"Object Rexx and Windows Automation Interfaces"

2004 International Rexx Symposium
Sindelfingen/Böblingen, Germany (May 2004)

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu-wien.ac.at)
Agenda

- COM, OLE, ActiveX, ActiveScript
  - Basic architecture
- Object Rexx class ".OLEObject"
  - Some Object Rexx examples
- Problem statement
- Package "RGF_OLEINFO"
  - Description of utilities and examples
- Roundup
• COM
  – Component Object Model
    • RPC ("remote procedure call")
    • Interfaces (e.g. "IUnknown")
    • Further developments
      – DCOM, COM+

• OLE
  – Object Linking and Embedding
    • COM-based
    • Linking of documents (Dynamic Data Exchange)
      – Cold link
      – Warm link
      – Hot link
    • Embedding of "alien" documents
• VBX, OCX, ActiveX
  – Set of COM interfaces defining Windows "Components"
    • Windows programs, which can be combined
    • Pre-defined interface for interfacing with components
  – Acronyms
    • Visual Basic Extension (VBX)
      – Mostly for GUI
    • Object Component Extension (OCX) and ActiveX
      – Independent of Visual Basic, hence deployable by all Windows programs
• OLE (ActiveX) Automation
  – Interface for addressing and remote controlling Windows applications/components
  – Set of COM based interfaces
    • Standardized definition of APIs for (scripting) languages
      – Invoking functions of Windows programs
      – Querying and setting values of attributes in a Windows component
      – Intercepting of events, which occur in Windows components
    • Recording of user-actions, which later can be reproduced with the help of a scripting language ("macros")
OLE, ActiveX

Applications/components are registered with the "Windows registry"

- HKEY_CLASSES_ROOT
  - CLSID
    - GUID resp. UUID
    - Global resp. Universal Unique Identifier
  - ProgID
    - Easier to understand/memorize for humans, a unique string
    - VersionindependentProgID

Addressing of such registered components

- CLSID, PROGID or via a "Moniker" (a string)
Object Rexx Class "OLEObject", 1

- "Proxy" class for interfacing with OLE- resp. ActiveX- Windows programs, enables
  - Finding and addressing of running OLE/ActiveX components
  - Creating new instances of OLE/ActiveX components
  - Querying of the published APIs, attributes, constants and events
- Addressing (invoking) of the published APIs by means of plain Object Rexx messages!
  - Arguments are automatically converted from/to Object Rexx
  - Return values are automatically converted to Object Rexx
Object Rexx Class "OLEObject", 2

- Converting between the following data types
  - VARIANT, VT_EMPTY, VT_NULL, VT_VOID, VT_I1, VT_I2, VT_I4, VT_I8, VT_UI1, VT_UI2, VT_UI4, VT_UI8, VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, VT_DISPATCH, VT_VARIANT, VT_PTR, VTSAFEARRAY

- Querying/setting of attribute values
  - As if Object Rexx attributes

- Intercepting Windows component events in Object Rexx
• Methods of ".OLEObject"
  – `Init( ProgID | CLSID [, NoEvents|WithEvents] )`
    • Creates and returns a new instance of a Windows component
  – `GetObject( Moniker [, SubclassOfOLEObject] )`
    • Returns an existing instance of a Windows component
  – `GetConstant( [ConstantName] )`
    • Returns the value of a constant with the name `ConstantName`, or
    • Returns all published constants in a Rexx stem
  – `GetKnownEvents, GetKnownMethods`
    • Returns a stem with all published events or methods
  – `GetOutParameters`
    • Returns an array object with the values for the "Out"-only arguments of the last message sent to the Windows component
Object Rexx Class "OLEObject", 4

- `UNKNOWN(MessageName [, ArrayOfArguments] )`
  - This method forwards all messages unknown to Object Rexx to the Windows component, hence
  - Be careful about message-names which exist in the Object Rexx classes `OLEObject` or its superclass `Object`
    - If an Object Rexx method is found, it gets invoked
    - Problem mostly for the message names "COPY" and "CLASS", which sometimes are defined in Windows components, but exist in the Object Rexx root class `Object`
    - Sending a message directly to the Windows component is possible, by directly using the message name "UNKNOWN", e.g.

```
proxy~UNKNOWN("COPY")
```
Some Examples, Hints

- The Windows version of Object Rexx is distributed with numerous OLE/ActiveX Examples

  ?\Programs\ObjRexx\SAMPLES\OLE

- Some examples from IBM
  - Remote controlling MS Internet Explorer
  - Remote controlling MS Excel
  - Interfacing with the "advanced directory services" (ADS)
/* create an object for IE */
myIE = .OLEObject~New("InternetExplorer.Application")
myIE~Width = 800
myIE~Height = 256

