

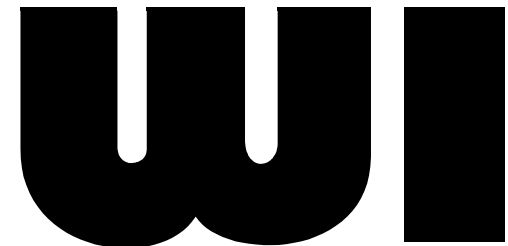
"Applying the Object REXX Windows Scripting Engine with Windows Scripting Host "

13th International Rexx Symposium, Raleigh, North Carolina
April 28th - May 1st, 2002

Rony G. Flatscher


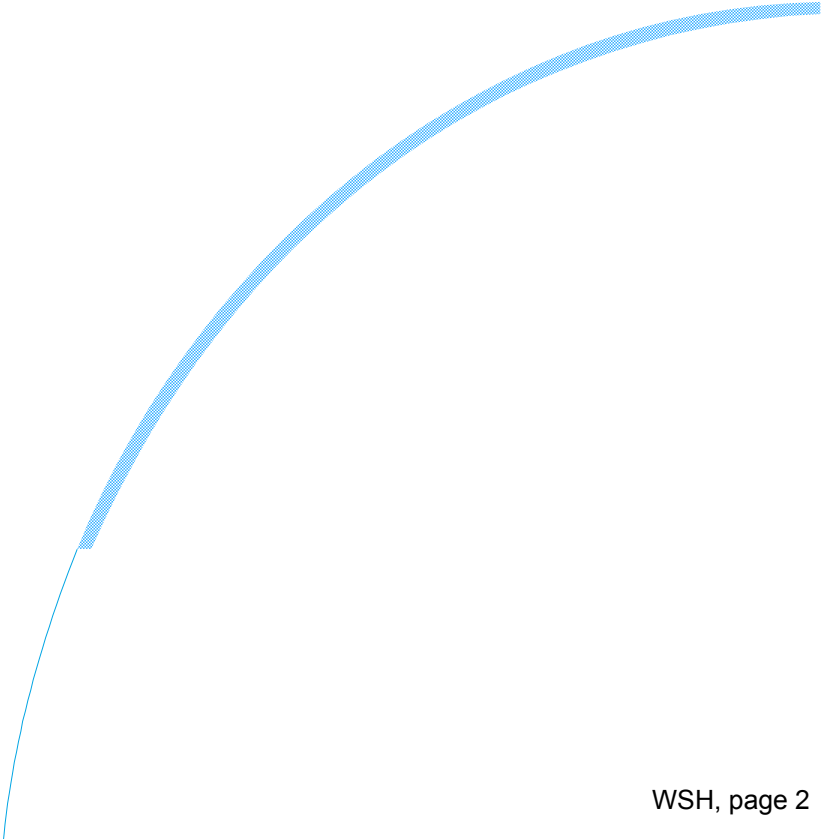
Vienna University of Economics and Business Administration (WU)

Abteilung für
Wirtschaftsinformatik





Agenda

- 
- Terms
 - Building blocks
 - WSH and WSE
 - Examples
 - Roundup
- 

Terms (1)

■ Script

– A sequence of recurrent commands directed at application programs or shells

- *usually maintained in a file*

- *used to remote-control and/or automate applications*

– Scripting languages

- *Usually programming languages with a limited set of functionality and/or easy syntax*

