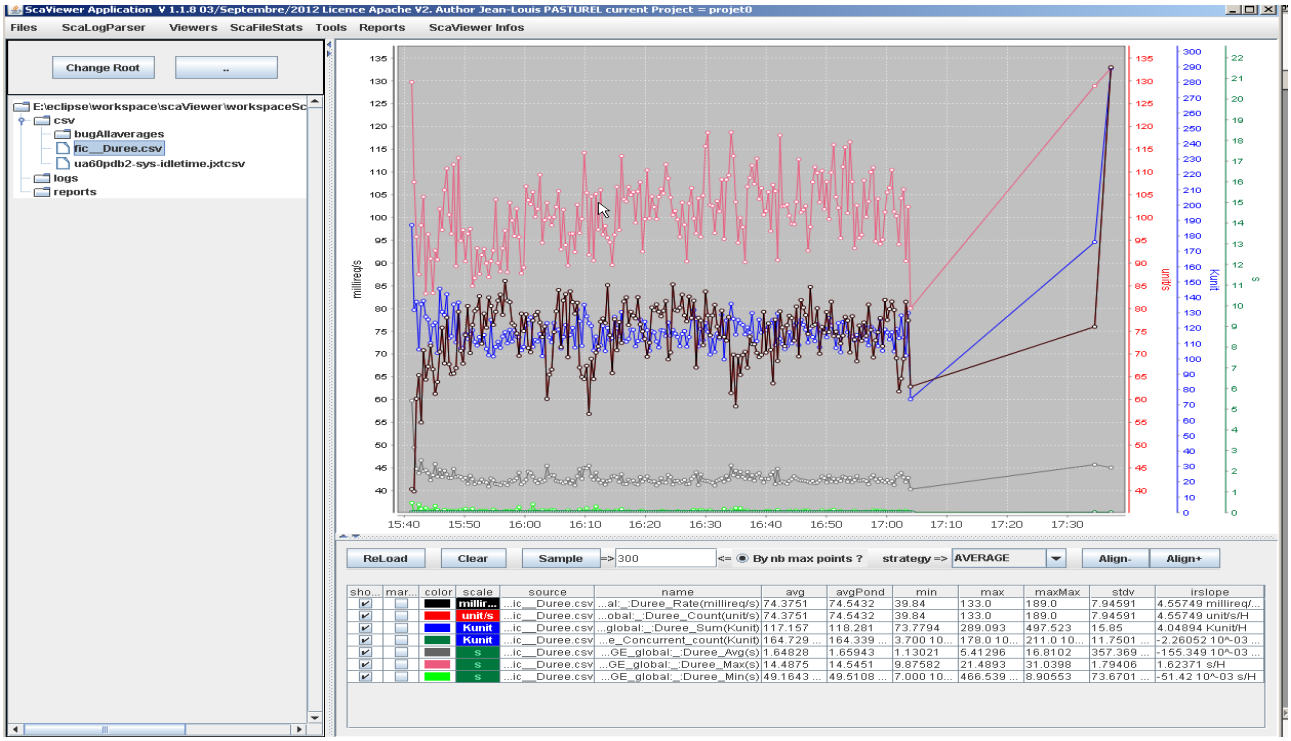
This 2 sub-menus are similar, the difference is that the ScaChartingDyn sub-menu refreshes periodically the graph if the csv file has changed. The period of refreshing is given by the variable : **`scaviewer.dyn.timeout`** in the file <swingScaViewer_Home>/config/**scaViewer.properties**

DnD means Drag and Drop.
This menu permits to choose csv files, the tree at the left is positioned at the root folder of the current project.
Only the files, where the suffix is in the list of the variable **`jtree.suffixes`** , are displayed (file <swingScaViewer_Home>/config/**scaViewer.properties)**



After deploying a branch of a tree by clicking on node, you can drag and drop a csv file or a discontinued list of csv files to the right side of the screen. You can also add csv file by DnD from the same or another directory.

The graph must be zoomed by selecting a region. It can be de-zoomed by dragging on it from the right to the left.



Clicking right on the graph provides also other feature (copy in the clipboard ..)
Clicking right on the title of the table permits to add/remove columns.
Clicking right in the contains of the table permits remove selected row series on the table and on the graph.

Buttons :

Reload

gives the possibility to reload the initial csv files

Clear

Cleanes all CSV file from the Chart and the table

Sample

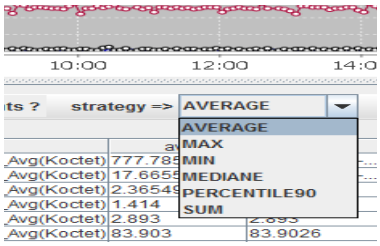
The button is bound with the text field and the radio button at its right

- if the radio button “By nb max points ?” is selected, the chart is shown with the number of points given by the text field (in fact the time interval shown is divided in 300 intervals for the example above)
- if the radio button “By nb max points ?” is unselected, the time interval shown is divided in periods of the given by the text field expressed in milliseconds.

Hint :

When you zoom a part of the chart, click on **Sample** to get more points.

Strategy



When the csv files have more lines than can be displayed on chart (or the number in the text field described just above), you can choose the strategy of aggregation in this combo-box.

Align- : (comes with V 1.1.7**)**

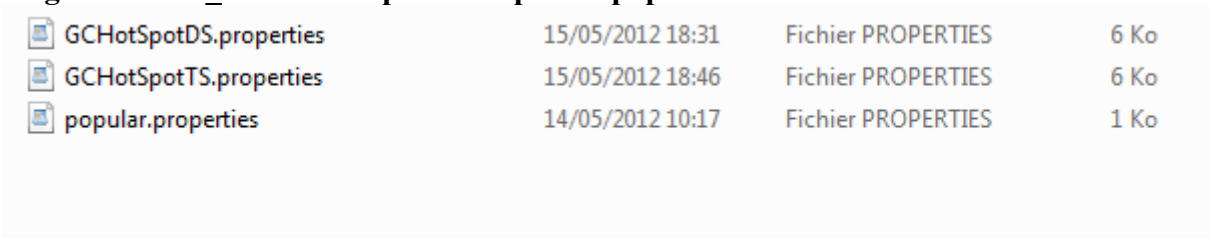
Permits, when 2 or more series are “marked” in the table (column “marked”), to align these series to the left side of the chart by shifting the beginning of all series to the minimum of the beginning of the marked series. It helps for comparison or correlation between series. It can be combined with the “Translate” feature of the table (§ 3.4.3).

Align+ : (comes with V 1.1.8**)**

Permits, when 2 or more series are “marked” in the table (column “marked”), to align these series to the right side of the chart by shifting the beginning of all series to the maximum of the beginning of the marked series. It helps for comparison or correlation between series. It can be combined with the “Translate” feature of the table (§ 3.4.3).

3.4.2 Sub-Menu "ParseAndView"

This sub-menu is for parsing popular logs files that have not to be configured by hands.
The principle is to recognise the type of file (JVM GC Logs for examples) and to apply to it a template of scaLogParser.
This templates are located in the directory :
<swingScaViewer_Home>/templates/scaparser/popular



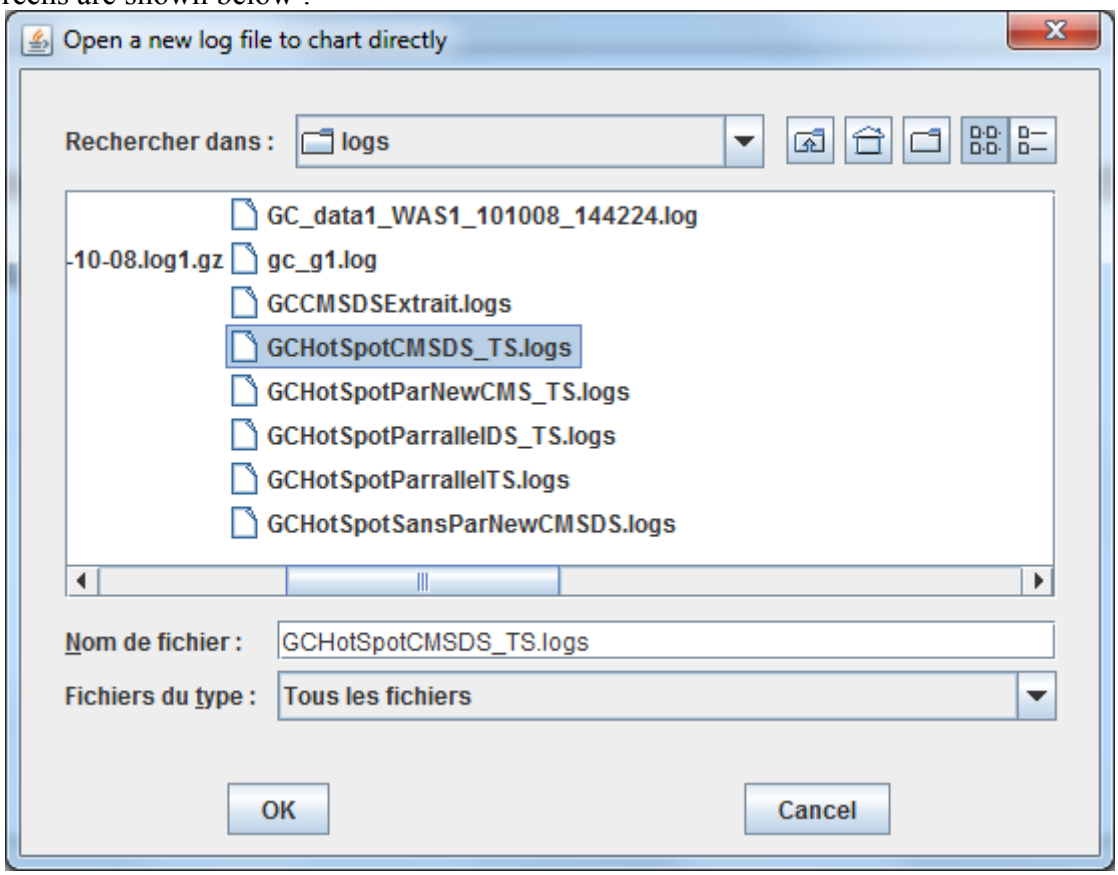
The files that you meet here are (but it will be extended in the future) :
The files **GCHotSpotDS.properties** and **GCHotSpotTS.properties** are the templates build from the ScaLogParser menu described more above in this document.

The file **popular.properties** file helps to recognize the “footprint” of the log file. If the file log has no footprint in this file, a popup message is displayed.
popular.properties :

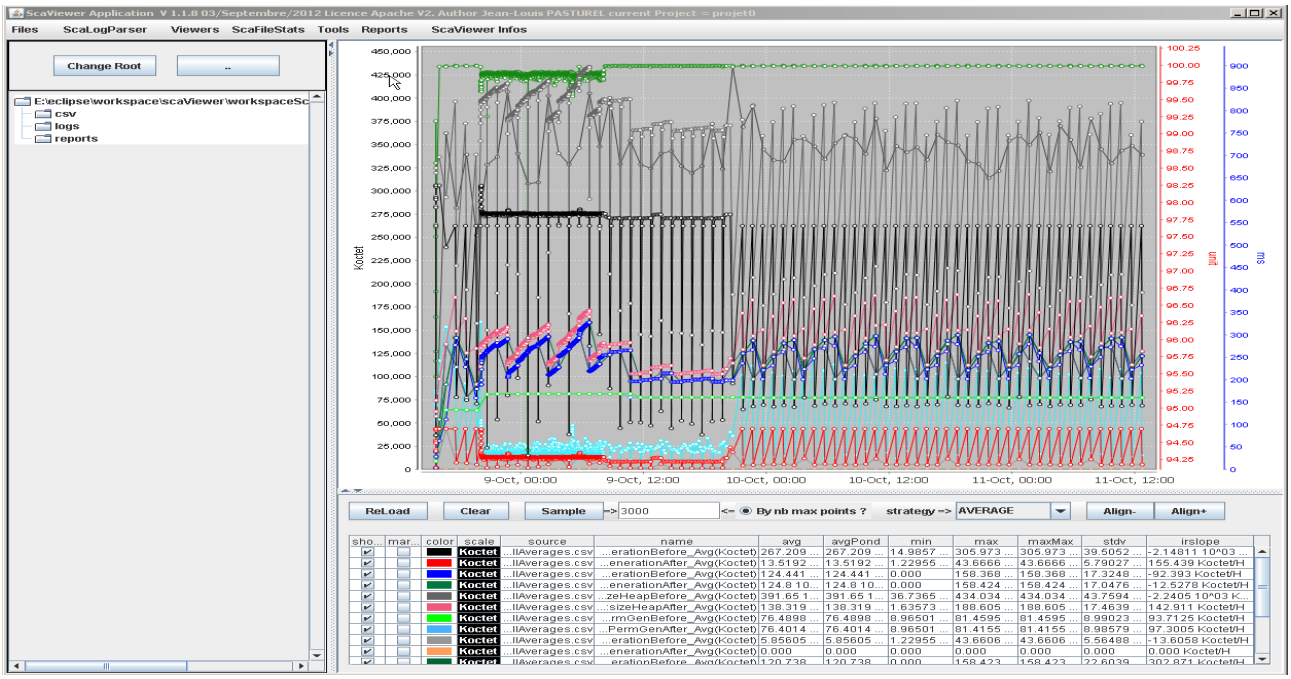
```
popular.list=GCHotSpotDS;GCHotSpotTS


## Empreinte GCHotSpotDS
popular.GCHotSpotDS.debEnr=^(\\d{4}-\\d\\d-\\d\\dT\\d\\d\:\\d\\d\:
\\d\\d\\.\\d\\d\\d(\\+|-)\\d{4}\:\\s+\\d+\.\\d+\:\\s+\\[)
popular.GCHotSpotDS.finEnr=
popular.GCHotSpotDS.isDateExplicit=true
popular.GCHotSpotDS.reg1=(\\[GC|\\[CMS|\\[Full GC)



# Empreinte GCHotSpotTS
popular.GCHotSpotTS.debEnr=^(\\d+\\.\\d+\:\\s+\\[)
popular.GCHotSpotTS.finEnr=
popular.GCHotSpotTS.isDateExplicit=false
popular.GCHotSpotTS.reg1=(\\[GC|\\[CMS|\\[Full GC)
```

popular.list contains the list of templates treated (without the **.properties** suffix)

