

SEP #44: Scilab History Browser

Title: Scilab History Browser

Version: 1

Author: Sylvestre Koumar – sylvestre.koumar@scilab.org

Review:

Commented:

State: Draft

Scilab-Version: 5.3

Vote: « No: discussions and suggestions »

Created: Monday, 7th April 2010

Abstract

This SEP proposes to create a graphical history browser for Scilab.

This new application will use the Scilab docking system (introduced in Scilab 5.0).

Scilab history browser

History browser is a feature that allows the user to view, to search and to recall previously run statements. We baptize this feature '**Command history**'.

SCILAB HISTORY BROWSER: COMMAND HISTORY

Command history shows all the statements entered in a Scilab session. From the Command history the user can view, search and recall previously run statements.

How to launch Command history:

- through the menu Preferences, choose **Applications => Command history**
- from the command line by typing: *commandHistory*

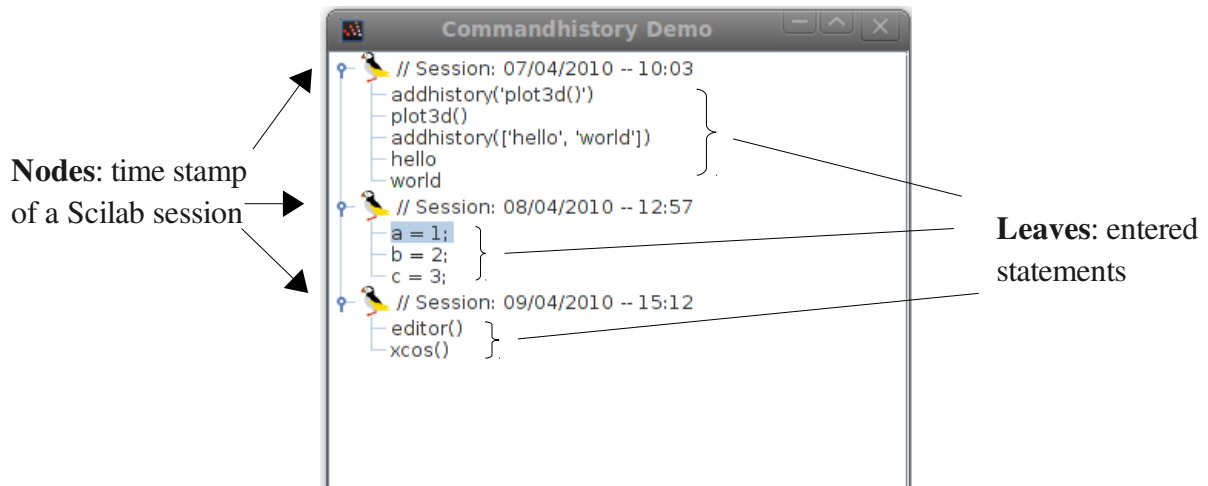
Rationale

Developed in Java, Command history will (in addition of existing functions) enable the user to:

- visualize entered statements
- export entered statements into a file
- find an entered statement
- print entered statements

Command history prototype

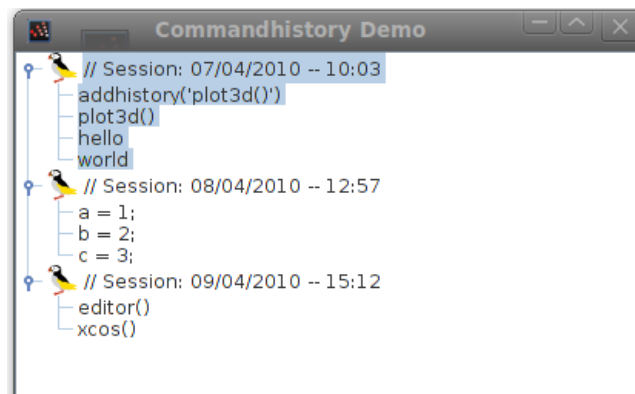
Command history will be a dockable window using the Scilab docking system. This window will contain an arborescence which will be organized in the following way:



- Command history window -

Selection mode

Clicking (right or left click) on a **node** will automatically select the node and it's children.