- *BASIC dialects like VBScript, LotusScript*

- *DOS batch language*


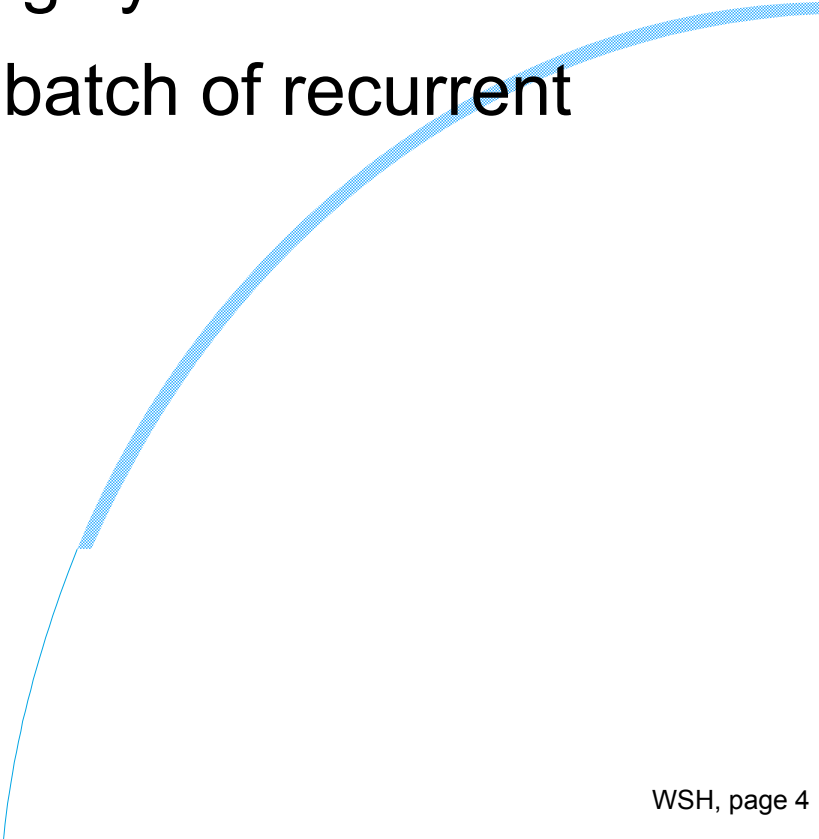
- *REXX*


- *Unix-shell based languages like Perl, Tcl*

- *WWW-browser based JavaScript a.k.a. JScript*



Terms (2)

- 
- Macro
 - A script driving an end-user application
 - Batch
 - A script driving the operating system
 - Derived from the notion "a batch of recurrent commands"
- 



Building Blocks

OLE



OLE

- Object Linking and Embedding
- Windows based
- Specific set of "COM" interfaces
 - *Component Object Model*
 - *Defines Interfaces*
 - *Object based (not object-oriented!)*
 - *RPC invocation*
 - *DCOM*
 - *"distributed COM"*
 - *COM+*
- "ActiveX": practically a synonym for "OLE"



Building Blocks

OLE Automation

- OLE automation a.k.a. ActiveX automation
 - Name for a set of COM interfaces aimed at being used by scripting languages
 - Allows for runtime interrogation of
 - *Available functions ("methods")*
 - *Available data items ("attributes")*
 - *Available (interceptable) events*
 - Allows for explicitly recording automatable application specific commands
 - Security considerations
 - *Script/macro runs in the context of the application*
 - *Uncontrollable execution !*

OLE/ActiveX Automation Example

■ Remote control MS Internet Explorer

```
-- start MSIE (create an MSIE instance)
myIE=.OLEObject~new("InternetExplorer.Application")

-- make MSIE visible by setting an MSIE attribute
myIE~visible=.true

-- navigate to given URL by invoking a MSIE method
myIE~navigate("http://www.wu-wien.ac.at")
```



Overview

Windows Scripting Engine (WSE)

- A programming language supporting the COM interfaces
 - OLE/ActiveX automation and at least:
 - IActiveScript, IActiveScriptParse, IPersist
- Is able to make "objects" from applications available in the runtime environment of the scripting language
- "Native" Windows Scripting Engines
 - VBScript (Visual Basic Scripting edition)
 - JScript (Microsoft's JavaScript/ECMAScript)



Overview

Windows Scripting Host (WSH)

- Windows Scripting Host (WSH)
 - Any application employing the IActiveScript family of interfaces
 - Being able to add initialized "objects" to the runtime environment of the WSE
- Microsoft Windows Scripting Hosts
 - Internet Explorer (MSIE)
 - Internet Authoring Tools
 - *Active Server Pages (ASP)*
 - Shell
 - *New synonym: "Windows Scripting Host" !!!*



Overview

Windows Scripting Files (WSF)

- Collection ("package") of functions ("methods")
 - Can be written in any WSE
 - WSE languages can be intermixed
- Container of scripts structured with XML markup
 - `<package>...`
 - `<job>...`
 - `<script>...`



Overview

Windows Scripting Components (WSC)

- Set of functions, attributes and events implemented in any WSE language
 - Languages can be even intermixed!
- Wrapped up in form of an XML file
- Made available as COM object
 - All programs are able to use such WSCs
 - With "Shell" WSCs are made available even via DCOM
 - *Truly distributable !*