The screens are shown below :



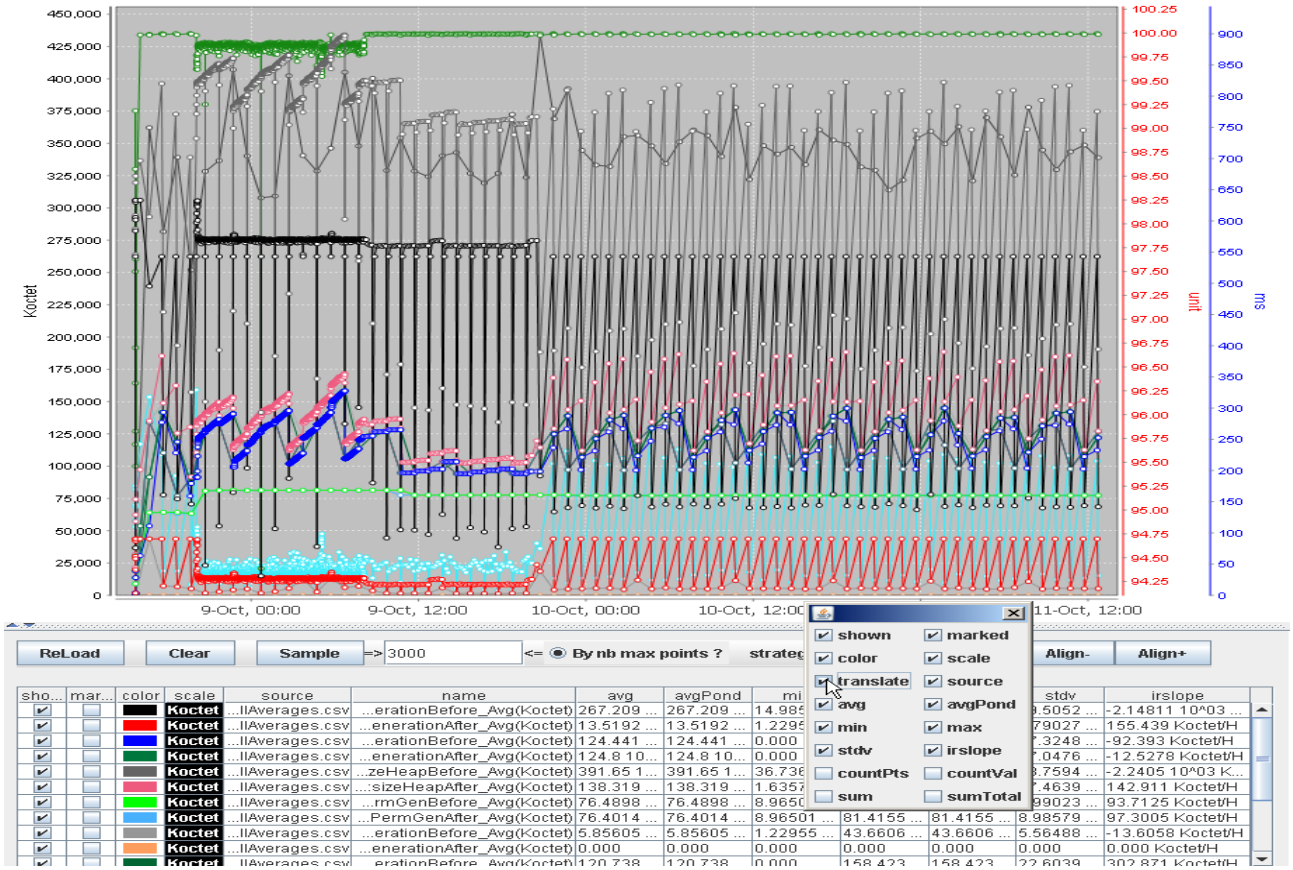
After choosing and click on OK, the chart is directly displayed :

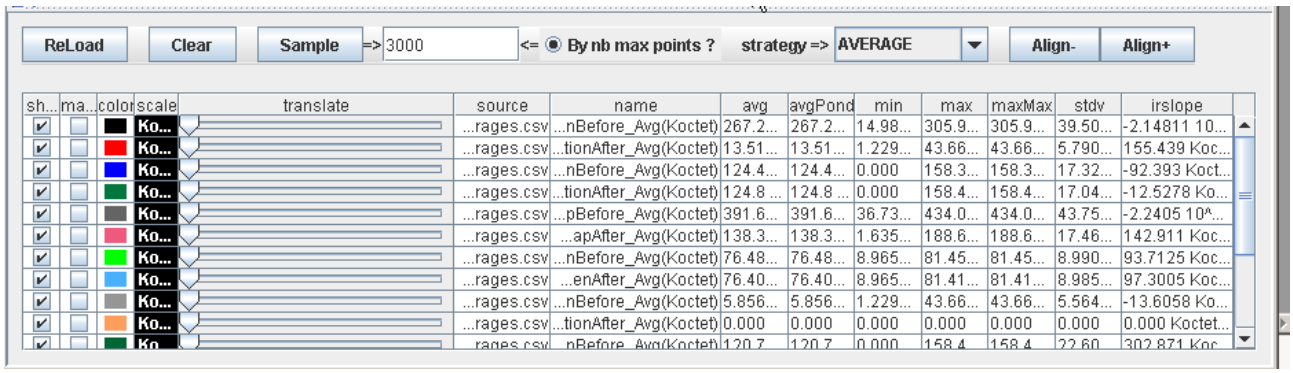


You can go on manipulating the chart as described in the precedent paragraph.

3.4.3 “Funny” feature

When you right click on the title of the table at the button of the Chart window, you can add a column “Translate” :



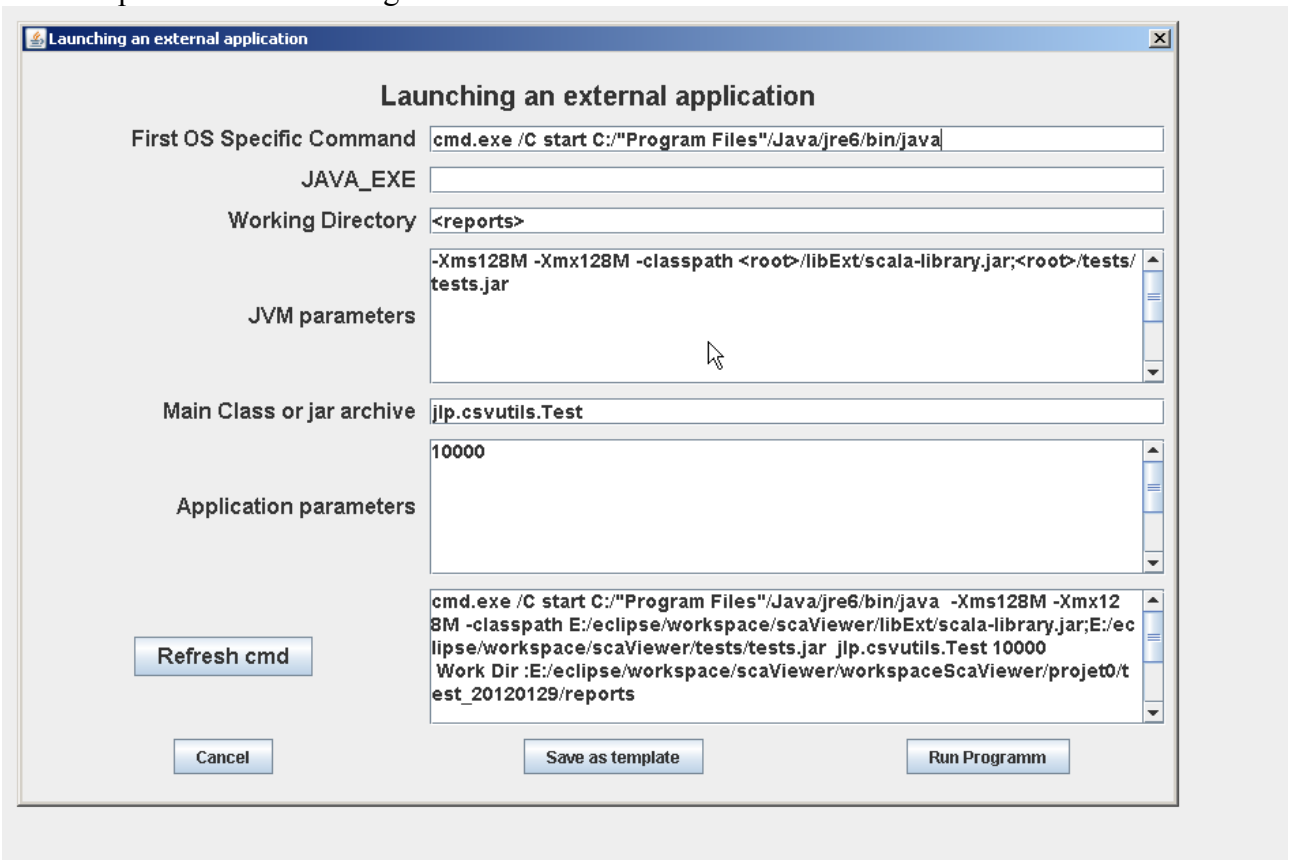


By playing in the slider you can translate Time Series to the right. It would be useful to compare the behaviour of two time series where the dates are different (every hours, every day, every week ...).

3.5 Menu "MyCommands"

This menu allowes to launch external commands, by using results or files located in the swingScaViewer application.
You can create template for interesping external commands. The principle is the same as described for the other precedent menus (ScaFileStats, ScaLogParser)

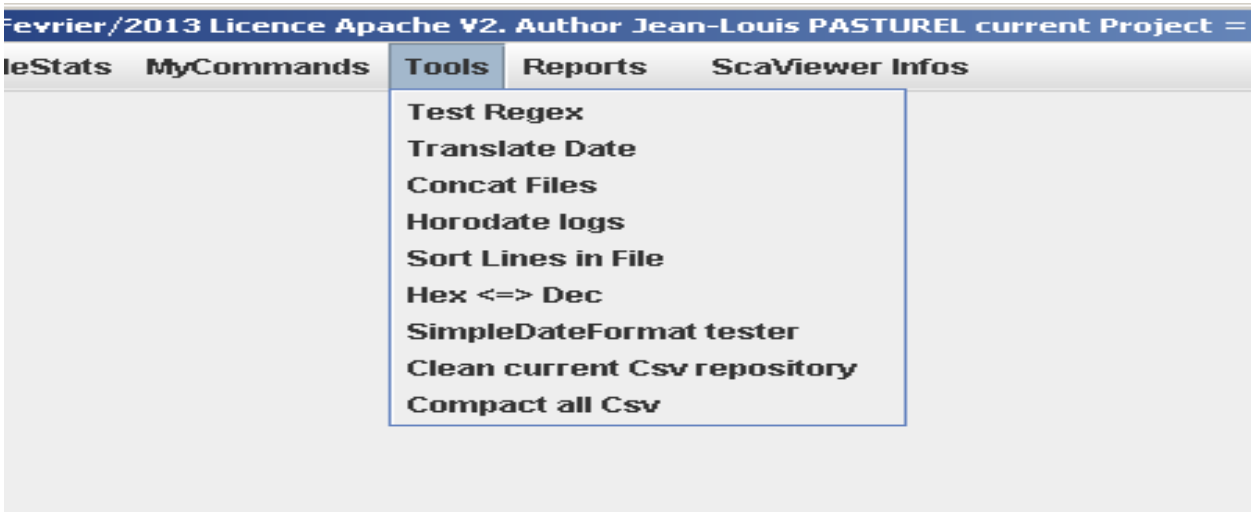
An example of the main dialog is shown below :



You can launch java application or native command.
I don't detail the fields, they are explicit.
For each field, you can use patterns ;
- <workspace> the directory of the workspace of swingScaViewer
- <libCommands> the directory jars of the libCommand external commands
- <root> the root directory of the installation of swingScaViewer
- <currentProject> the name of the current project
- <pathYoungTirDir> the last directory created in the current project folder
- <reports> -> the report directory of the last directory created in the current project folder

3.6 Menu "Tools"

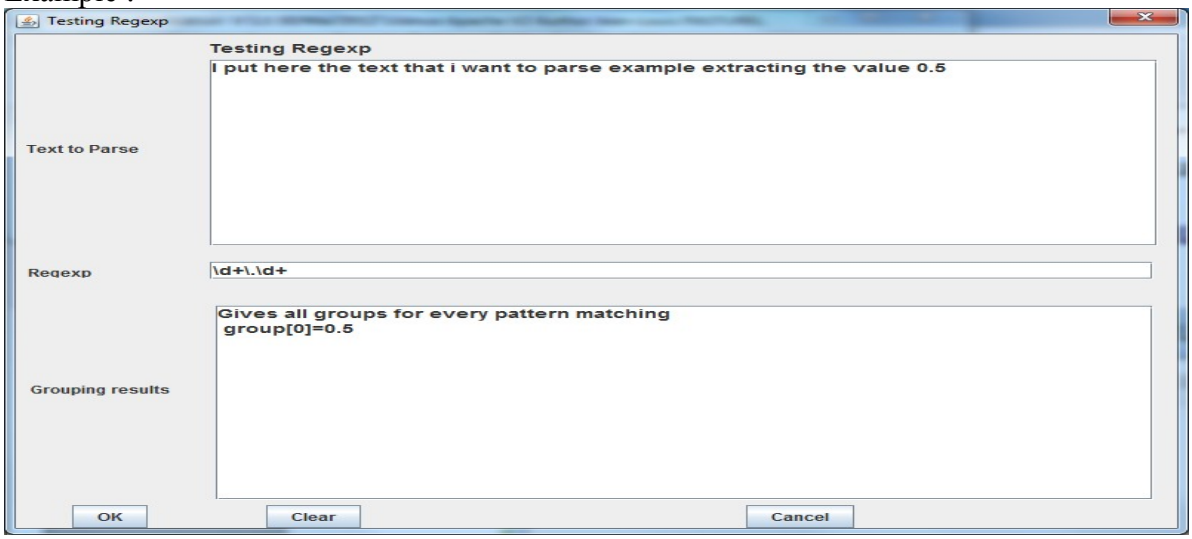
This menu groups tools that are useful for swingScaViewer. Other can be added in future versions.