Multiple selection is available (Ctrl + click).



User interaction description (mouse left-click)

Node type	Click		Double-click	
	<i>Leaf</i>	<i>Node</i>	<i>Leaf</i>	<i>Node</i>
text	select the leaf	select the node and it's children	Executes selection in Scilab	Executes children of the node in Scilab
icon	-	select the node and it's children	-	Open or close the node & Executes children of the node in Scilab
node developer	Open or close the corresponding node		The same effect of 2 single clicks	

Drag and Drop

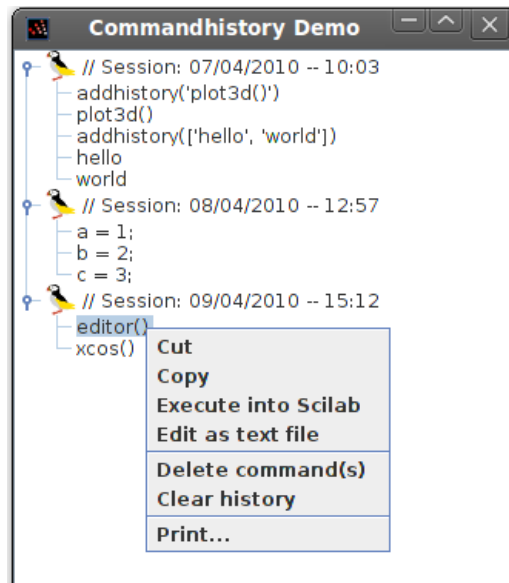
This feature will allow the user to drag and drop selected statements into the Scilab console.

Drag and drop from the console to Command history is not allowed.

Drag and drop in the arborescence itself is not allowed.

User interaction description (mouse right-click)

Features available through a context menu.

