Flytrap: Tcl Debugging Tools

Version 0.2

Alex Baker

https://github.com/ambaker1/flytrap

June 2, 2023

Abstract

Say goodbye to debugging with countless *puts* statements, and say hello to "flytrap"!

Pausing a Script

The *pause* command pauses a Tcl script, prints the file and line number, and enters command-line mode, allowing the user to query variables and insert code into an analysis. If the command entered while paused returns an error, the error message will be displayed and the script will remain paused. If the command entered is "return", the pause will be exited and the corresponding result and options will be passed to the caller. For example, a loop can be broken by entering *return -code break* in pause mode. Pressing enter with no commands will simply continue the script.

```
pause <$varName>
```

$varName	Optional variable name to store the info displayed on screen.

Example 1: Pausing an analysis

Code:

```
pause
```

Output:

```
PAUSED...
(line 407 file "C:/User/Documents/MyFile.tcl")
>
```

Note: If in interactive mode, there may not be a file to pause in. In this case, it will list the procedure or script where the pause occurred.

Advanced Tcl Debugger

The *flytrap* command parses a Tcl script, and prints out the evaluation steps and results if an error is reached. Additionally, if an error is reached, the script will pause at the line where the error occurred, allowing for interactive introspection of the problem, at the depth specified.

```
flytrap -file $filename <$depth> <$verbose>
flytrap -body $script <$depth> <$verbose>
```

$filename	File path of Tcl script to debug.
$script	Tcl script to debug.
$depth	Optional recursive depth to step into procedures (default 0).
$verbose	Optional flag to always print out all steps and results (default 0).

Example 2: Verbose evaluation of a procedure

Code:

```
set DEPTH 1
set VERBOSE true
proc add {a b} {
    return [expr {$a + $b}]
}
set a 5
set b 7
flytrap -body {
    add [expr {$a*2}] $b
} $DEPTH $VERBOSE
```

Output:

```
> expr {$a*2}
10
> add 10 7
  > expr {$a + $b}
  17
  > return 17
  17
17
```

Variable Viewer

The command *varViewer* is a TclOO class that creates widget objects that display the values of variables.

```
varViewer new $varList <$title>
varViewer create $name $varList <$title>
```

$name	Object name.
$varList	List of variables to view.
$title	Optional title. Default “Workspace”.

Example 3: Monitoring variable values

Code:

```
package require tin
tin import flytrap
set i 0
set j 0
varViewer new {i j}
for {set i 0} {$i < 1000} {incr i} {
    for {set j 0} {$j < 1000} {incr j} {
        update
    }
}
```

The command *viewVars* is short-hand for creating a *varViewer* object for all variables in the current scope.

```
viewVars
```

Example 4: Workspace viewer

Code:

```
set a 5
set b 7
array set c {1 5 2 6}
viewVars
pause
```

Output: