error-gui.ss 2020-01-02 16:09:19

```
;;;;  debug.ss
;;;;  created 05.10.1
;;;;     destroy not work !!
;;;;  Not destroy version
;;;;

(define *win* '.errorwindow)
(define *hookend* #f)
(in-package "error")

(define (ewin)
   (let* ((top *win*))
    (if (winfo 'exists *win*)
        (wm 'deiconify *win*)
       (let* ((lab1 (tk-name top ".lab_er"))
              (lab2 (tk-name top ".lab_er2"))
              (bt1 (tk-name top ".bt1"))
          )
         (toplevel *win*)
         (button  bt1 :text "Quit" :command
               `(set! *hookend* #t))
         (grid  bt1  :row 0 )
         (label lab1 )
         (label lab2 )
         (grid  lab1  :row 1 :sticky 'nws)
         (grid  lab2  :row 2 :sticky 'nws)
          (make-lbsc top "Environment" 4 )
          (make-lbsc top "Function history" 6 )
        )
    )
  *win*
))

(define (ehook enum obj)
  (let* ((his  (call-history))(top 0)(bt1 0) (err
  0))
    (set! his (cdr his)) ;;; needs after
    (print-error enum obj)
    (set! top (ewin))
    ;;;(format #t "top ok~%")
   (pr-error top  enum obj)
  ;; (pr-tcl top)
   (lbsc-win top 4 (map env->dsp *errorenv*))
   (lbsc-win top  6 (his->dsp his))
   ;;;(tkwait 'visibility top)
  ;;; (focus top)
  ;;; (grab top)
   (wait-int top)
   (set! *errorhook* ehook)
))

(define (env->dsp x)
   (let* ((v (car x))(val (cdr x)))
     (format #f "~s : ~s" v val)
))

(define (his->dsp his)
   (let* ((n 0)(res ()))
     (while (pair? his)
       (push (format #f "~s : ~s" n (pop his)) res
       )
       (inc n)
     )
   (reverse res)
))



(define (pr-error top enum obj)
  (let* ((lab1 (tk-name top ".lab_er"))
         (lab2 (tk-name top ".lab_er2"))
         (ms (assq enum *system-error-msg*)))
    (if ms
        (set! ms (cdr ms))
        (set! ms "User Error")
    )
    (funcall lab1 'configure :text (tk-string "Err
    or " enum " : " ms) )
    (funcall lab2 'configure :text (format #f  "Ob
    ject : ~s " obj) )
))

(define (pr-tcl top )
  (let* ((lab (tk-name top ".lab_tcl")) (tc (tcl-l
  ast-return)))
    (label lab :text (format #f "TCL LAST Message
     Level ~s : ~a"
              (car tc)(cadr tc)))
    (grid lab :row 3 :sticky 'nws)
))
```

```
(define (make-lbsc hf title row  )
   (let* ( (hflis  (tk-name hf  '.dlm row))
          (hf-ysc  (tk-name hf '.dlmysc row))
          (hf-lab  (tk-name hf '.lab row))
         )
   (listbox  hflis :width 60 :height 10 :exportsel
   ection #t
        :selectMode 'extended
        :yscrollcommand (lambda l (apply (function
         hf-ysc)  'set l)))
   (scrollbar hf-ysc :orient 'vertical
            :command (lambda l (apply (function hf
            lis) 'yview l)))
   (label  hf-lab :text title )
   (grid  hf-lab :sticky 'news :row row :column 0)
   (inc row)
   (grid  hflis  :sticky 'news :columnspan 2 :row
    row )
   (grid  hf-ysc :sticky 'news :row row :column 1)
   (grid 'columnconfigure hf 0 :weight 1)
   (grid 'rowconfigure hf row :weight 1)
))

(define (lbsc-win hf row dsp)
  (let* ( (hflis  (tk-name hf  '.dlm row))
         )
   (funcall hflis 'delete 0  'end )
   (while (pair? dsp)
       (funcall hflis 'insert 'end (pop dsp))
   )
))

(define (wait-int top)
   (let* ((x 0))
      (set! *hookend* #f)
      (while (not *hookend*)
          (after 200)
          (update)
         ;;; (format #t "waiting sig ~s ~%" x)
      )
     (wm 'withdraw top)
     (format #t "out of error-gui~%" )
    (destroy top)
))

(set! *errorhook* ehook)
```