"Extending the Workplace Shell with Object REXX"

Keywords: SOM, WPS, Object Rexx

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)

Vienna University of Economics and Business Administration (Wirtschaftsuniversität Wien) (http://www.wu-wien.ac.at)
IS Department (http://www.wu-wien.ac.at/wi)

University of Essen (http://www.Uni-Essen.de)
IS Department (http://nestroy.wi-inf.Uni-Essen.de)

© Rony G. Flatscher, 2000
Overview

- IBM "System Object Model" (SOM)
  - SOM 1.x, SOM 2.1, SOM 3.0
  - Brief intro and overview

- IBM "Workplace Shell" (WPS)
  - Brief intro and overview

- Object Rexx (ORX)
  - "switchRx"
    - Interface to SOM
      - Example: Querying the SOM Interface Repository
    - Interface to WPS
      - "wpsinst +"
      - Example: Creating a password protected WPS folder
      - Example: Querying the active workplace shell
Object-oriented run-time system

- OO-shaped interface for programs
  - COBOL (!)
  - C
  - C++
  - Object Rexx ...

- Instead of \( n! \) interfaces between all programming languages only \( n \) needed:
  - per language one interface to SOM

- Interface definitions
  - define methods (functions) and their signatures
  - define attributes
  - define type of arguments: IN, OUT, INOUT

- Knowledge used for shaping OMG’s CORBA standard
Object-oriented run-time system

- SOM 1.0
  - introduced with OS/2 2.0 (1992!)
  - single process
  - WPS built on it

- SOM 2.x
  - introduced with OS/2 3.0
  - multiple processes: OMG CORBA 1.1 compliant ORB
  - distributable: DSOM
  - available also for AIX, Windows/NT

- SOM 3.0
  - December 1996
  - fully CORBA 2.0 compliant (inter-ORB)
  - unsupported (!) but freely downloadable for OS/2, AIX, Windows/NT from IBM’s websites
Class hierarchy

- root: class "SOMObject"
  - e.g. methods "somInit", "somUninit", "somGetClass", "somIsA"
- metaclass: class "SOMClass"
  - e.g. methods "somNew", "somGetName", "somFindMethod"

SOM Frameworks

- Interface Repository
- Metaclass Management
- Event Management
Workplace Shell (WPS, 1)

- Object-oriented user interface
  - introduced with OS/2 2.0 (1992!)
  - extensible framework
  - built with SOM technology
    - each WPS class is in effect a SOM class
      - hence, at least all SOM methods of SOM's root class "SOMObject" are available
    - all WPS classes are retrievable via the SOM runtime, which manages all SOM classes

- Rexx-interface via Rexx Utility Functions
  - SysCreateObject, SysDestroyObject, SysSetObjectData, SysSetIconData, SysRegisterObjectClass, SysDeregisterObjectClass, SysQueryClassList, SysGetEa, SysPutEa, SysIni
Class hierarchy

- root: class "WPObject"
  - a specialization (direct subclass) of class "SOMObject"
    - all "SOMObject" methods are available to all WPS-classes via inheritance
  - e.g. methods
    - "wpOpen", "wpClose", "wpDelete", "wpCopyObject", "wpSaveState", "wpRestoreState" etc.
  - superclass of
    - "WPAbstract", "WPFileSystem", "WPTransient"
Class hierarchy (continued)

- **class "WPAbstract"**
  - *not reflected in the file system as files*
  - *data stored in INI-files*
  - *superclass of*
    - *e.g. "WPClock", "WPKeyboard", "WPMouse", "WPPalette", "WPProgram", "WPShadow", "WPSound" etc.*

- **class "WPFileSystem"**
  - *classes which are reflected in the file system as files*
  - *data stored in files*
  - *superclass of*
    - "WPDataFile" (superclass of e.g. "WPHtml", "WPPointer")
    - "WPFolder" (superclass of e.g. "WPDesktop", "WPDrives", "WPNetwork", "WPStartup", "WPTemplates")
Class hierarchy (continued)

- class "WPTransient"
  - classes which just exist during an operating system run
    - i.e. lifecycle starts with "Startup" and ends with "Shutdown"
  - storage of data usually not necessary
  - superclass of
    - "WPJob", "WPDevice" (e.g. "WPDevSerial", "WPDevAudio"),
      "WPPort", "WPPdr", "WPQdr"
Object Rexx and SOM (ORX, 1)