Say "Current dimensions of IE are:" myIE~Width "by" myIE~Height

/* set new dimensions and browse IBM homepage */
myIE~Width = 800
myIE~Height = 600
myIE~Visible = .True
myIE~Navigate("http://www.ibm.com")

/* wait for 10 seconds */
Call SysSleep 10

myIE~Navigate("http://www.ibm.com/news")

/* wait for 10 seconds */
Call SysSleep 10
myIE~quit

::REQUIRES "OREXXOLE.CLS"
```plaintext
excelObject = .OLEObject~new("Excel.Application")
Worksheet = excelObject~Workbooks~Add~Worksheets[1]
myTitles="ABCDEFGHI"

    do j = 1 to 10
        do i = 1 to myTitles~length
            title = myTitles~substr(i,1)
            cell = Worksheet~Range(title||j) -- e.g. ~Range("A1")
            if j = 1 then do
                cell~value = "Type" title -- header of first row
                cell~font~bold = .true
            end
            else if j = 10 then do -- final row? yes, build sums
                /* set formula, e.g. "=sum(B2:B9)" */
                cell~formula = "=sum(?2:?9)"~translate(title,"?")
                cell~Interior~ColorIndex = 24 -- light blue
            end
            else -- a row between 2 and 9: fill with random value
                cell~value = random()
            end
        end
    end

    /* save sheet in default TEMP directory */
    Worksheet~SaveAs( value("TEMP",,ENVIRONMENT)"\demo.xls")
excelObject~Quit
exit
::requires "orexxole.cls"
```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")
myComputer = .OLEObject~GetObject("WinNT://"||ComputerName||",computer")

say "Standard properties of this computer:"
say left("Name:",10," ") myComputer~name

/* in this case, using myComputer~class would invoke the standard REXX */
/* method "Class", therefore the OLE objects' "class" method has to be */
/* called explicitly using the "Unknown" method (see documentation for */
/* details on this mechanism). */
say left("Class:",10," ") myComputer~unknown("class",.nil)

say left("GUID:", 10, " ") myComputer~guid
say left("ADsPath:",10, " ") myComputer~adspath
say left("Parent:", 10, " ") myComputer~parent
say left("Schema:", 10, " ") myComputer~schema

exit::requires "OREXXOLE.CLS"
Some Problems

• Great functionality
  – Interfacing and remote controlling Windows components as if they were Object Rexx objects
  – Object Rexx can replace Visual Basic

• Problems
  – Documentation of the APIs
    • Sometimes not installed
    • APIs not documented in online help
    • Documentation chaotically organized
    • Documentation usually only refers to symbolic names of constants, not their values!
  – Unknown set of installed Windows components
Package "RGF_OLEINFO", 1

• RGF_OLEINFO
  – Set of Object Rexx utilities for exploring and documenting OLE/ActiveX components
  – Utility to create Object Rexx include files for OLE/ActiveX constants (rgf_oleconstants.rex)
  – HTA-application written in Object Rexx serving as GUI and as a rendering processor (rgf_oleinfo.hta using rgf_oleinfo.rex)
    • Analyzes and lists all registered OLE/ActiveX components
    • Analyzes and renders interfaces of OLE/ActiveX components the users selects
      – Resulting HTML files can be saved
Package "RGF_OLEINFO", 2

- Object Rexx program to allow Object Rexx programs to analyze OLE objects at runtime
  - Makes it e.g. possible to analyze the interface of OLE/ActiveX components which are created and returned from another OLE/ActiveX component

- WSC ("Windows Script Component", rgf_oleinfo.wsc)
  - Implemented in Object Rexx
  - Allows any OLE/ActiveX application to use the Object Rexx analyze and rendering mechanism
  - Only available method "analyze"
• Queries predefined constants

• Creates Object Rexx code to save all constants in the local environment using the directory object ".ole.const"
  – Easy to refer to OLE/ActiveX constants from Object Rexx
    .ole.const~csc_navigateBack

• Usage from the command line
  
  rgf_olenconstants progid/clsid [outfile]

• Example
  
  rgf_olenconstants InternetExplorer.Application iec.rex
Content of "iec.rex" (excerpt)


-- OLE/ActiveX-application/clsid: [InternetExplorer.Application] - there is/are [84] constants

-- create directory 'ole.const', if necessary; maybe shared with OLE constant definitions of other programs
if .local~hasentry('ole.const')=.false then .local~ole.const=.directory~new -- create directory 'ole.const' in .local

