

File created: 5-Dec-2020 16:34:55 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOM
S>MEDLEY-35>ROOMS-WINDOW-TYPES.;2

previous date: 17-Aug-90 13:37:32 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-W
INDOW-TYPES.;1

Read Table: XCL

Package: ROOMS

Format: XCCS

; Copyright (c) 1987, 1988, 1990, 2020 by Venue & Xerox Corporation. All rights reserved.

```
(IL:RPAQQ IL:ROOMS-WINDOW-TYPESCOMS
  ((FILE-ENVIRONMENTS IL:ROOMS-WINDOW-TYPES)
   (IL:P (EXPORT ' (DEF-WINDOW-TYPE WINDOW-TYPE WINDOW-TYPE-PROP))
          (REQUIRE "ROOMS"))
   (IL:STRUCTURES WINDOW-TYPE)
   (IL:FUNCTIONS WINDOW-TYPE-PROP)
   (IL:DEFINE-TYPES IL:WINDOW-TYPES)
   (IL:VARIABLES *WINDOW-TYPES*)
   (IL:FUNCTIONS DEF-WINDOW-TYPE WINDOW-TYPE-NAMED WINDOW-TYPE WINDOW-TYPE-INTERNAL ABSTRACT-WINDOW
                 RECONSTITUTE-WINDOW)
   (IL:SEDIT-FORMATS DEF-WINDOW-TYPE)))

(DEFINE-FILE-ENVIRONMENT IL:ROOMS-WINDOW-TYPES :COMPILER :COMPILE-FILE
 :PACKAGE "ROOMS"
 :READTABLE "XCL")

(EXPORT ' (DEF-WINDOW-TYPE WINDOW-TYPE WINDOW-TYPE-PROP))

(REQUIRE "ROOMS")

(DEFSTRUCT (WINDOW-TYPE (:PRINT-FUNCTION (LAMBDA (WINDOW-TYPE STREAM DEPTH)
                                              (FORMAT STREAM "#<Window Type ~S>" (WINDOW-TYPE-NAME
                                                                    WINDOW-TYPE))))
  (NAME NIL :TYPE STRING)
  (DEPENDENCIES NIL :TYPE LIST)
  (RECOGNIZER NIL :TYPE FUNCTION)
  (ABSTRACTER NIL :TYPE FUNCTION)
  (RECONSTITUTER NIL :TYPE FUNCTION)
  (PLACER NIL :TYPE FUNCTION)
  (UPDATER NIL :TYPE FUNCTION)
  (PROPS NIL :TYPE LIST))

(DEFMACRO WINDOW-TYPE-PROP (WINDOW-TYPE PROP &OPTIONAL (NEW-VALUE NIL NEW-VALUE-SUPPLIED))
  (IF NEW-VALUE-SUPPLIED
    `(SETF (GETF (WINDOW-TYPE-PROPS ,WINDOW-TYPE)
                 ,PROP)
           ,NEW-VALUE)
    `(GETF (WINDOW-TYPE-PROPS ,WINDOW-TYPE)
           ,PROP)))

(DEF-DEFINE-TYPE IL:WINDOW-TYPES "Window types"
 :UNDEFINER (LAMBDA (NAME)
              (REMHASH NAME *WINDOW-TYPES*)))

(DEFGLOBALVAR *WINDOW-TYPES* (MAKE-HASH-TABLE :TEST 'EQ)
 "Hash table mapping from window type names to window type objects.")

(DEFDEFINER DEF-WINDOW-TYPE IL:WINDOW-TYPES (NAME &REST REST-KEYS &KEY DEPENDENCIES RECOGNIZER ABSTRACTER
                                              RECONSTITUTER PLACER UPDATER &ALLOW-OTHER-KEYS)

;;; defines a window type

(FLET ((KWOTE (X)
        ;; we want lambda expressions wrapped in FUNCTION and named functions just quoted
        (TYPECASE X
          (CONS (LIST (CASE (FIRST X)
                        ((LAMBDA IL:LAMBDA) 'FUNCTION)
                        (T 'QUOTE))
                  X)
                ((SATISFIES CONSTANTP) X)
                (T (LIST 'QUOTE X))))))
  `(SETF (GETHASH ',NAME *WINDOW-TYPES*)
    (MAKE-WINDOW-TYPE :NAME ',NAME :DEPENDENCIES ',DEPENDENCIES :RECOGNIZER ,(KWOTE RECOGNIZER)
                      ,@(WHEN ABSTRACTER
                          `(:ABSTRACTER ,(KWOTE ABSTRACTER)))
                      ,@(WHEN RECONSTITUTER
                          `(:RECONSTITUTER ,(KWOTE RECONSTITUTER)))
                      ,@(WHEN UPDATER
```

```

      `(:UPDATER , (KWOTE UPDATER)))
    , @ (WHEN PLACER
      `(:PLACER , (KWOTE PLACER)))
    :PROPS
    (LIST , @ (MAPCAR #'KWOTE (LET ((PROPS (COPY-LIST REST-KEYS))
      (DOLIST (KEYWORD ' (:DEPENDENCIES :RECOGNIZER :ABTRACTER
        :RECONSTITUTER :PLACER :UPDATER))
        (REMF PROPS KEYWORD))
      PROPS))))))

```

```

(DEFUN WINDOW-TYPE-NAMED (NAME &OPTIONAL NO-ERROR?)
  (OR (GETHASH NAME *WINDOW-TYPES*)
    (UNLESS NO-ERROR? (ERROR "No window type named ~S." NAME))))

```

```

(DEFUN WINDOW-TYPE (WINDOW &OPTIONAL NO-ERROR?)