3.6.1 Sub-Menu "TestRegexp"

As seen below, the log parsing uses a lot the mechanism of Regexp Pattern Matching.
The javadoc link is :
http://java.sun.com/j2se/1.6.0/docs/api/java/util/regex/Pattern.html

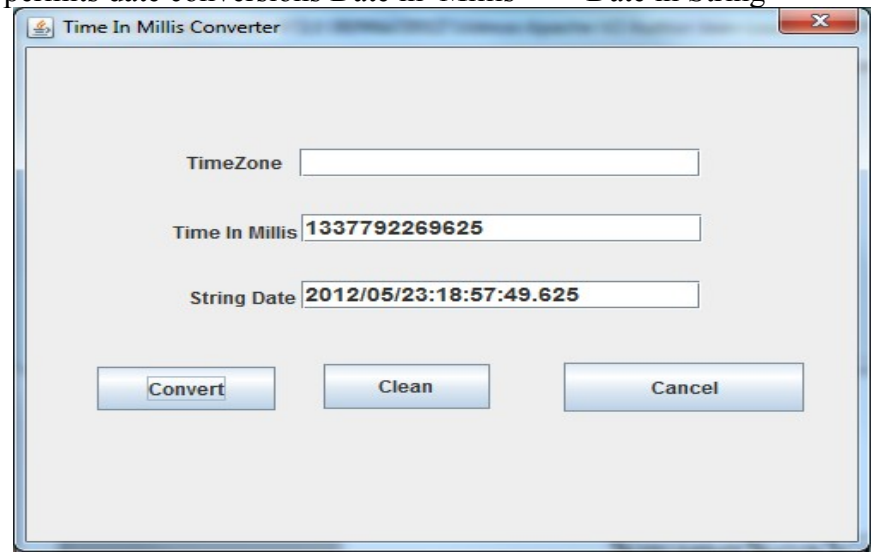
Example :



For **swingScaViewer**, the interesting result is the result contained in **group[0]**.

3.6.2 Sub-Menu "TimeInMillis"

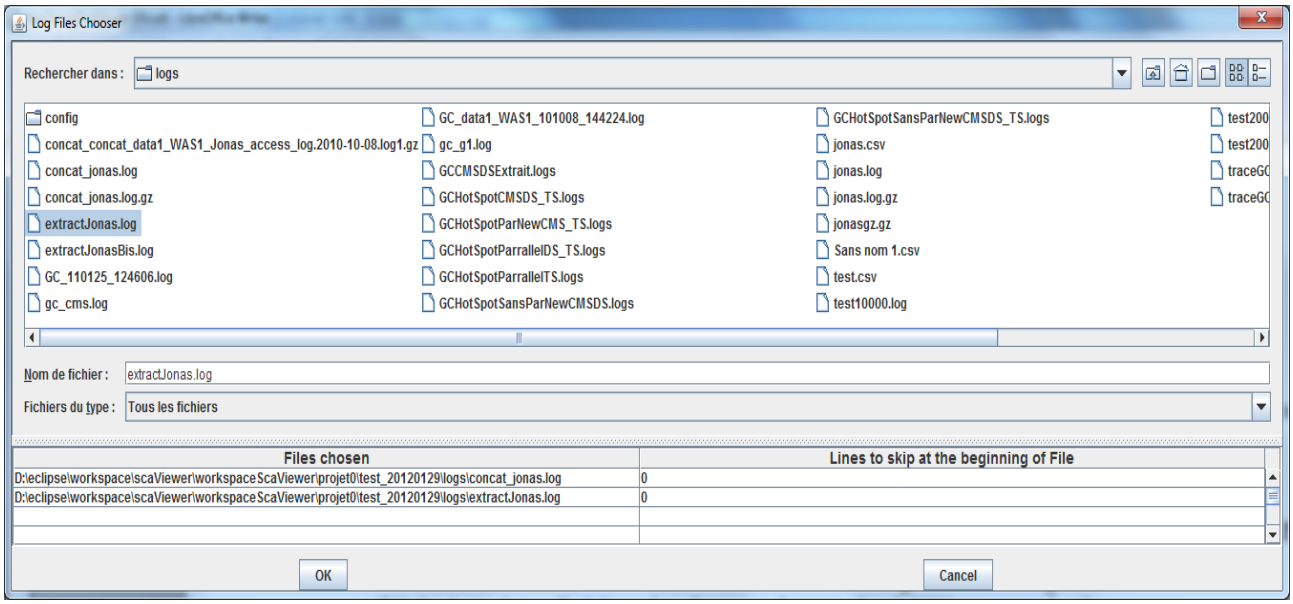
DialogBox that permits date conversions Date in Millis <=> Date in String

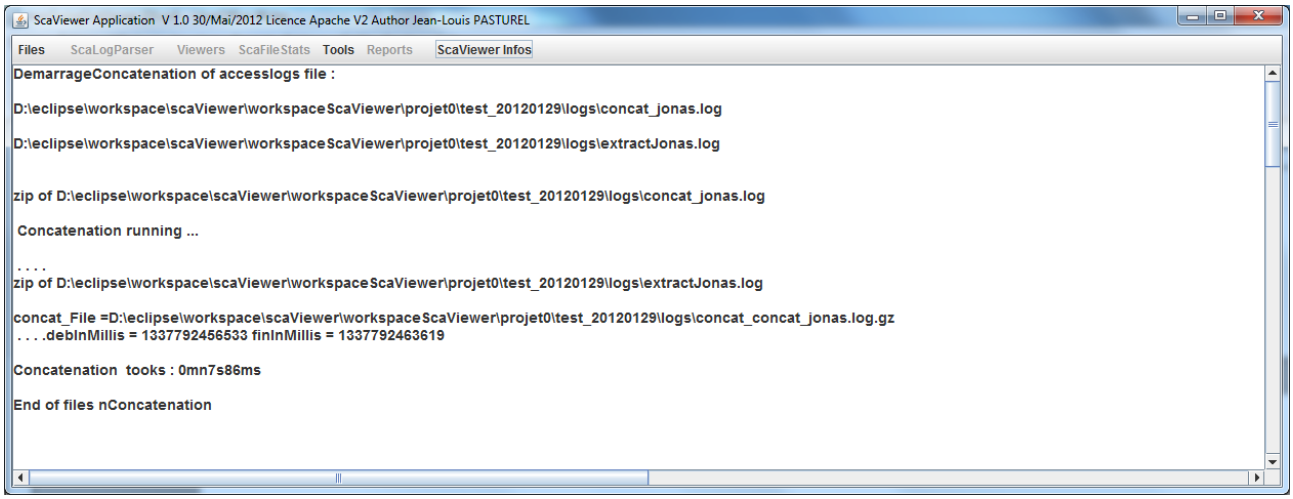


We can :
- not fill any field, and the current date is returned both in milliseconds time and string time
- fill milliseconds Time field and afterwards get the String date
- fill String date field and afterwards get the Milliseconds date
- in the three cases we can modify the Timezone with the format (+|-)dddd => ex : +0200

3.6.3 Sub-Menu "Concat Files"

This sub-menu permits to concatenate files with the same structure in one file. The result file is in gzip format.
The concat files menu, concatenates files, some following the others.





3.6.4 Sub-Menu "Horodate Logs"

It is a tool, that can simplify the parsing of logs files with the structure like below :

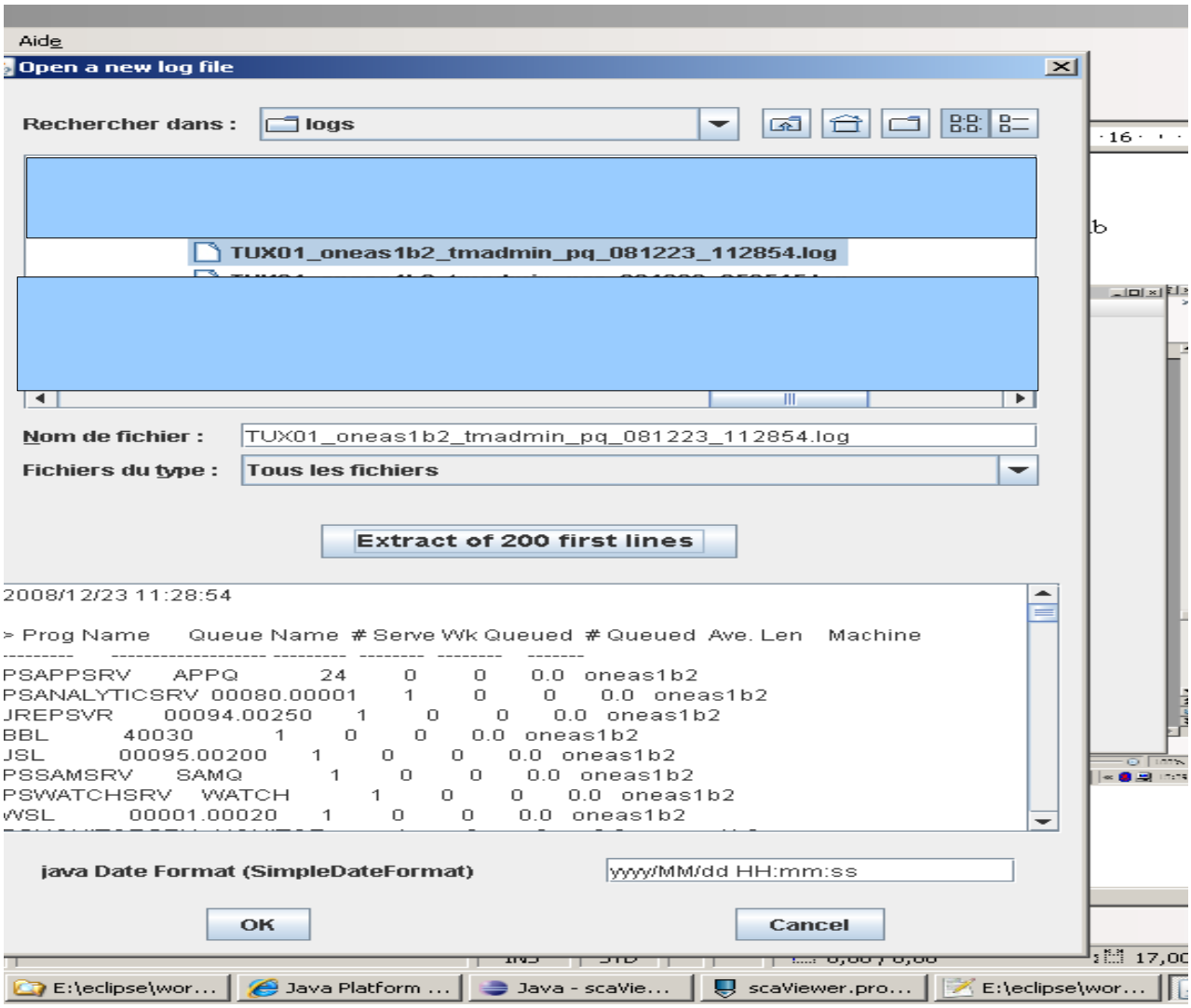
```
2012/08/15 10:44:12
useless line1 with a specific pattern
useless line 2  with a specific pattern
pivot1 value1 value2
pivot2 value1 value2
…
2012/08/15 10:44:14
useless line1 with a specific pattern
useless line 2  with a specific pattern
pivot1 value1 value2
```

```
pivot2 value1 value2
…
2012/08/15 10:44:16
useless line1 with a specific pattern
useless line 2  with a specific pattern
pivot1 value1 value2
pivot2 value1 value2
```

The tool produces a file like this :