.ole.const~CSC_NAVIGATEBACK = 2
.ole.const~CSC_NAVIGATEFORWARD = 1
.ole.const~CSC_UPDATECOMMANDS = -1
.ole.const~OLECMDEXECOPT_DODEFAULT = 0
.ole.const~OLECMDEXECOPT_DONTPROMPTUSER = 2
.ole.const~OLECMDEXECOPT_PROMPTUSER = 1
.ole.const~OLECMDEXECOPT_SHOWHELP = 3
.ole.const~OLECMDF_DEFHIDEONCTXMENU = 32
.ole.const~OLECMDF_ENABLED = 2
.ole.const~OLECMDF_INVISIBLE = 16
.ole.const~OLECMDF_LATCHED = 4
.ole.const~OLECMDF_NINCHED = 8
.ole.const~OLECMDF_SUPPORTED = 1
.ole.const~OLECMDID_ALLOWUILESSSAVEAS = 46
.ole.const~OLECMDID_CLEARSELECTION = 18
.ole.const~OLECMDID_CLOSE = 45
.ole.const~OLECMDID_COPY = 12
... cut ...

"rgf_oleinfo.rex", 1

- Queries all available OLE/ActiveX information
  - Information about implementation of OLE/ActiveX components
    - Description, CLSID, ProgID and VersionIndependentProgId (if any)
    - Date when OLE/ActiveX component got registered with Windows
    - DLL/EXE which implements the component, its date and size
  - APIs, attributes, events, constants

- Renders results in HTML

- Rendering may occur in one of two modes
  - Normal mode
    - separate listing of APIs ("methods"), Read-only attributes ("properties"), Write-only attributes ("properties"), Read/Write attributes ("properties"), Events, Constants
  - Compact mode
    - All attributes (properties) are folded together
    - No constants
• Usage
  – Command line
    \texttt{rgf\_oleinfo progid/clsid [mode [display]]}
  – From Object Rexx as a function
    \texttt{res=rgf\_oleinfo id | oleobj [, [header] [, [mode] [, display] ] ]}
  – Where
    • \texttt{progid/clsid} or \texttt{oleobj}
      – PROGID/CLSID of OLE/ActiveX component, or any OLEobject
    • \texttt{header}
      – Optional: HTML header (displayed in title of browser)
    • \texttt{mode}
      – Optional: 0=normal, 1=compact, default
    • \texttt{display}
      – Optional: 0=no display, 1=display with Internet Explorer, default
"rgf_oleinfo.hta"

- Web-Browser frontend for users
  - Analyzes registry for OLE/ActiveX components
  - Allows selection of OLE/ActiveX components to be analyzed
  - HTML with embedded Object Rexx code, which in turn employs "rgf_oleinfo.rex"

- ".hta"
  - Hypertext Application
    - HTML with embedding code, e.g. Object Rexx code
    - Like an EXE-program!
"rgf_oleinfo.wsc", 1

- "rgf_oleinfo.wsc"
  - An OLE/ActiveX component which is **implemented in Object Rexx (!!)**
  - Allows C++, VisualBasic, VBScript, JScript etc. to use "rgf_oleinfo.rex"
  - Employs the Windows script shell functionality
    - Needs to get registered
      - Right click in Explorer, choose "Register"
Defines the OLE/ActiveX component named "REXX.OLEinfo", which has one method with the following signature:

```plaintext
```

- Invokes "rgf_oleinfo.rex" with the supplied arguments:
  - Cf. description of arguments in the appropriate section about tat utility above
– JScript (JavaScript) Example

```javascript
// JScript
var mxVar, myVar
mxVar = new ActiveXObject("Rexx.OLEinfo")
myVar = new ActiveXObject("InternetExplorer.Application")

WScript.echo( "about to use 'Rexx.OLEinfo.analyze()...'")
mxVar.analyze(myVar, "invoked via JScript !")
WScript.echo( "done. (js)"
```
- VBScript (Visual Basic) Example

' VBScript
dim mxVar, myVar
Set mxVar = createObject("Rexx.OLEinfo")
Set myVar = createObject("InternetExplorer.Application")

WScript.echo "about to use 'Rexx.OLEinfo.analyze()...'
' OLEobject *must* be enclosed in parenthesis, otherwise
' the default string value is retrieved!
mxVar.analyze ( myVar )
WScript.echo "done. (vbs)"
Roundup

- **Object Rexx for Windows**
  - Implemented as a ActiveScript engine
  - Can be used wherever VBScript, JScript etc. are used
- "OLEObject" serves as proxy class
  - Takes over the communication between Object Rexx and the OLE/ActiveX components
- "RGF_OLEINFO"
  - A package for analyzing and documenting OLE/ActiveX interfaces in HTML
  - Allows C++, VBScript, Visual Basic, JScript etc. to take advantage of the Object Rexx solution!