```

;;; return the window type object for WINDOW.

```

  (LET ((CACHED-TYPE-NAME (IL:WINDOWPROP WINDOW 'WINDOW-TYPE)))
    (IF CACHED-TYPE-NAME
      (LET ((TYPE (WINDOW-TYPE-NAMED CACHED-TYPE-NAME T)))
        (IF (AND TYPE (FUNCALL (WINDOW-TYPE-RECOGNIZER TYPE)
          WINDOW))
          TYPE
          (PROGN ;; invalidate cache
            (IL:WINDOWPROP WINDOW 'WINDOW-TYPE NIL)
            ;; try again
            (WINDOW-TYPE WINDOW NO-ERROR?))))))
      (LET ((TYPE (WINDOW-TYPE-INTERNAL WINDOW NO-ERROR?)))
        ;; should cache misses here too
        (WHEN TYPE
          (IL:WINDOWPROP WINDOW 'WINDOW-TYPE (WINDOW-TYPE-NAME TYPE))
          ; cache it
          ; return it
          (TYPE))))))

```

```

(DEFUN WINDOW-TYPE-INTERNAL (WINDOW NO-ERROR?)

```

;;; We only want the most specific type -- that which no others are dependent upon. We find this by first enumerating all the types whose recognizer fires
 ;;; on WINDOW. We then delete from this list any types upon which others in the list are dependent. The remaining list should have only one element --
 ;;; the right type.

```

  (LET* ((ALL-TYPES (WITH-COLLECTION (MAPHASH #'(LAMBDA (NAME TYPE)
    (WHEN (FUNCALL (WINDOW-TYPE-RECOGNIZER TYPE)
      WINDOW)
      (COLLECT TYPE))))
    *WINDOW-TYPES*))
    (REMAINING-TYPES (COPY-LIST ALL-TYPES)))
    (DOLIST (TYPE ALL-TYPES)
      (DOLIST (DEPENDENCY (WINDOW-TYPE-DEPENDENCIES TYPE))
        (SETQ REMAINING-TYPES (DELETE (WINDOW-TYPE-NAMED DEPENDENCY)
          REMAINING-TYPES))))))
    (COND
      ((NULL REMAINING-TYPES)
        (UNLESS NO-ERROR? (ERROR "Can't find window type for ~S." WINDOW)))
      ((ENDP (REST REMAINING-TYPES))
        (FIRST REMAINING-TYPES))
      (T (UNLESS NO-ERROR?
        (ERROR "Type conflict: ~S is of types ~S." WINDOW (MAPCAR #'WINDOW-TYPE-NAME REMAINING-TYPES)
          ))))))

```

```

(DEFUN ABSTRACT-WINDOW (WINDOW &OPTIONAL SHH)

```

;;; returns an abstraction suitable for passing to RECONSTITUTE-WINDOW, or NIL if it can't find one.

```

  (LET* ((TYPE (WINDOW-TYPE WINDOW T))
    (ABTRACTER (AND TYPE (WINDOW-TYPE-ABSTRACTER TYPE))))
    (IF ABSTRACTER
      (LIST* :TYPE (WINDOW-TYPE-NAME TYPE)
        (FUNCALL ABSTRACTER WINDOW))
      (UNLESS SHH
        (FRESH-LINE *ERROR-OUTPUT*)
        (IF TYPE
          (FORMAT *ERROR-OUTPUT* "Can't abstract windows of type ~S." (WINDOW-TYPE-NAME TYPE))
          (FORMAT *ERROR-OUTPUT* "~S has no type." WINDOW))
        (LET ((HIDDEN? (WINDOW-HIDDEN? WINDOW))
          (WHEN HIDDEN? (UN-HIDE-WINDOW WINDOW))
          (IL:FLASHWINDOW (IF (SHRUNKEN? WINDOW)
            (WINDOW-ICON WINDOW)
            WINDOW)))

```

```
(WHEN HIDDEN? (HIDE-WINDOW WINDOW)))  
(FORMAT *ERROR-OUTPUT* " Ignoring.~%"  
NIL)))
```

```
(DEFUN RECONSTITUTE-WINDOW (TYPE-NAME ARG)  
  (LET ((TYPE (WINDOW-TYPE-NAMED TYPE-NAME T)))  
    (IF TYPE  
      (FUNCALL (WINDOW-TYPE-RECONSTITUTER TYPE)  
                ARG)  
      (PROG1 NIL (WARN "Can't reconstitute windows of type ~A." TYPE-NAME))))))
```

```
(SEEDIT:DEF-LIST-FORMAT DEF-WINDOW-TYPE :ARGS (NIL :KEYWORD NIL)  
:INDENT (1))
```

```
(IL:PUTPROPS IL:ROOMS-WINDOW-TYPES IL:COPYRIGHT ("Venue & Xerox Corporation" 1987 1988 1990 2020))
```

FUNCTION INDEX

ABSTRACT-WINDOW2 WINDOW-TYPE2 WINDOW-TYPE-NAMED2
RECONSTITUTE-WINDOW3 WINDOW-TYPE-INTERNAL2

SEDIT-FORMAT INDEX

DEF-WINDOW-TYPE3

DEFINER INDEX

DEF-WINDOW-TYPE1

VARIABLE INDEX

WINDOW-TYPES1

FILE-ENVIRONMENT INDEX

IL:ROOMS-WINDOW-TYPES1

DEFINE-TYPE INDEX

IL:WINDOW-TYPES1

MACRO INDEX

WINDOW-TYPE-PROP1

STRUCTURE INDEX

WINDOW-TYPE1