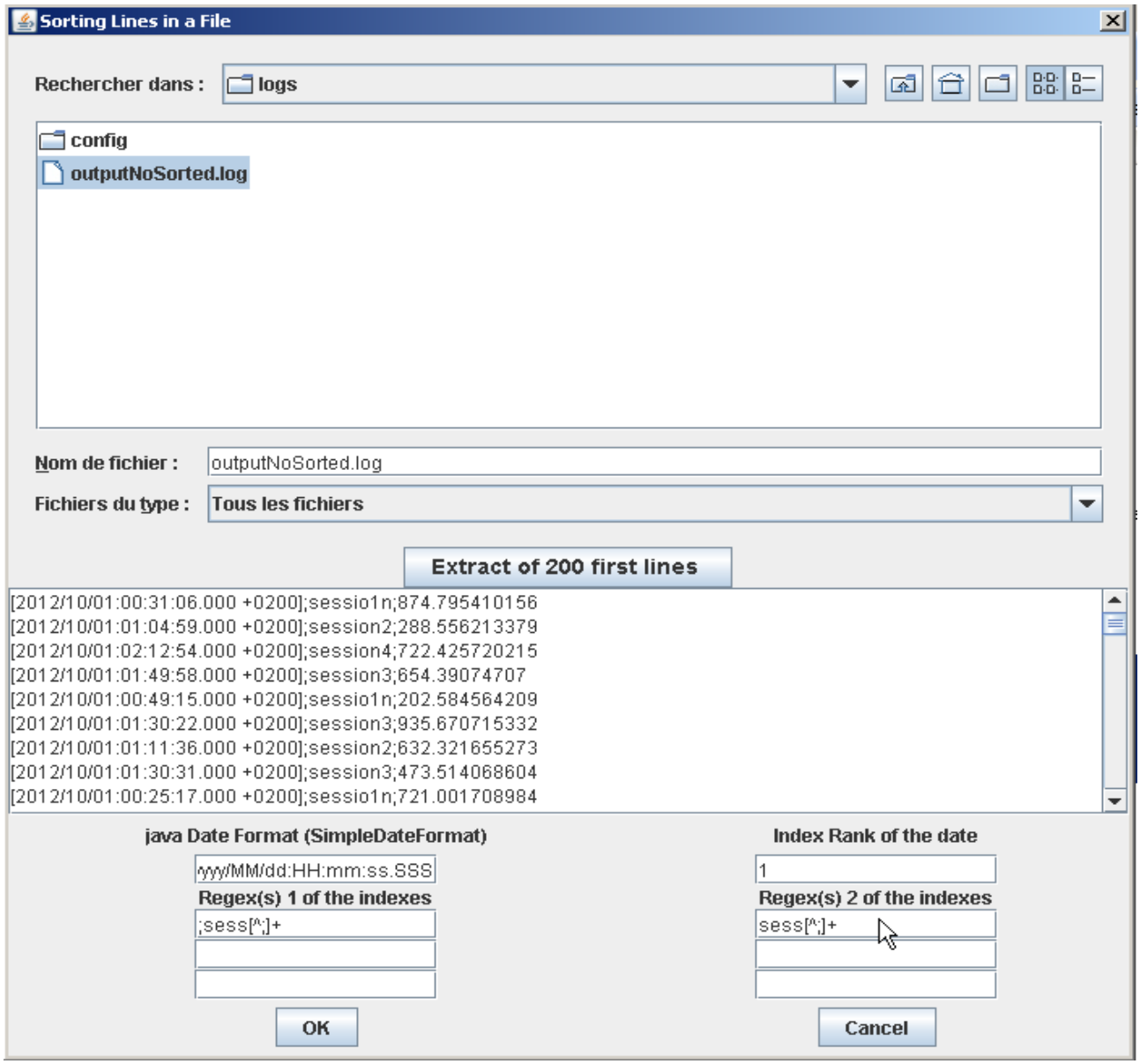
```
2012/08/15 10:44:12 useless line1 with a specific pattern
2012/08/15 10:44:12 useless line 2  with a specific pattern
2012/08/15 10:44:12 pivot1 value1 value2
2012/08/15 10:44:12 pivot2 value1 value2
2012/08/15 10:44:12 …
2012/08/15 10:44:14 useless line1 with a specific pattern
2012/08/15 10:44:14 useless line 2  with a specific pattern
2012/08/15 10:44:14 pivot1 value1 value2
2012/08/15 10:44:14 pivot2 value1 value2
2012/08/15 10:44:14 …
2012/08/15 10:44:16 useless line1 with a specific pattern
2012/08/15 10:44:16 useless line 2  with a specific pattern
2012/08/15 10:44:16 pivot1 value1 value2
2012/08/15 10:44:16 pivot2 value1 value2
```

This kind of file is easy to parse with scaLogParser. Useless lines can be excluded in the tab “Enregistrements”
The dialog is shown below.
You have to verify that the java Date Format is correct in regards of the extract of the file. If it isn't, you have to correct it. The java date format must be described in the file $root/config/scaViewerDates.properties



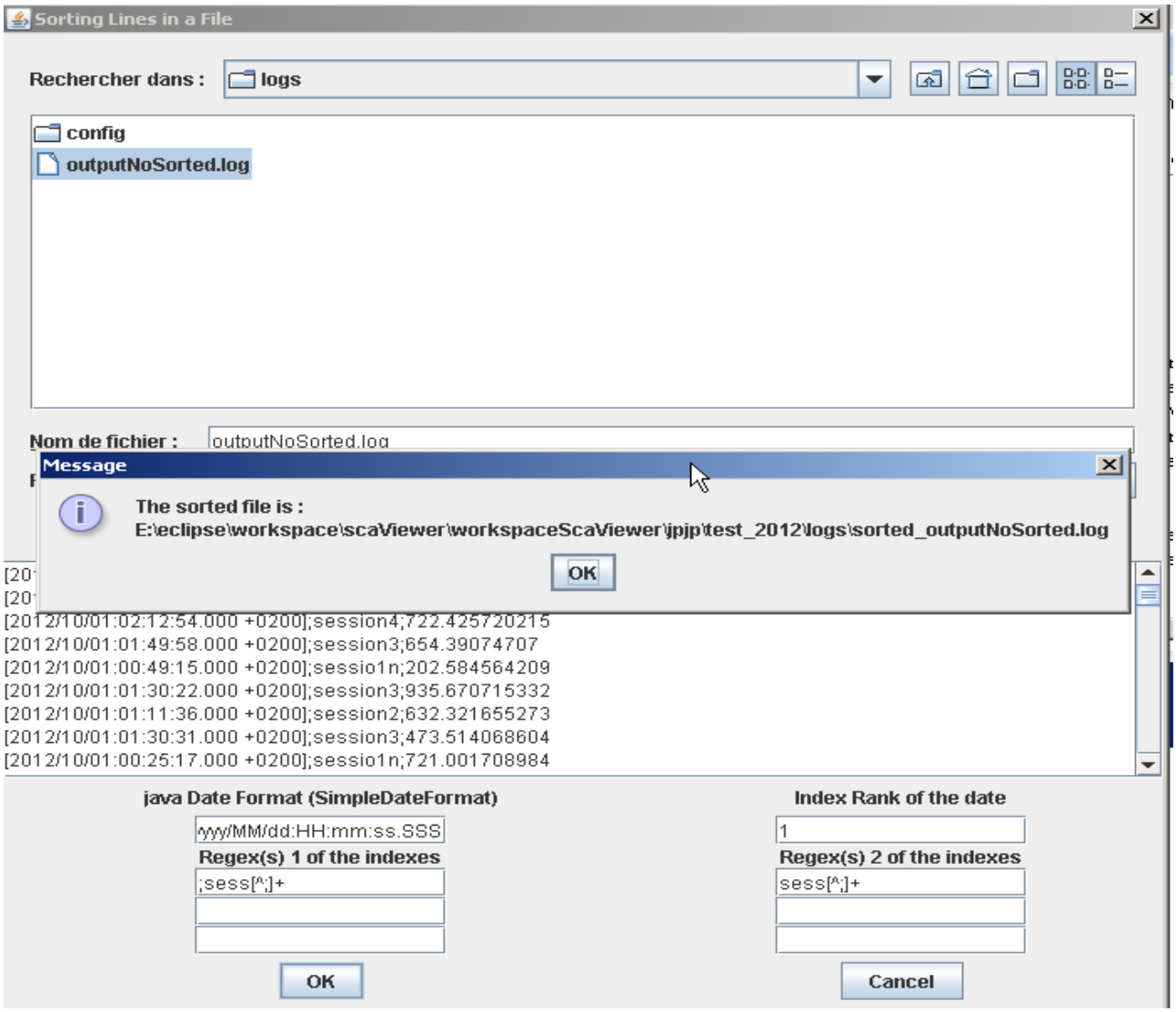
3.6.5 Sub-Menu "Sort Lines In File"

This tool permits, to group lines in a file, that share constant values in this lines. This kind of log files describes different states of a transaction for examples, and the lines are mixed in the file. The mechanist to sort the lines is based on regex, as seen above, with a two pass phases. If the lines contain a date, it can added everywhere in the concatenated index to help to sort the line.



The file contains a pivot sessionId, we want to sort with this pivot, and also with the date at the end; the first index (index 0) is the first production of the first couple of regex, the second index is the date (rank set to 1). If the rank of the date is < 0, the date doesn't participate to the composed index.

After clicking on OK, the sort runs and the result is done on the popup as shown below :



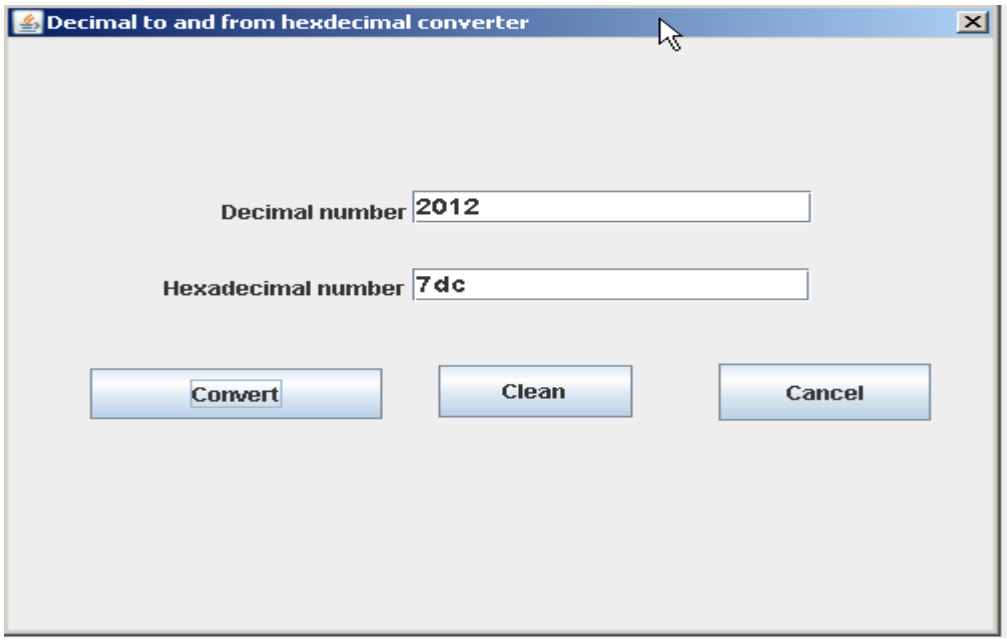
an extract of the new sorted file :

```
1  date;SessionID;valeur
2  [2012/10/01:00:00:00.000 +0200];session1n;896.453674316
3  [2012/10/01:00:00:01.000 +0200];session1n;565.190917969
4  [2012/10/01:00:00:02.000 +0200];session1n;364.403259277
5  [2012/10/01:00:00:03.000 +0200];session1n;527.036804199
6  [2012/10/01:00:00:04.000 +0200];session1n;513.226074219
7  [2012/10/01:00:00:05.000 +0200];session1n;421.101989746
8  [2012/10/01:00:00:06.000 +0200];session1n;170.625595093
9  [2012/10/01:00:00:07.000 +0200];session1n;960.712280273
10 [2012/10/01:00:00:08.000 +0200];session1n;209.536499023
11 [2012/10/01:00:00:09.000 +0200];session1n;383.40625000
12 [2012/10/01:00:00:10.000 +0200];session1n;453.962280273
13 [2012/10/01:00:00:11.000 +0200];session1n;395.857696533
14 [2012/10/01:00:00:12.000 +0200];session1n;365.098632812
15 [2012/10/01:00:00:13.000 +0200];session1n;926.809204102
16 [2012/10/01:00:00:14.000 +0200];session1n;397.192901611
17 [2012/10/01:00:00:15.000 +0200];session1n;623.795166016
18 [2012/10/01:00:00:16.000 +0200];session1n;625.507507324
19 [2012/10/01:00:00:17.000 +0200];session1n;153.096374512
20 [2012/10/01:00:00:18.000 +0200];session1n;408.479034424
21 [2012/10/01:00:00:19.000 +0200];session1n;628.981140137
```

This tool can be useful, when there is a sessionID, with a line for every state of the transaction (Start, running, end). Whith 3 index and the date, we are able to sort the lines, and so the file is parsable with scaLogParser in mutiLine records.

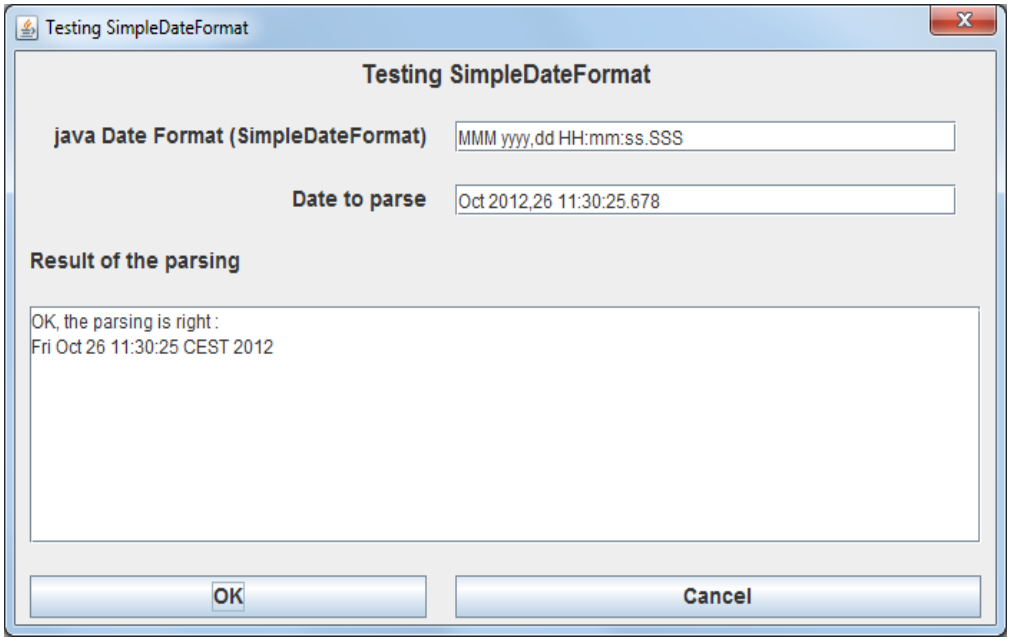
3.6.6 Sub-Menu "Hex <=> Dec"

This tool permit, to convert an hexadecimal number to a decimal number, and the opposite.



3.6.7 Sub-Menu "SimpleDateFormat tester"

This tool permits, to test a SimpleDateFormat against a test date.