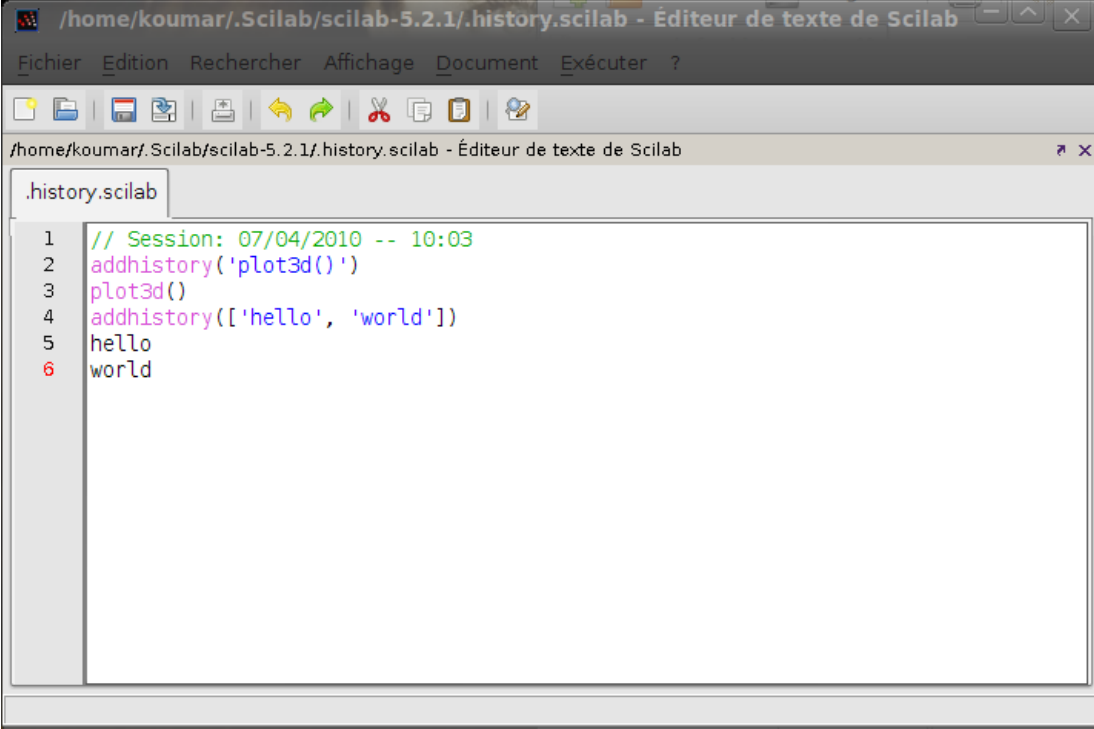


- Right click action in Command history -

Note: behavior of right-click is the same for (text, icon and node developer). Also features available through the context menu will only occur for the selected items in the arborescence.

<i>Features</i>	<i>Actions</i>
<i>Cut</i>	Removes selected statement(s) and put them in the clipboard.
<i>Copy</i>	Copy selected statements and put them in the clipboard.
<i>Execute into Scilab</i>	Executes selected statement(s) in Scilab console.
<i>Edit as text file</i>	Opens Editor with the selected statement(s). [1]
<i>Delete command(s)</i>	Deletes selected statement(s) of history. (a confirmation window will occur for deletion, in this window a check box will ask if the confirmation window should occur for the next deletion)
<i>Clear history</i>	Deletes all statements of history. (a confirmation window will occur, in this window a check box will ask if the confirmation window should occur for the next clear action)
<i>Print...</i>	Prints the history.

[1] Edit as text file ('*Export...*' of the context menu)



The screenshot shows a text editor window titled "/home/koumar/.Scilab/scilab-5.2.1/.history.scilab - Éditeur de texte de Scilab". The window has a menu bar with "Fichier", "Edition", "Rechercher", "Affichage", "Document", and "Exécuter ?". Below the menu bar is a toolbar with icons for file operations and editing. The main text area contains the following code:

```
1 // Session: 07/04/2010 -- 10:03
2 addhistory('plot3d()')
3 plot3d()
4 addhistory(['hello', 'world'])
5 hello
6 world
```

- Editor window -

Tool bar

A tool bar in the Command history window can be useful, this one will have following buttons:

- Cut
- Copy
- Undo
- Redo
- Find
- Delete
- Clear
- Settings
- Print
- Help

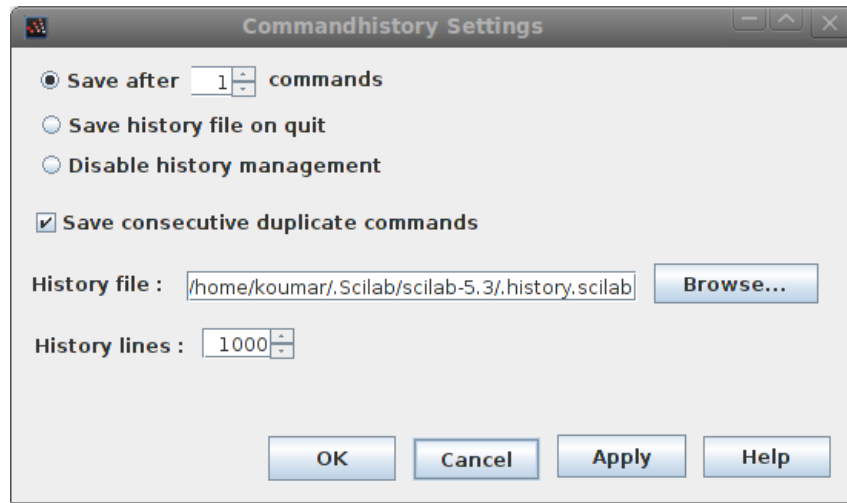
This tool bar can be shown or hidden by a defined shortcut.

Command history settings

We have 2 solutions for the presentation of Command history settings:

- by using a tool bar in Command history window, with a menu item called 'Settings'
- with a button 'Settings' at the bottom of the Command history window

Both solutions will open a window which allows the user to configure the history browser.