- Object-oriented version of the great Rexx-Interpreter
  - Introduced with Warp 4 (1996!)
  - needs to get explicitly activated
    - `switchrx.cmd`
- direct interface to SOM (and DSOM)
  - `pre-requisites`
    - documented in "somreq.doc" which can be found in the directory containing the ORX examples together with the SOM animal example, e.g.
      - built with SOMObjects 2.1 toolkit or higher (for dynamically finding infos via the Interface repository)
      - SOM class must be in a DLL (along the SOMIR-environment path variable) with a defined "SOMInitModule" routine
- `OUT` and `INOUT` arguments
  - support via the predefined Object Rexx classes
  - found in "\os2\dlfclass.cmd"
Object-oriented version of the great Rexx-Interpreter

- direct interface to SOM (and DSOM) (continued)
  - allows to use any SOM/DSOM class
  - allows to send any SOM/DSOM message
  - allows to specialize SOM/DSOM classes
  - SOM classes appear as Object Rexx classes
    - sending messages to SOM objects as simple as sending messages to ORX objects (ORX message operator: twiddle ~)
Object-Rexx example of querying the SOM Interface Repository (SIR)
- gets access to the SOM Interface Repository Framework
- queries all SOM classes available in the system
- iterates over received container, displays names of SOM classes
- frees the resources reserved by the SIR framework
/* querying the SOM interface repository with Object REXX */
aRepository = .somClassMgrObject~_get_somInterfaceRepository

SAY "repository:" pp(aRepository) "of class:" pp(aRepository~class)
SAY

aContainer = aRepository~contents("InterFaceDef", .true)
SAY "aContainer:" pp(aContainer) "items" pp(aContainer~items)

length = LENGTH(aContainer~items)
i = 0
DO anItem OVER aContainer
   i = i + 1
   SAY RIGHT(i,length) "id:" LEFT(pp(anItem~_get_id),35) "name:" pp(anItem~_get_name)
anItem~somFree
END

aRepository~somFree
exit 0

::ROUTINE pp
RETURN "[" || arg( 1 ) || "]"

/* class to get access to SOM */
::CLASS Test PUBLIC EXTERNAL 'SOM SOMObject'

© Rony G. Flatscher, 2000

(Object Rexx: Extending the WPS...), page 13
repository: [a Repository] of class: [The SOMProxy class]

aContainer: [an Array] items [423]

1 id: [::SOMObject] name: [SOMObject]
2 id: [::Sockets] name: [Sockets]
3 id: [::AnyNetSockets] name: [AnyNetSockets]
4 id: [::Contained] name: [Contained]
5 id: [::AttributeDef] name: [AttributeDef]
6 id: [::BOA] name: [BOA]
7 id: [::SOMEEvent] name: [SOMEEvent]
8 id: [::SOMEClientEvent] name: [SOMEClientEvent]
9 id: [::Context] name: [Context]
10 id: [::ConstantDef] name: [ConstantDef]
11 id: [::Container] name: [Container]
12 id: [::SOMPCodecAbstract] name: [SOMPCodecAbstract]
13 id: [::SOMPAttrEncoderDecoder] name: [SOMPAttrEncoderDecoder]

... cut ....

122 id: [::TypeDef] name: [TypeDef]
124 id: [::WPObj] name: [WPObj]
125 id: [::M_WObject] name: [M_WObject]
126 id: [::WPFileSys] name: [WPFileSys]
127 id: [::M_WPFileSys] name: [M_WPFileSys]
128 id: [::WPFold] name: [WPFold]
129 id: [::M_WPFolder] name: [M_WPFolder]
130 id: [::WDataFile] name: [WDataFile]
131 id: [::M_WDataFile] name: [M_WDataFile]
132 id: [::WPAbstract] name: [WPAbstract]
133 id: [::M_WPAbstract] name: [M_WPAbstract]

... cut ....