3.6.8 Sub-Menu "Clean current csv directory"

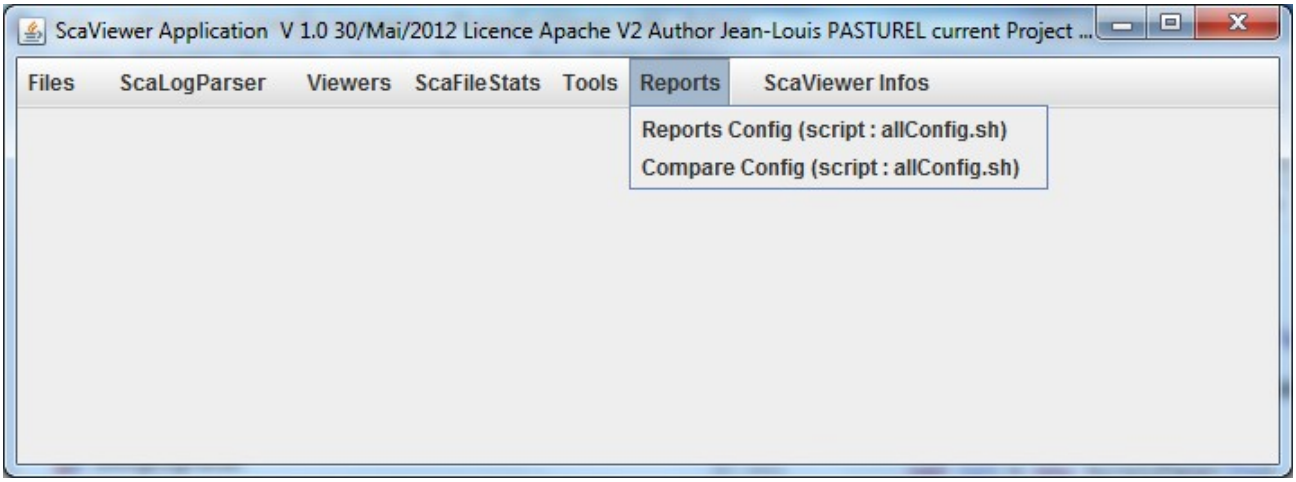
Remove all duplicated directories all project csv directory of the youngest scenario (current project/scenario).
The kept directory is the youngest.
A duplicate directory :

- a directory name is compound by <prefix><suffix>
- a suffix is a formatted date as (-|_)?\d{8}(-|_)\d{6} corresponding to java SimpleDateFormat : -yyyyMMdd-HHmmss ou _yyyMMdd_HHmmss
- a directory is duplicated if the prefix are the same

3.6.9 Sub-Menu "Compact all Csv"

Remove all duplicated directories (see below for the definition of duplicate directory) in the csv directories for all projects.
The kept directories are the youngest.
Warning : it can take a long time.

3.7 Menu "Reports" (Experimental and very specific)



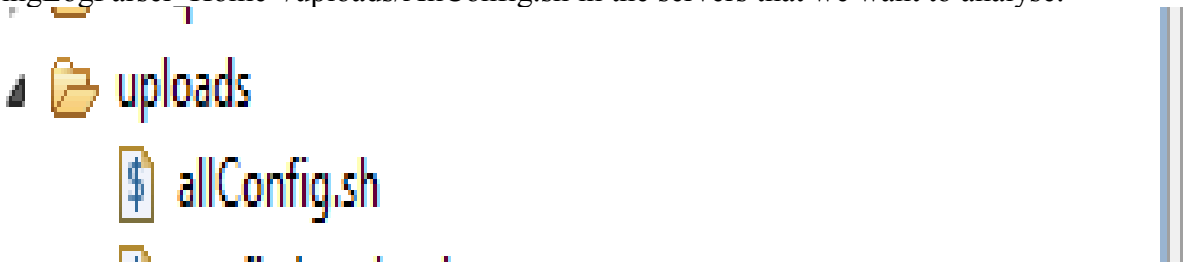
The goal of this Menu is to generate reports in HTML format. There are 2 Sub-Menus :

- **Report Config (script : allConfig.sh)**
- **Compare Config (script : allConfig.sh)**

This sub-menus are very contextual and strongly bound with the script
<swingScaViewer_Home>/uploads/allConfig.sh
They analyse Linux configuration (the basics), Apache configuration, JOnAS 4.x and JonAS 5.x configurations, Weblogic10 configuration. These menus will not evolute in the future.

3.7.1 Sub-Menu Report Config

This menu permits an analyse of principal configuration files (System, Apache, PHP, mod-jk, Jonas4/5, Weblogic). The generated report gives also some tips for tuning.
Before running this Menu, we have to run the script
<SwingLogParser_Home>/uploads/AllConfig.sh in the servers that we want to analyse.



There is a semi-automatic manner to do that when servers are directly accessible in SSH protocol (Menu : **SSH Cnx uploads/downloads**).

The manual procedure is described below :

- upload file AllConfig.sh under /tmp of all servers that we want to analyse
- connect with user root if possible.
- cd /tmp

- chmod 777 AllConfig.sh
- Verify that all the applications are running
- launch the script by : ./ AllConfig.sh
- under /tmp the script produce .xml files and zip files
- Download these files (xml and zip files) under the folder of **workspace** of swingScaViewer : **<Workspace>/<project>/<Tir_Current>/reports**

After you can generate the report by clicking on **Reports Configs (AllConfig.ksh)**
A html file **reportConfig.html** is created in the folder
<Workspace>/<project>/<Tir_Current>/reports/reportConfig.html

The file is displayed in the screen, it is generated in the folder
$workspace/**<project>/<Tir_Current>/reports/**
By right-clicking of the EditorPane, you copy it in the clipboard, and you can paste it in a document.
The file can also be read by FireFox or IE, and parts of it can be copied and pasted in Words document.

3.7.2 Sub-Menu Compare Config

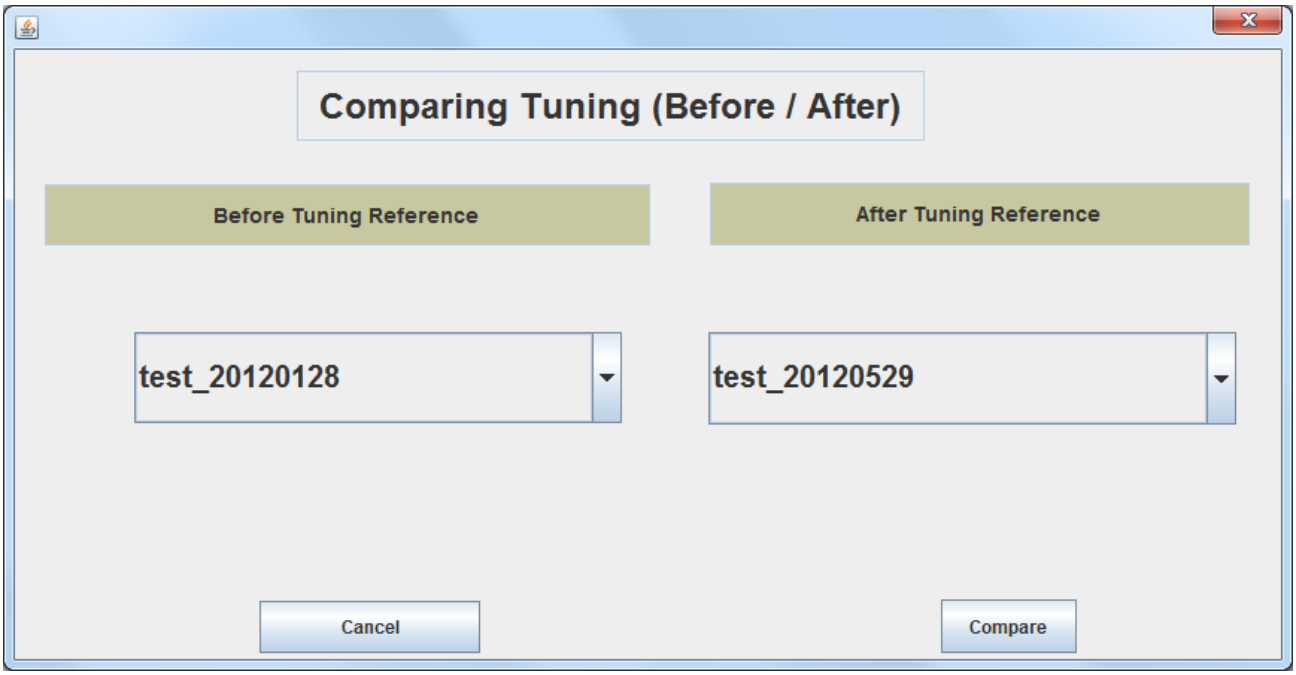
At the end of a campaign of tests, this Menu permits to compare configuration before Tuning and configuration after tuning, in the reference tests.

The html report gives the differences between BeforeTuning configuration files and the final reference Tuning.

The report may be generated, if at least you have done the operation described in the menu **Report Config** once.

To make this report, you have to choose 2 references test in the campaign.

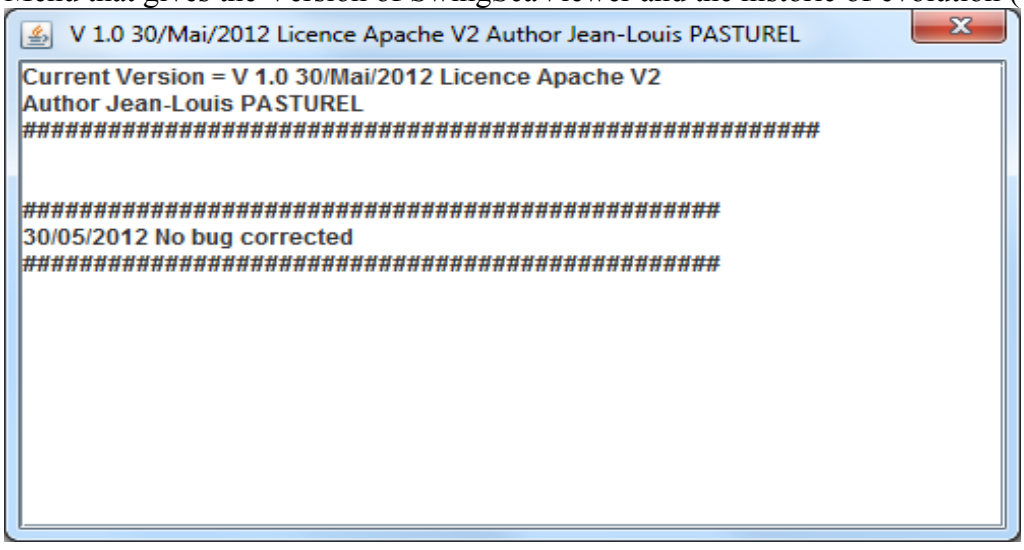
The next Dialog permits to choose the reference tests:



Clicking on Compare button generates the report (It can take a long time):
The file is displayed in the screen, it is generated in the folder $workspace/project/compareConf.html.
By right-clicking of the EditorPane, you copy it in the clipboard, and you can paste it in a document.
The file can also be read by FireFox or IE, and parts of it can be copied and pasted in Words document.

3.8 Menu "ScaViewer Infos"

Menu that gives the Version of SwingScaViewer and the historic of evolution (bugs and features)



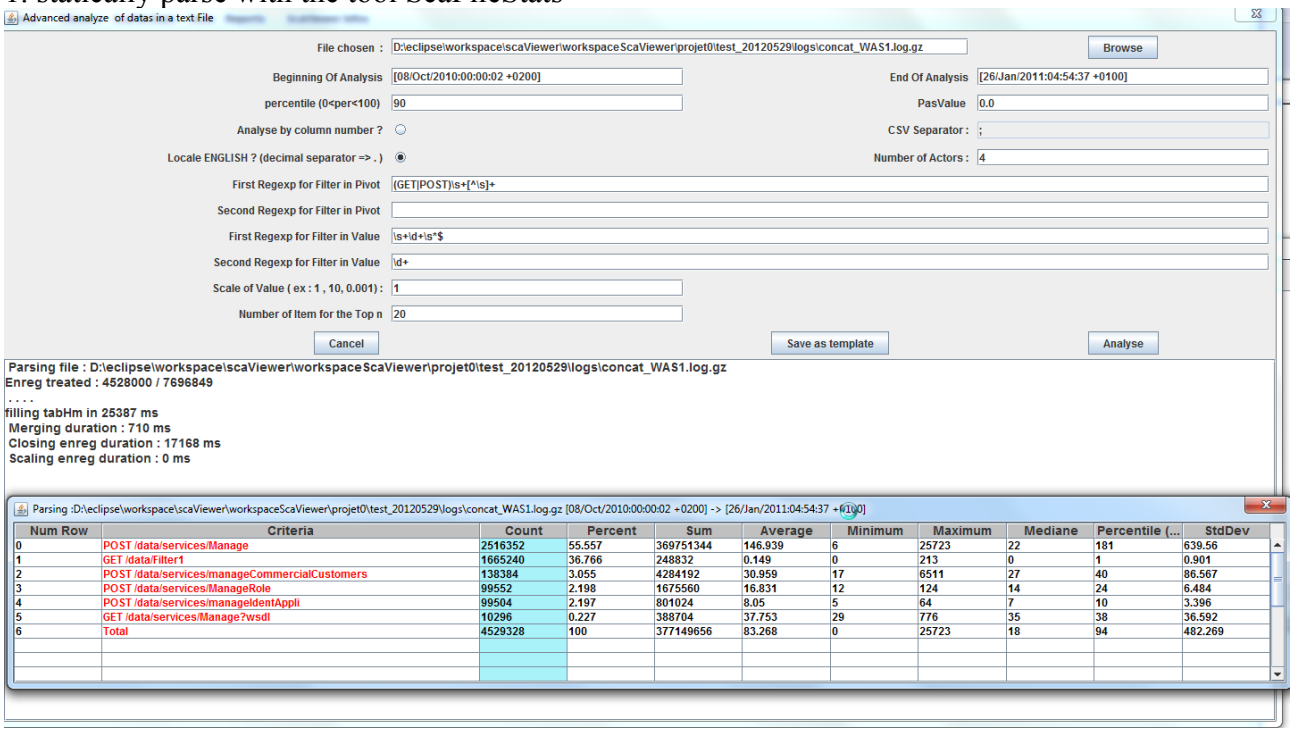
4 General Procedure

The procedure describes how to treat a dated log (Apache access logs, Was logs, Application logs , log4J logs, JMeter logs and traces, …).
The method is :