- Command history Settings window -

Save after n commands

Save the history file after a given number of statements.

Will be represented by a radio button and a spinner.

Save history file on quit

Save (automatically) the history file while we quit Scilab.

Will be represented by a radio button.

Disable history management

Will be represented by a radio button.

Save consecutive duplicate commands

Will be represented by a check box.

History file

Set the file to use for history. Button 'Browse...' will open a file chooser to select a file.

Default file is `'/home/user/.Scilab/scilab-5.3/.history.scilab'`.

Will be represented by a text field and a button.

History lines

Set the maximum number of statements managed by Command history.

Will be represented by a spinner.

Help button

Opens Scilab's help for Command history.

Find a command

User will have to type the name of the seeking statement, if the typed name matches with one or many statements in the arborescence, these statements will be highlighted.

Another behavior can be, when the user types in the text field, the first matching statement in the arborescence is highlighted.

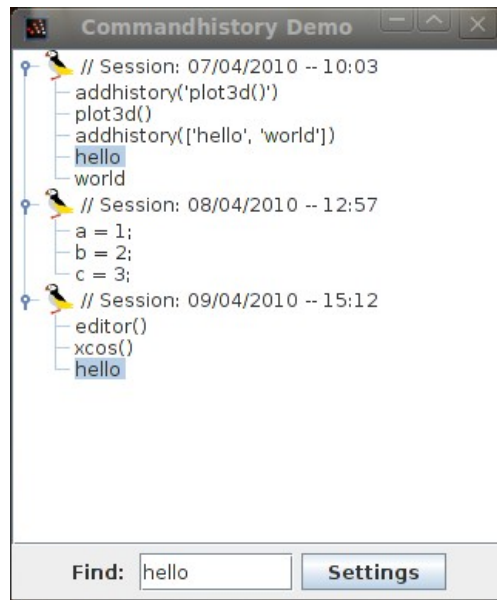
If there is many matching statements the user can look for other statements with buttons '*next*' and '*previous*'. Another button called '*highlight all*' will highlight all the matching statements.

If we do not have matching statements a message at the bottom of the window will occur ('Statement not found').

We can hide or show the find bar from the tool bar of Command history window or with the shortcut Ctrl-F.

Note: this function of search is inspired of Firefox's search function.

Will be represented by a text field.



- Find representation -

Evolution for future versions

Paste of several lines in Scilab console

While a statement of several lines is paste in Scilab console, Command history displays this bloc of statements in a single leaf instead of one leaf for each statement's line.

Warnings for statements

If a statements is not valid or has generated an error, a tag indicating a warning in front of the concerned statement can be useful.

Undo/Redo actions

Manage undo/redo action on the history arborescence.

Syntactic color

Use syntactic color on history arborescence.

Warning & error messages

Not defined yet.

Appendix

DESCRIPTION OF THE CURRENT HISTORY MANAGEMENT

The current history management is organized around several functions (only available in the Scilab console in textual mode).

These functions are:

addhistory

addhistory (*newLine*)

Adds new line(s) to current history.

Input: we should specify the line we want to add.
newLine should be a string or a string matrix

Output: nothing, will add the given line to current history.

→ `addhistory('plot3d()')`
Will add the line 'plot3d()' to current history.
→ `addhistory(['hello', 'world'])`
Will add lines 'hello' and 'world' to current history.

displayhistory

displayhistory ()

Displays current history.

Input: -

Output: nothing, will only display current history in the prompt.

→ `displayhistory()`

example:

```
0 : // Début de la Session : Ven Avr 9 09:34:48 2010
1 : addhistory('plot3d()')
2 : plot3d()
3 : addhistory(['hello', 'world'])
4 : hello
5 : world
6 : displayhistory()
```

gethistory

```
history gethistory ()  
line gethistory (lineNumber)
```

Returns current history or a line from current history.

Input: we should specify the number of the line we want to get.
lineNumber should be an integer

Output: - *history* is a string matrix
- *line* is a string

→ myHistory = gethistory ()
Will return current history into the string matrix myHistory.