423 id: [::M_OverrideFlWorkerEx] name: [M_OverrideFlWorkerEx]
SOM-Animal, SOM-Dogs (1)

- contained in Object Rexx examples for SOM
  - part of the Object Rexx package downloadable from IBM for free or from DevCon ("Developer Connection")
  - Definition of SOM-classes as IDL and C-programs
    - "Animal" (superclass of "Dog")
      - methods: _get_name, _set_name, _get_sound, _set_sound, _get_genus, _get_species, talk, display
    - "Dog" (superclass of "BigDog" and "LittleDog")
      - methods: _get_breed, _set_breed, _get_color, _set_color
      - overrides: _get_genus, _get_species, display
    - "BigDog"
      - overrides method: talk
    - "LittleDog"
      - overrides method: talk
/* derived from IBM’s animal.cmd example */

spot = .dog~new
Say "spot’s default name:" spot
say "spot’s ClassName:" spot~somGetClassName
say "display"; spot~display
say "now talk, spot:"; spot~talk

sadie = .bigDog~new /* Create new Big Dog Object */
sadie~setup('Sadie', 'German Shepard', 'black and tan', 'Steve')
say "sadie’s default name:" sadie
say "sadie’s ClassName:" sadie~somGetClassName
say "display:"; sadie~display
say "now talk, sadie:"; sadie~talk

/* import some SOM Classes to use */
::Class Dog Public EXTERNAL 'SOM Dog'
::Class BigDog Public EXTERNAL 'SOM BigDog'
::method setup /* setup object */
  expose owner
  use arg name, breed, color, owner /* Owner assign on use Arg.... */
  self~_set_name(name) /* Set the SOM attribute */
  self~_set_breed(breed)
  self~_set_color(color)
  self~objectName = name /* set up the object’s name to be the name as well */
::method display /* display attribute values */
  expose owner
  say ‘The Big ‟ ‟ Dog ‟ is owned by ‟ ’
spot’s default name: a Dog
spot’s ClassName: Dog
display

The animal named (Genus: Canis, Species: Familiaris) says:
   <Unknown>
It’s breed is and its color is .
now talk, spot:
   <Unknown>

sadie’s default name: Sadie
sadie’s ClassName: BigDog
display:
The Big <black and tan> Dog <Sadie> is owned by <Steve>
now talk, sadie:
   WOOF WOOF
   WOOF WOOF
   WOOF WOOF
   WOOF WOOF
Direct WPS-support

- "wpsinst +"
  - faster
  - "wpuser.cmd"
    - serves as "startup.cmd" for starting WPS
      - called by the direct Object Rexx WPS-support
      - e.g. defining WPS-specializations in Object Rexx and making them available each time the WPS starts up

- support definitions
  - "\os2\wpsysobj.cmd"
    - defines access to most used WPS-classes by placing them into Object Rexx' global environment ".environment"
      - accessible as environment symbol
Direct WPS-support

- support definitions (continued)
  - "\os2\wpconst.cmd"
    - defines most important WPS constants and stores them in the directory "wpconst"
    - stored in the global environment
    - accessible as the environment symbol ".wpconst"
  - "\os2\wpfind.cmd"
    - finds WPS-object by the given name
    - can be called from the command line or from within an Object Rexx program
    - demonstrates usage of WPS-methods from Object Rexx
Object Rexx and WPS (3)  
Password Protected Folder

- Choose class to specialize
  - "WPFolder"
- Choose methods to override
  - "wpSetup"
  - "wpSaveState"
  - "wpRestoreState"
  - "wpOpen"
  - "wpClsQueryTitle"

- use IBM’s "VREXX.ZIP" (author: Steve Lamb) for GUI-interface
- require this class in "wpuser.cmd"
   ---rgf, wuw; (using VREXX.ZIP and changing from WPDLF to DLF-data type classes)
   using VREXX.ZIP, ews from Steve Lamb (IBM)
*/

call RxFuncAdd 'VInit', 'VREXX', 'VINIT'

if Vinit() = "ERROR" then /* error loading VRexx-functions */
do
  call VExit /* clean-up */
  raise syntax 40.1 array ("VREXX.Vinit()") /* abort program */
end
.local~lock_icon = STREAM("REXX.ICO", "C", "QUERY EXISTS")
.environment~WPLockFolder = .WPLockFolder /* make class available */

::REQUIRES DLFClass /* needs the support for INOUT/OUT datatypes */
::CLASS VXPWPrompts mixinclass object
::METHOD ask4Password /* ask for a password */
    use arg title, prompt
    buttons = 3 /* use "OK"- and "CANCEL"-buttons */
    prompt.0 = 1; /* prompt */
    if arg(2, "E") then prompt.1 = prompt
        else prompt.1 = 'Password'
    width.0 = 1; width.1 = 64 /* widths in character units */
    hide.0 = 1; hide.1 = .true /* don't echo PW */
    answer.0 = 1; answer.1 = '' /* default value: empty string */
    call VDialogPos 50, 50 /* center message box on screen */
    button = VMultBox(title, "prompt", "width", "hide", "answer", buttons)

    if button = 'OK' then return answer.1 /* return entered password */
    return .nil /* "CANCEL" pressed; indicate no PW entered */