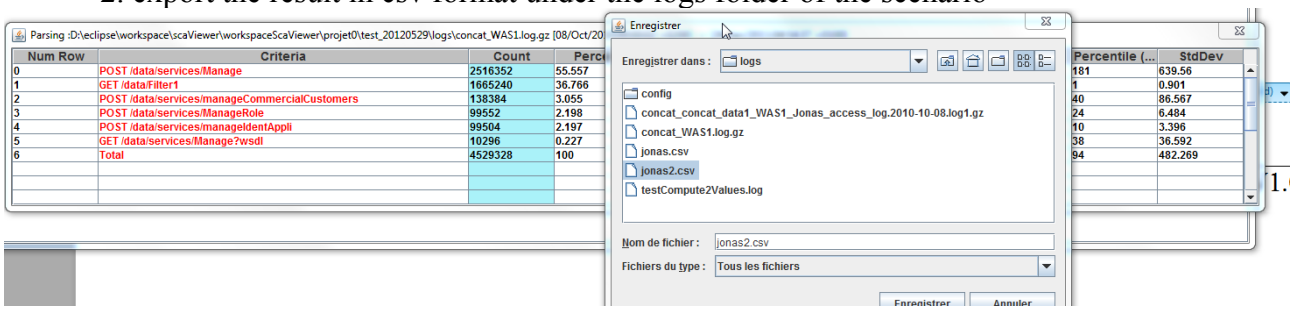
1. statically parse with the tool ScaFileStats
2. export the result in csv format under the logs folder of the scenario
3. configure ScaLogParser (from scratch or from an existing template)
4. configure the Pivot Panel of ScaLogParser importing the csv file saved in point 2

Below the screen-shots of the method for the file **concat_WAS1.log.gz** (23 Mo zipped, 440 Mo unzipped, 4 500 000 rows)

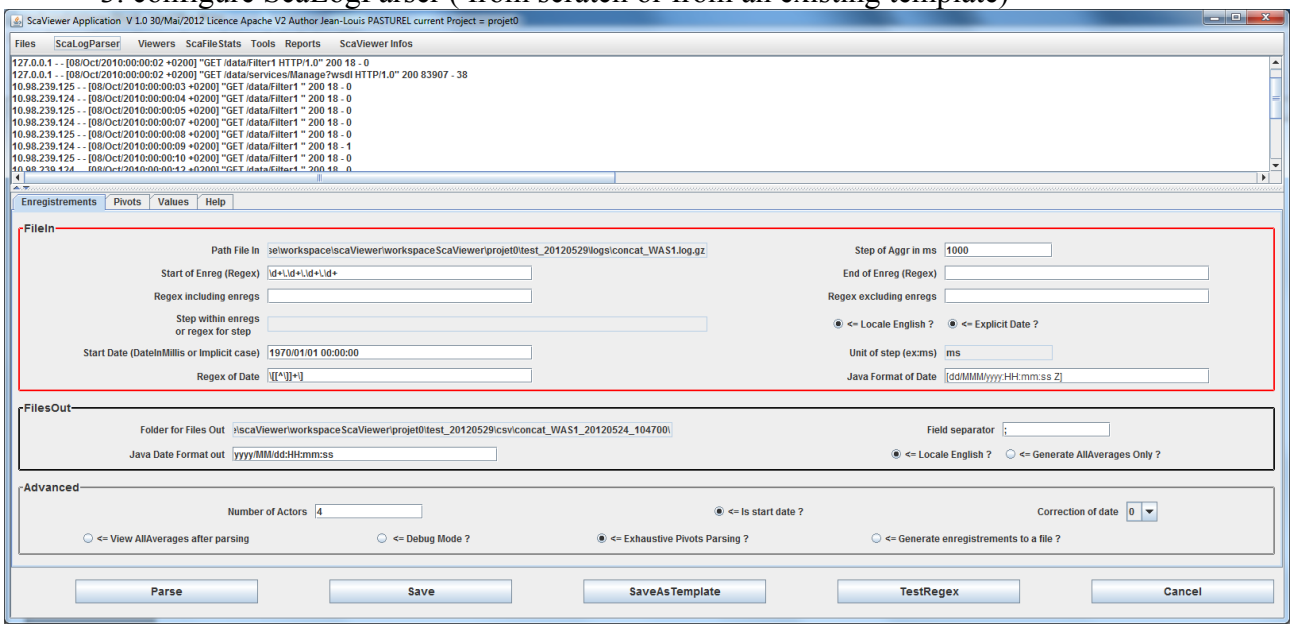
1. statically parse with the tool ScaFileStats

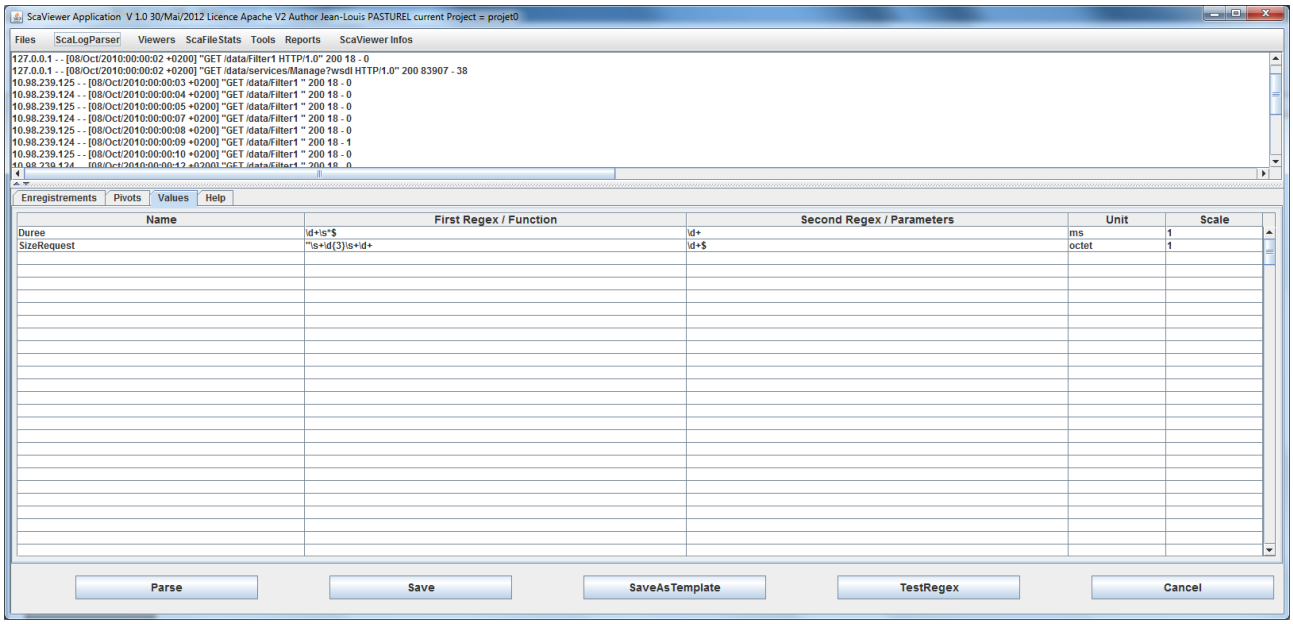


2. export the result in csv format under the logs folder of the scenario

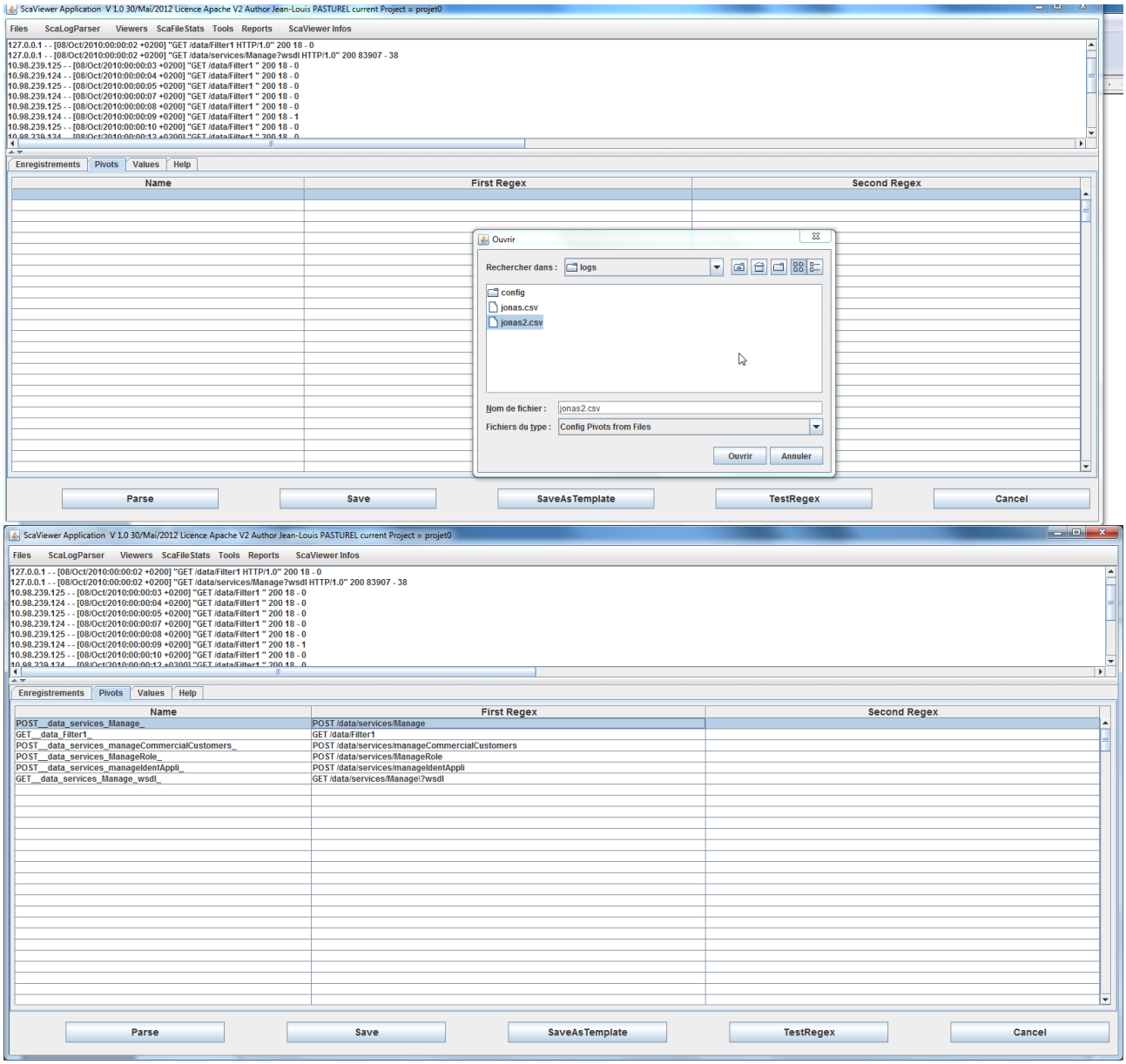


3. configure ScaLogParser (from scratch or from an existing template)