WSH Examples: MSIE DOM/DHTML

- WWW browser parses files marked up with
 - HTML
 - XML
- Allows full interaction to any WSE via WSH
 - DOM objects
 - *Window object*
 - *Document object with all its nodes*
 - DOM Events, e.g.
 - *Keyboard, Mouse, Session infos*
- Microsoft's name for DOM: DHTML
 - "Dynamic HTML"

WSH Example: MSIE Using the Rexx WSE

```
<head>
  <title>Demonstrating the REXX Windows
    Scripting Engine (WSE)...</title>
</head>
<body>
  <script language="Object Rexx">
    document~writeln( "Greetings from REXX!" )
  </script>
</body>
```



WSH Example: MSIE

Using the VBScript WSE

```
<head>
  <title>Demonstrating the VBScript Windows
    Scripting Engine (WSE)...</title>
</head>
<body>
  <script language="VBScript">
    document.writeln "Greetings from VBScript!"
  </script>
</body>
```



WSH Example: MSIE Using the JScript WSE

```
<head>
  <title>Demonstrating the JScript Windows
    Scripting Engine (WSE)...</title>
</head>
<body>
  <script language="JScript">
    document.writeln( "Greetings from JScript!" )
  </script>
</body>
```

WSH Examples: "Shell"

- Pre-installed since
 - *Windows 98 (DOS-based Windows)*
 - *Windows 2000 (32-Bit-Kernel based Windows)*
- Allows interaction with the Windows shell
 - "WScript" object
 - *Input, output, parsing of arguments*
 - *Maintenance of the desktop, registry, installation of applications, network setup, ...*
- Supplies two additional helpful objects
 - "Scripting.Directory"
 - "Scripting.FileSystemObject" (FSO)

WSH Examples: "Shell"

■ Querying miscellaneous information

```
-- Rexx using the "Shell" WSH  
wsn = .OLEObject~new("WScript.Network")  
wscript~echo( "ComputerName:" wsn~ComputerName )  
wscript~echo( "UserName:" wsn~UserName )  
wscript~echo( "UserDomain:" wsn~UserDomain )
```

WSH Examples

Defining a WSC "Rexx.Counter" (1)

```
<?xml version="1.0"?>
<component>
  <?component error="true" debug="true"?>
  <registration
    description="Counter"
    progid="Rexx.Counter"
    version="1.00"
    classid="{cfe63bb0-391f-11d6-a3d7-006094eb4d95}"
  />
  <public>
    <property name="counter">
      <get/>
    </property>
    <method name="increment" />
  </public>
```

WSH Examples

Defining a WSC "Rexx.Counter"

```
<script language="Object Rexx">
  <![CDATA[
    .local~counter=100    -- initialize .counter to "100"

    ::routine increment public  -- increment counter
      .local~counter=.counter+1  -- increment counter
      return .counter           -- return value

    ::routine get_counter public -- accessor function
      return .counter           -- return value

  ]]>
</script>
</component>
```

WSH Examples

Using the WSC "Rexx.Counter"

```
' VBScript
dim MyVar
Set MyVar = createObject("Rexx.Counter")
wscript.echo "Counter: " & MyVar.counter
wscript.echo "Counter: " & MyVar.increment
.....
// JScript
var MyVar
MyVar = new ActiveXObject("Rexx.Counter")
WScript.echo( "Counter: " + MyVar.counter )
WScript.echo( "Counter: " + MyVar.increment() )
.....
-- REXX
MyVar = .OLEObject~new("Rexx.Counter")
wscript~echo( "Counter: " MyVar~counter )
wscript~echo( "Counter: " MyVar~increment )
```



Security (1)

- Scripts execute in the context of the application that started them
 - Access
 - *Local (standalone PC?)*
 - *Via network*
 - Spying
 - Changing/destroying content
 - Creating viruses
 - *"Love Letter Virus" and MS Outlook*
 - All functionality of the host application available to the scripts!

Security (2)

- Only as late as October 2001 some security management made available!
 - WSH 5.6
 - *Signing of scripts*
 - *Applying concept of "trust"*
 - *Possibility of defining execution rights based on trust*
- Timid security
- Still no "sandbox" by WSH foreseeable!
 - Use security managers of scripting languages if available
 - *e.g. IBM's Object Rexx*



Roundup

- OLE/ActiveX automation
- WSE
 - VBScript
 - JScript
- WSH
 - MSIE
 - ASP
 - Shell a.k.a. "WSH"
- WSC
 - D/COM