::METHOD displayError
    use arg msg
    do i=1 to 10 while msg <> ""
        pos = length(msg)
        if pos > 80 then /* VRexx allows 80 chars per msg-line only */
            do
                pos = lastpos(" ", msg, 80) /* try to break at a blank */
                if pos = 0 then pos = 80 /* no blank in first 80 chars, force break */
            end
        msg.i = substr(msg, 1, pos) /* assign chunk to msg-stem */
        msg = substr(msg, pos+1)
    end
    msg.0 = i /* assign message */
    call VDialogPos 50, 50 /* center message box on screen */
    return VMsgBox('Important error message!', "msg", 1) /* show OK-button only */
::CLASS SMPPWChange SUBCLASS WPAbstract
::METHOD wpOpen
    use arg handleContainer, view, params
    if view \= 2 & view \= 3 then Do /* Opening Default view? Dbl-click */
        lockf = self~wpQueryFolder /* Get our containing lock folder */
        /* Ask for new password */
        newpw = lockf~ask4Password(’New LockFolder Password’, ’Enter New Password’)
        if newpw \= .nil Then Do /* Get a new password? */
            lockf~password = newpw /* Yup, set new pw. */
            lockF~wpSaveImmediate /* Save object state (PW) */
        End
    End
    return 0
End
/* Forward wpOpen to super class to handle. */
forward class (super)
::CLASS WPLockFolder SUBCLASS WPFolder INHERIT VXPWPrompts
::METHOD wpclsQueryTitle CLASS
   return 'LockFolder'

::METHOD init
   expose password
   self~init:super /* let superclass initialize it */
   /* Create object to allow PW Change */
   .smpPwChange~new('Change Password', 'ICONFILE=' || . lock_icon || ';' , self , 1)
   if var('PASSWORD') Then /* PW initialized via SetupString? */
      password = '' /* Nope, give default '' */

::METHOD wpOpen
   expose password
   use arg handleContainer, view, params
   if password == '' then /* no password set? */
      return self~wpOpen:super(handleContainer, view, params) /* go ahead and open this*/
      /* Ask user for password. */
      enterpw = self~ask4Password('Locked Folder Password', 'Enter Password')
      if password = enterPw then Do /* Was correct password entered */
         /*Yup, forward to WPFolder top Open */
         return self~wpOpen:super(handleContainer, view, params)
      End
      /*Incorrect pw entered. */
      reply .false /* Return failure, and return to WPS */
      guard off /* Now display error to user. */
      self~displayError('LockFolder Error! [should be: "" || password || "]")
   End

© Rony G. Flatscher, 2000 (Object Rexx: Extending the WPS ...), page 24
::METHOD wpSetup
    use arg setupString
    maxLength = 64
    strLength = .DLFULong~new(maxLength) /* Will allow for up to 64 char PW */
    /* Get INOUT String parm */
    str = .DLFString~new~~_set_maxSize(maxLength)

    /* see if setup string has PW */
    if self~wpScanSetupString(setupString, 'PASSWORD', str, strLength) then
        self~password = str~asString /* Yup, set password. */
    return self~wpSetup:super(setupString) /* Superclass does remainder. */

::METHOD scrollTitle unguarded /* unguarded, want to run concurrently*/
    title = self~wpQueryTitle /* Get current title */
    do 2 /* Will scroll twice. */
       do i = 1 to title~length /* For length of title. */
          self~wpSetTitle(right(left(title, i), title~length)) /* display 1st 1 chars of title */
       end
    end
::METHOD password ATTRIBUTE
::METHOD wpSaveState /* Save the password data */
self~wpSaveString(self~somGetClassName, 1, self~password)
return self~wpSaveState:super /* Let parent save any state. */
::METHOD wpRestoreState
self~initButtons /* make sure OREXX side initialized. */
size = .DLFULong~new /* Get DLFUlong for size query. */
/* Retrive size of string for restore */
self~wpRestoreString(self~somGetClassName, 1, .nil, size)
/* Create DLFString large enough to contain the string, plus NULL */
str = .DLFString~new~_set_maxSize(size~_get_value + 1)
/* Now get saved password */
self~wpRestoreString(self~somGetClassName, 1, str, size)
self~password = str~asString /* Save password state value. */
/* let parent restore state. */
return self~wpRestoreState:super(arg(1))