→ myLine = gethistory (3)
Will return the 3rd line of current history into the string myLine.

gethistoryfile

```
filename gethistoryfile ()
```

Get the name of the file used for Scilab's history.

Input: -

Output: *filename* is a string

→ myFile = gethistoryfile ()
Will return the absolute path of the file used for Scilab's history in myFile.

historymanager

```
state historymanager ()  
state historymanager (stateMode)
```

Enable or disable history manager.

Input: we should specify the state of the history manager we want.
stateMode should be a string ('on' or 'off')

Output: *state* is a string ('on' or 'off')

→ myState = historymanager ()
Will return the current state of history manager.

→ myState = historymanager ('on')
Will enable history manager.

historysize

`size historysize()`

Get number of lines in current history.

Input: -

Output: size is an integer

→ `historySize = historysize()`

Will return the number of lines of current history in historySize.

loadhistory

`loadhistory()`

`loadhistory(file)`

Load a history file.

Input: we should specify the file's name to load.
file should be a string (file's absolute path)

Output: nothing, will load the history file (by default, history file is `SCIHOME+/.history.scilab'`).

→ `loadhistory()`

Will load the default history file (`SCIHOME+/.history.scilab'`).

→ `loadhistory(myFile)`

Will load myFile for the current history.

removelinehistory

`removelinehistory(lineNumber)`

Remove a line in history from the given line number.

Input: we should specify the number of the line we want to remove in history.
lineNumber should be an integer

Output: nothing, will remove a line in history from the given line number.

→ `removelinehistory(2)`

Will remove the 2nd line of current history.

resethistory

`resethistory()`

Deletes all entries of current history.

Input: -

Output: nothing, will delete all entries of current history.

→ `resethistory()`

Will delete all entries of current history.

saveafterncommands

```
saveafterncommands (numberOfStatements)
currentNumber saveafterncommands ()
```

Save the history file after a given number of statements.

Note: Use this option if you do not want to risk losing entries to the saved history because of an abnormal termination, such as a power failure.

Input: we should specify the number of statements we want for the history file saving.
numberOfStatements should be an integer

Output: *currentNumber* is an integer (0 is the default value)

→ `saveafterncommands (5)`

Will set the number of statements to 5 for history file saving.

→ `nbStatements = saveafterncommands ()`

Will return the current number of statements for the history file saving.

saveconsecutivecommands

```
saveconsecutivecommands (saveConsecutive)
consecutiveMode saveconsecutivecommands ()
```

Save consecutive duplicate commands.

Input: we should specify if consecutive executions of the same statement should be saved or not in the history file.
saveConsecutive should be a boolean

Output: *consecutiveMode* is a boolean (false is the default value)

→ `saveconsecutivecommands (%T)`

Will enable the save of consecutive duplicate commands in the history file.

→ `consecutiveMode = saveconsecutivecommands ()`

Will return if the save of consecutive duplicate commands in the history file is enable or not.

savehistory

```
savehistory ()
savehistory (file)
```

Save the current history in a file.

Input: we should specify the file's name to save.
file should be a string (file's absolute path)

Output: nothing, will save the current history in a file (by default, history file is `SCIHOME+/.history.scilab'`).

→ `savehistory ()`

Will save current history in the default history file (`SCIHOME+/.history.scilab'`).

→ `savehistory (myFile)`

Will save the current history in `myFile`.

sethistoryfile

```
sethistoryfile ()  
sethistoryfile (file)
```

Set a file for Scilab history.

Input: we should specify the file to be set for history.
file should be a string (file's absolute path)

Output: nothing, will set a file for Scilab history (by default, settled file is SCIHOME+/.history.scilab').

→ sethistoryfile ()

Will set the default file for history (SCIHOME+/.history.scilab').

→ sethistoryfile (myFile)

Will set myFile for history.

Example Usage

Changelog

1.0 – Sylvestre Koumar <sylvestre.koumar@scilab.org> Initial version

Copyright

This document has been placed under the licence CeCILL-B.