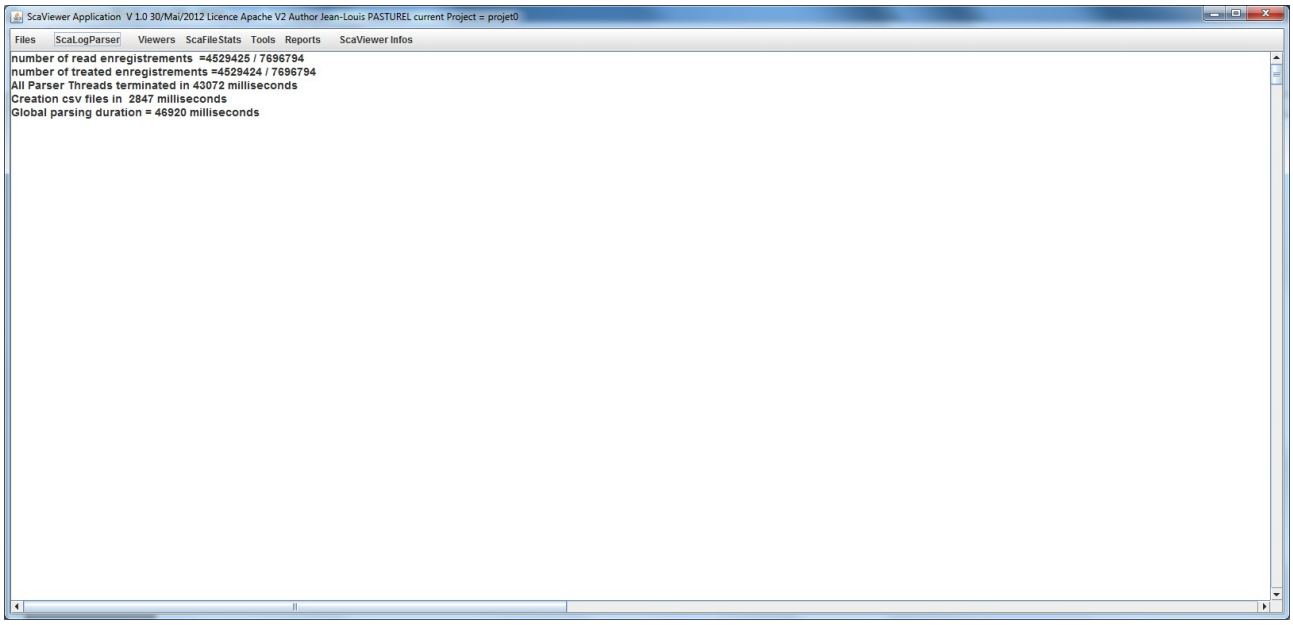




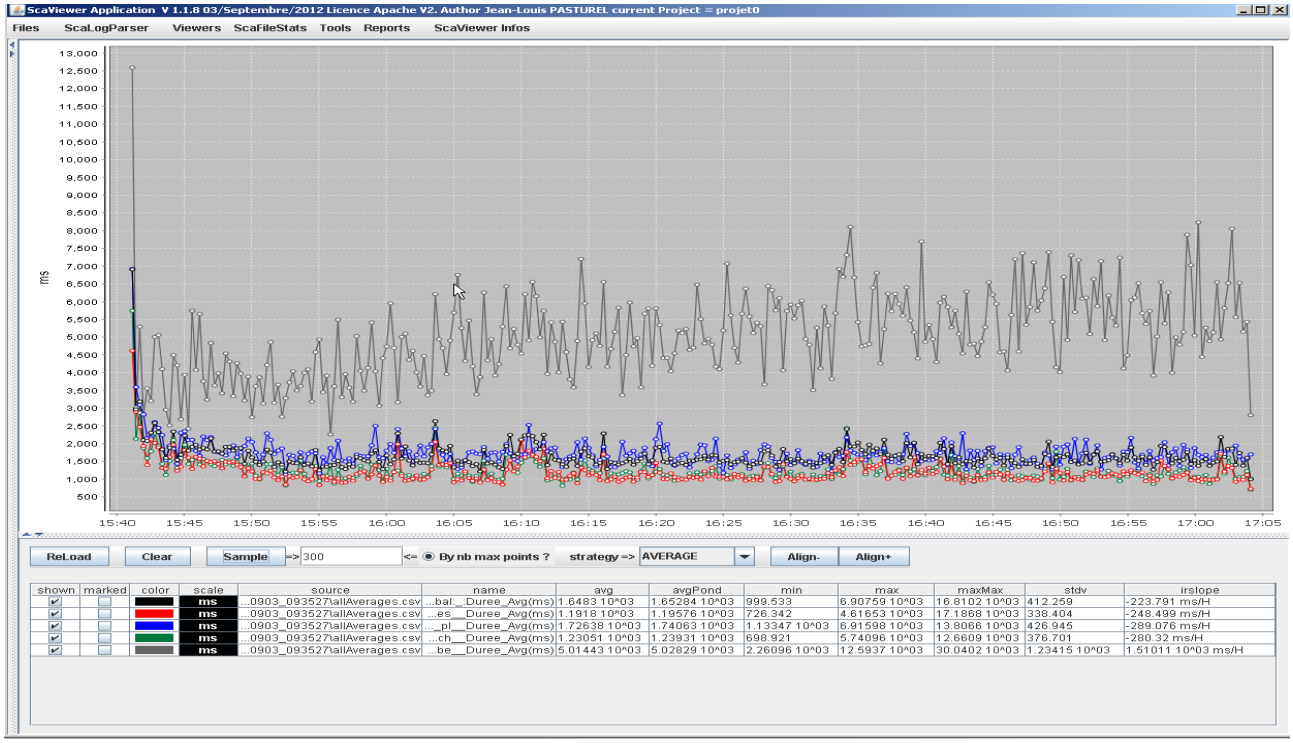
4. configure the Pivot Panel of ScaLogParser importing the csv file saved in point 2


ScaViewer Application V 1.0 30/Mai/2012 Licence Apache V2 Author Jean-Louis PASTUREL current Project = projet0
Files ScaLogParser Viewers ScaFileStats Tools Reports ScaViewer Infos
127.0.0.1 - - [08/Oct/2010:00:00:02 +0200] "GET /data/Filter1 HTTP/1.0" 200 18 - 0
127.0.0.1 - - [08/Oct/2010:00:00:02 +0200] "GET /data/services/Manage?wsdl HTTP/1.0" 200 83907 - 38
10.98.239.125 - - [08/Oct/2010:00:00:03 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.124 - - [08/Oct/2010:00:00:04 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.125 - - [08/Oct/2010:00:00:05 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.124 - - [08/Oct/2010:00:00:07 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.125 - - [08/Oct/2010:00:00:08 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.124 - - [08/Oct/2010:00:00:09 +0200] "GET /data/Filter1 " 200 18 - 1
10.98.239.125 - - [08/Oct/2010:00:00:10 +0200] "GET /data/Filter1 " 200 18 - 0
Enregistrements Pivots Values Help
Name First Regex Second Regex
Ouvrir
Rechercher dans : logs
config
jonas.csv
jonas2.csv
Nom de fichier : jonas2.csv
Fichiers du type : Config Pivots from Files
Ouvrir Annuler
Parse Save SaveAsTemplate TestRegex Cancel
ScaViewer Application V 1.0 30/Mai/2012 Licence Apache V2 Author Jean-Louis PASTUREL current Project = projet0
Files ScaLogParser Viewers ScaFileStats Tools Reports ScaViewer Infos
127.0.0.1 - - [08/Oct/2010:00:00:02 +0200] "GET /data/Filter1 HTTP/1.0" 200 18 - 0
127.0.0.1 - - [08/Oct/2010:00:00:02 +0200] "GET /data/services/Manage?wsdl HTTP/1.0" 200 83907 - 38
10.98.239.125 - - [08/Oct/2010:00:00:03 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.124 - - [08/Oct/2010:00:00:04 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.125 - - [08/Oct/2010:00:00:05 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.124 - - [08/Oct/2010:00:00:07 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.125 - - [08/Oct/2010:00:00:08 +0200] "GET /data/Filter1 " 200 18 - 0
10.98.239.124 - - [08/Oct/2010:00:00:09 +0200] "GET /data/Filter1 " 200 18 - 1
10.98.239.125 - - [08/Oct/2010:00:00:10 +0200] "GET /data/Filter1 " 200 18 - 0
Enregistrements Pivots Values Help
Name First Regex Second Regex
POST__data_services_Manage_ POST /data/services/Manage
GET__data_Filter1_ GET /data/Filter1
POST__data_services_manageCommercialCustomers_ POST /data/services/manageCommercialCustomers
POST__data_services_ManageRole_ POST /data/services/ManageRole
POST__data_services_manageIdentAppli_ POST /data/services/manageIdentAppli
GET__data_services_Manage_wsdl_ GET /data/services/Manage\?wsdl
Parse Save SaveAsTemplate TestRegex Cancel

and parse :


ScaViewer Application V 1.0 30/Mai/2012 Licence Apache V2 Author Jean-Louis PASTUREL current Project = projet0
Files
ScaLogParser
Viewers
ScaFileStats
Tools
Reports
ScaViewer Infos
number of read enregistrements =4529425 / 7696794
number of treated enregistrements =4529424 / 7696794
All Parser Threads terminated in 43072 milliseconds
Creation csv files in 2847 milliseconds
Global parsing duration = 46920 milliseconds

and view :


ScaViewer Application V 1.1.8 03/Septembre/2012 Licence Apache V2. Author Jean-Louis PASTUREL current Project = projet0
Files
ScaLogParser
Viewers
ScaFileStats
Tools
Reports
ScaViewer Infos
ms
ReLoad
Clear
Sample
300
By nb max points ?
strategy =>
AVERAGE
Align-
Align+
shown
marked
color
scale
source
name
avg
avgPond
min
max
maxMax
stdv
irslope

5 Annexe

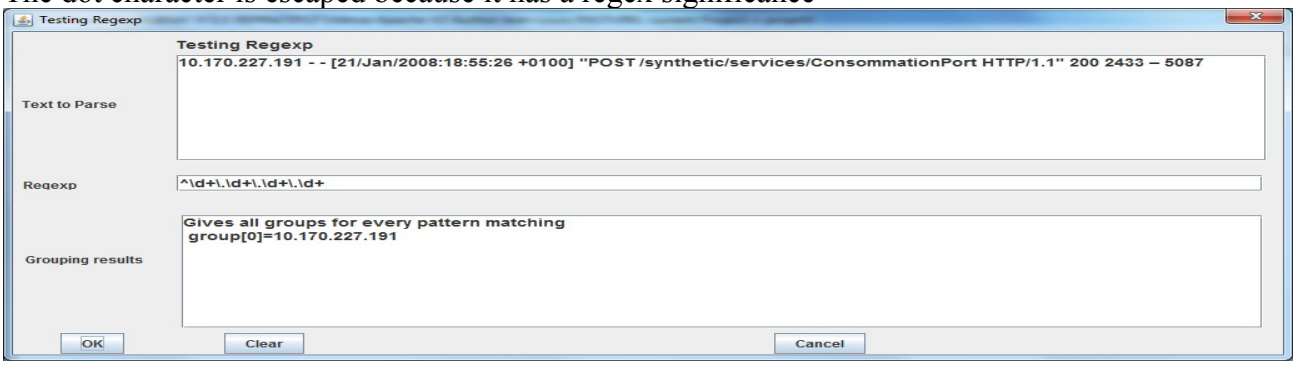
5.1 Examples of Java/Perl regex

Below some examples of regex to extract datas from a line:

```
10.170.227.191 - - [21/Jan/2008:18:55:26 +0100] "POST /synthetic/services/ConsommationPort
HTTP/1.1" 200 2433 – 5087
```

5.1.1 Regexp : ^\d+\.\d+\.\d+\.\d+

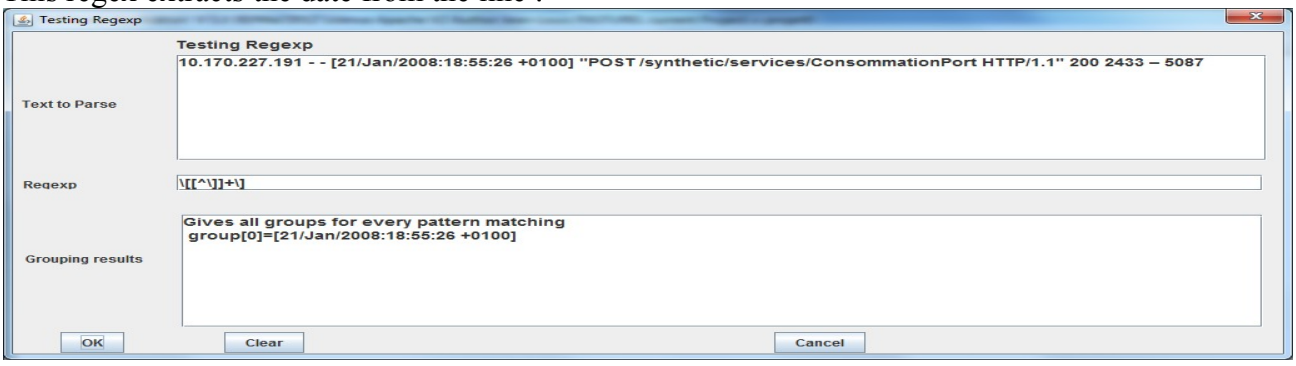
This regexp reads the line from the beginning of the line (character ^) , expects one or several numbers until a dot (**\d+\.**) , this repeated three times, and finally expects one or several more numbers (**\d+**).
The dot character is escaped because it has a regex significance



so you extract the IP address at the beginning of the line.

5.1.2 Regexp : \[[^\]]+\]

This regexp, reads from the first opening square bracket (**\[**) that must be escaped because it has a regex significance , and after it takes all characters that **are not** closing square bracket, note that in this context, the ^ character is the negation operator when it is not the first character of the regex=> ([^\]]+) , and the matching is terminated by the closing square bracket (**\]**)
This regex extracts the date from the line :

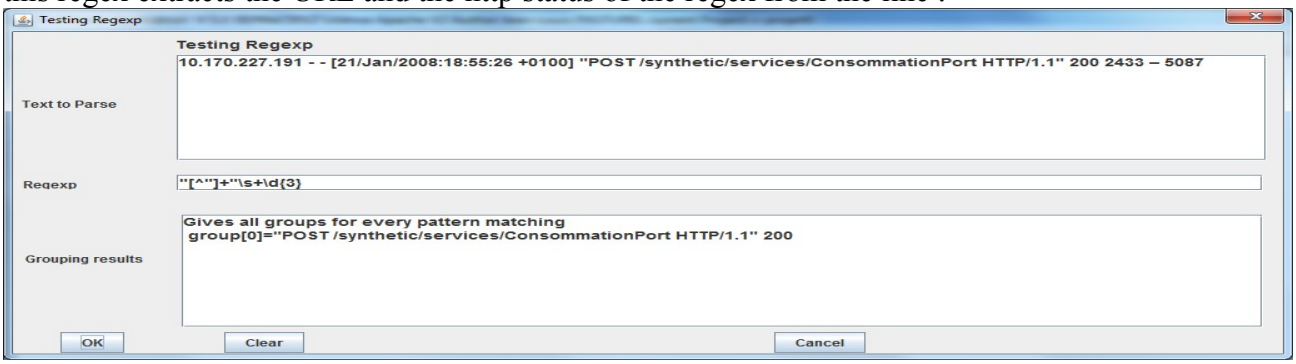


Note : Be care of the difference of signification of the ^ character when it is the first character of the regex, and when it is somewhere else.

5.1.3 Regexp : "[^"]+"\s+\d{3}

That must be read from the first double quotation marks (**"**) and after all characters that are not double quotation marks (**[^"]+**) , after a double quotation mark (**"**) , after one or more space or tab characters (**\s+**), and terminated by 3 digits (**\d{3}**) .

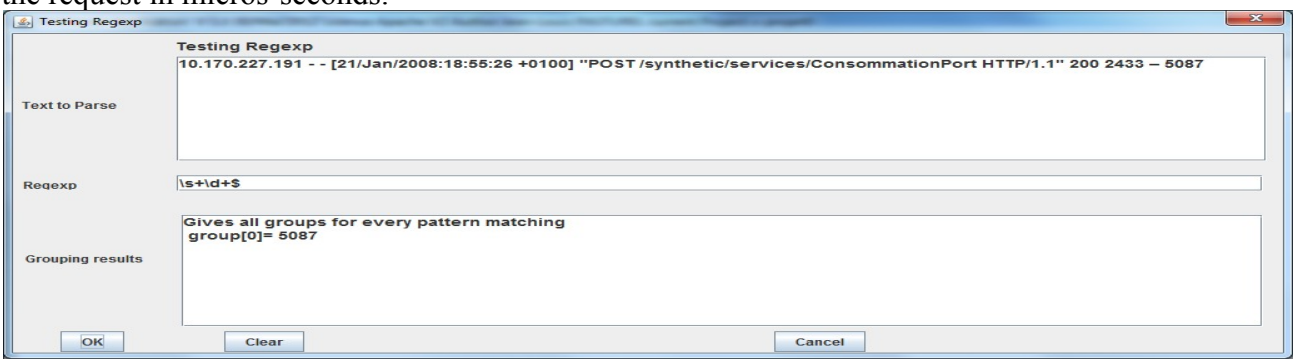
this regex extracts the URL and the http status of the regex from the line :



5.1.4 Regexp : \s+\d+$

This regexp reads beginning from the end of the line (last character is designated by **$**), before the end of the line one or more digits (**\d+**) and before one or more space or tab characters (**\s+**).

this regex extracts the last number in the line, for this Apache logs, it corresponds to the duration of the request in micros-seconds.



5.2 Use of key word “function” in swingScaViewer/scaLogParser

5.2.1 File to treat

Extract of the file to treat

```
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:40 val1= 10.3 val2=15
```

```
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:42 val1= 10.3 val2=17
2008/01/27 17:49:42 val1= 10.3 val2=17
2008/01/27 17:49:43 val1= 10.3 val2=18
2008/01/27 17:49:46 val1= 10.3 val2=19
2008/01/27 17:49:47 val1= 10.3 val2=19
2008/01/27 17:49:47 val1= 10.3 val2=19
2008/01/27 17:49:48 val1= 10.3 val2=20
2008/01/27 17:49:48 val1= 10.3 val2=16
2008/01/27 17:49:48 val1= 10.3 val2=16
2008/01/27 17:49:48 val1= 10.3 val2=16
```

Note the space after val1= .
The goal of the function is to get the result of the 4 operations (+,-,*,/) between val1 and val2

5.2.2 Classes Scala "myPlugins"

Below the source of **ConcCompute2Values** class :

- **ConcCompute2Values.scala**

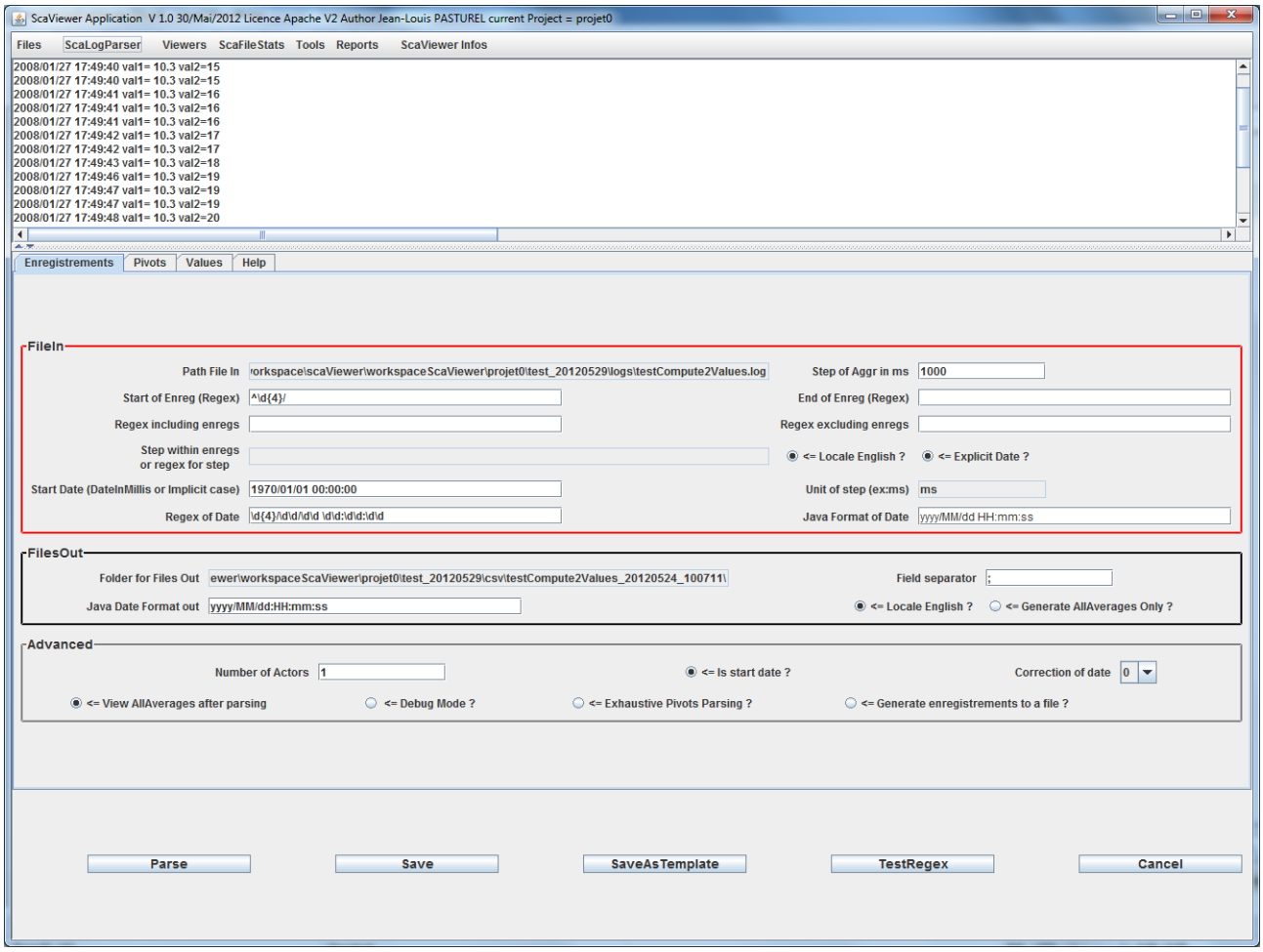
```
class ConcAdd2Values {
def metInit(tab:Array[String]=null) {
    // To reinitialise static variable if necessary
    // Nothing to do here
  }

      /**
      *tabStr(0) is the record to be treated. Afterwards the regex by tuple of 2
items for an extraction in two phases
      * regex tabStr(1) and tabStr(2) to extract the first value
      * regex tabStr(3) and tabStr(4) to extract the second value
      * tabStr(5) is the operand ("+","-","*","/")
      * @param tabStr
      * @return
      */
  def retour(tabStr:Array[String]):Double=
  {

            var retour=Double.NaN

            // extract first value
            var regex1=tabStr(1).r
            var ext1=regex1.findFirstIn(tabStr(0))
            var val1=0D
            var val2=0D
            if (None != ext1) {
                  val regex2=tabStr(2).r
                  val ext2=regex2.findFirstIn(ext1.get)
                  if (None != ext2) {
                        val1 = ext2.get.toDouble
                  } else {
                        return Double.NaN
```

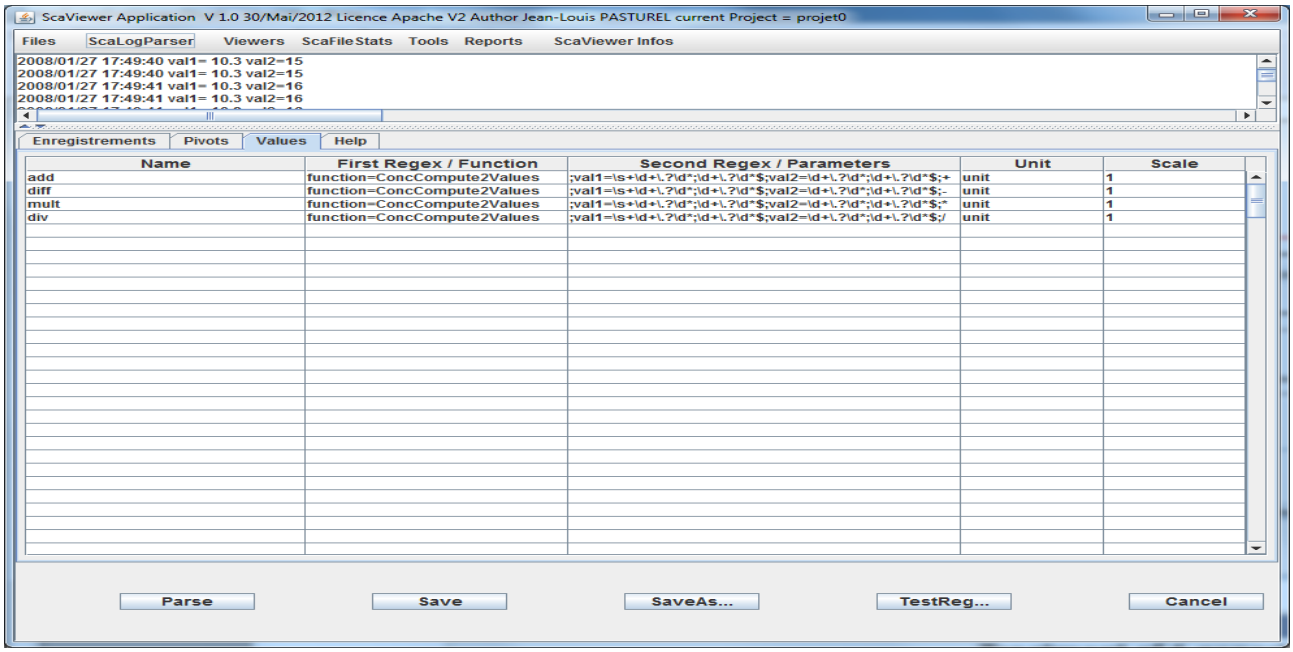
```
                  }
            } else {
                  return Double.NaN
            }

// extract second value
            regex1=tabStr(3).r
            ext1=regex1.findFirstIn(tabStr(0))

            if (None != ext1) {
                  val regex2=tabStr(4).r
                  val ext2=regex2.findFirstIn(ext1.get)
                  if (None != ext2) {
                        val2 = ext2.get.toDouble
                  } else {
                        return Double.NaN
                  }
            } else {
                  return Double.NaN
            }

            // return the result of the operation
            tabStr(5) match {
            case "+" => val1+val2
            case "-" => val1-val2
            case "/" => if (val2 !=0) val1/val2 else Double.NaN
            case "*" => val1 * val2
            case _ => Double.NaN
      }

  }

}
```

Note : the Conc prefix of the class means that this class can be used in multi threaded parsing (nb Actors >1)
The Scala code of this class is in the directory **<swingScaViewer_Home>/myPlugins .**
The binary of this class is in the jar archive **<swingScaViewer_Home>/myPlugins/myPlugins.jar**

5.2.3 Configuration of swingScaViever/ScaLogParser

The screens below show the configuration to parse this file :


ScaViewer Application V 1.0 30/Mai/2012 Licence Apache V2 Author Jean-Louis PASTUREL current Project = projet0
Files ScaLogParser Viewers ScaFileStats Tools Reports ScaViewer Infos
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:42 val1= 10.3 val2=17
2008/01/27 17:49:42 val1= 10.3 val2=17
2008/01/27 17:49:43 val1= 10.3 val2=18
2008/01/27 17:49:46 val1= 10.3 val2=19
2008/01/27 17:49:47 val1= 10.3 val2=19
2008/01/27 17:49:47 val1= 10.3 val2=19
2008/01/27 17:49:48 val1= 10.3 val2=20
Enregistrements Pivots Values Help
FileIn
Path File In
Start of Enreg (Regex)
Regex including enregs
Step within enregs or regex for step
Start Date (DateInMillis or Implicit case)
Regex of Date
Step of Aggr in ms
End of Enreg (Regex)
Regex excluding enregs
<= Locale English ?
<= Explicit Date ?
Unit of step (ex:ms)
Java Format of Date
FilesOut
Folder for Files Out
Java Date Format out
Field separator
<= Locale English ?
<= Generate AllAverages Only ?
Advanced
Number of Actors
<= Is start date ?
Correction of date
<= View AllAverages after parsing
<= Debug Mode ?
<= Exhaustive Pivots Parsing ?
<= Generate enregistrements to a file ?
Parse
Save
SaveAsTemplate
TestRegex
Cancel


ScaViewer Application V 1.0 30/Mai/2012 Licence Apache V2 Author Jean-Louis PASTUREL current Project = projet0
Files ScaLogParser Viewers ScaFileStats Tools Reports ScaViewer Infos
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:40 val1= 10.3 val2=15
2008/01/27 17:49:41 val1= 10.3 val2=16
2008/01/27 17:49:41 val1= 10.3 val2=16
Enregistrements Pivots Values Help
Name
First Regex / Function
Second Regex / Parameters
Unit
Scale
add function=ConcCompute2Values ;val1=\s+\d+\.?\d*;\d+\.?\d*$;val2=\d+\.?\d*;\d+\.?\d*$;+ unit 1
diff function=ConcCompute2Values ;val1=\s+\d+\.?\d*;\d+\.?\d*$;val2=\d+\.?\d*;\d+\.?\d*$;- unit 1
mult function=ConcCompute2Values ;val1=\s+\d+\.?\d*;\d+\.?\d*$;val2=\d+\.?\d*;\d+\.?\d*$;* unit 1
div function=ConcCompute2Values ;val1=\s+\d+\.?\d*;\d+\.?\d*$;val2=\d+\.?\d*;\d+\.?\d*$;/ unit 1
Parse
Save
SaveAs...
TestReg...
Cancel

5.2.4 Visualization

After clicking on Parse, the allAverages are graphed as shown below