Institut für Betriebswirtschaftslehre und Wirtschaftsinformatik

Automatisierung von Windows Anwendungen (7)

OLE-Automation/ActiveX-Automation, die Object Rexx Proxy Klasse "OLEObject", Beispiele, Microsoft Internet Explorer's DHTML

Prof. Dr. Rony G. Flatscher

Wirtschaftsuniversität Wien ■ Augasse 2-6 ■ A-1090 Wien

OLE (ActiveX) Automation, 1

- COM
  - Component Object Model
    - RPC ("remote procedure call")
    - Interfaces (Schnittstellen, z.B. "IUnknown")
    - Weiterentwicklungen
      - DCOM, COM+
- OLE
  - Object Linking and Embedding
    - COM-basiert
    - Verknüpfen von Dokumenten (Dynamischer Datenaustausch)
      - Kalt (cold)
      - Warm (warm)
      - Heiß (hot)
    - Einbetten von fremden Dokumenten

OLE (ActiveX) Automation, 2

- VBX, OCX, ActiveX
  - Satz von COM-Interfaces zur Definition von Windows "Komponenten"
    - Windows Programme, die als Bausteine kombinierbar sind
    - Definierte Schnittstelle zur Kommunikation mit Komponenten
  - Akronyme
    - Visual Basic Extension
      - Hauptsächlich für GUI
    - Object Component Extension und ActiveX
      - Visual Basic unabhängig, daher für alle Windows Programme einsetzbar

OLE (ActiveX) Automation, 3

- OLE (ActiveX) Automation
  - Schnittstelle zur An- und Fernsteuerung von Windows Anwendungen/Komponenten
  - Satz von COM-basierten Interfaces
    - Standardisierte Definition von Programmierschnittstellen für (Skript-) Sprachen
      - Aufruf von Funktionen im Windows Programm
      - Abfragen und Setzen von Werten in Variablen ("Attributen") eines Windows Programmes
      - Abfangen von Ereignissen, die im Windwos Programm entstehen
    - Protokollierung von Aktionen, die später mit Hilfe einer Skriptsprache wiederholt werden können ("Makros")

OLE (ActiveX) Automation, 4

- OLE, ActiveX
  - Anwendungen und Komponenten werden in der Windows Registry eingetragen
    - HKEY_CLASSES_ROOT
      - CLSID
        - GUID bzw. UUID
        - Global bzw. Universal Unique Identifier
      - ProgID
        - Für Menschen leichter verständliche (eindeutige) Zeichenkette
        - VersionindependentProgID
  - Adressierung erfolgt entweder über CLSID, PROGID oder über einen "Moniker"/"Spitznamen" (eine Zeichenkette)

Die Object Rexx Klasse ".OLEObject", 1

- "Proxy" Klasse für die Ansteuerung von OLE- bzw. ActiveX-Windows Programmen, erlaubt das
  - Auffinden und Ansteuern von laufenden OLE/ActiveX Programmen
  - Anlegen neuer Instanzen von OLE/ActiveX Programmen
  - Abfragen der publizierten Programmschnittstellen, Eigenschaften (Attribute), Konstanten und Ereignisse, auf die reagiert werden kann
  - Ansteuern (Aufrufen) der publizierten Programmschnittstellen, über das Abschicken von entsprechenden Object Rexx Nachrichten
    - Argumente werden automatisch von Object Rexx konvertiert
    - Rückgabewerte werden automatisch von Object Rexx konvertiert

Die Object Rexx Klasse ".OLEObject", 2

- das Abfragen/Setzen von Attributwerten, über entsprechende Object Rexx Nachrichten
  - Vgl. Direktive für Attributmethoden, die eine Abfrage- und Zuweisungsmethode (mit nachfolgendem Gleichheitszeichen) generiert
- das Reagieren auf das Eintreten von Ereignissen, indem auf der Object Rexx Seite entsprechende Methoden dafür definiert werden
- Führt eine automatische Datentyp-Umwandlung durch, von bzw. zu
  - 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**, VT_SAFEARRAY

Die Object Rexx Klasse ".OLEObject", 3

- Methoden
  - Init(ProgID | CLSID [, NOEVENTS|WITHEVENTS])
    - Legt eine neue Instanz des OLE/ActiveX Programms an
    - Liefert ein Object Rexx (Proxy) Objekt dafür zurück
  - GetObject(Moniker [, SubklasseVonOLEObject])
    - Sucht eine bestehende Instanz oder legt eine neue an
    - Liefert ein Object Rexx (Proxy) Objekt dafür zurück
  - GetConstant([ConstantName])
    - Liefert Wert, der mit ConstantName vordefiniert wurde, oder
    - Liefert *alle* publizierten Konstanten mit ihren Werten als Stem

Die Object Rexx Klasse ".OLEObject", 4

- Methoden
  - GetKnownEvents, GetKnownMethods
    - Liefert einen Stem (Stammvariable), mit allen publizierten Ereignissen bzw. Methoden (= "Programmschnittstellen" bzw. "Funktionen") des OLE/ActiveX-Programmes
  - GetOutParameters
    - Liefert ein Array-Objekt mit den "Out"-Parametern des letzten Aufrufes
  - Dispatch(Nachrichtenname [, Argument1, Argument2, …])
    - Führt auf der Windows-Seite die Methode aus, die so heißt, wie "Nachrichtenname" und übergibt allfällige übergebene Argumente

Die Object Rexx Klasse ".OLEObject", 5

- Methoden
  - UNKNOWN(Nachrichtenname [, ArrayFürArgumente])
    - Diese Methode leitet alle unbekannten Nachrichten an das OLE/ActiveX-Programm weiter, das durch das Object Rexx Proxy Objekt repräsentiert wird
      - Achtung auf Nachrichten, die so heißen wie die in den Object Rexx Klassen **OBJECT** bzw. **OLEOBJECT** ! Derartige Nachrichten werden auf der Object Rexx Seite abgearbeitet
      - Problem hauptsächlich bei der Nachricht "COPY" oder "CLASS", die in der Object Rexx Klasse OBJECT existiert, daher direkter Aufruf über das Proxy-Objekt notwendig, z.B.

```
proxy~UNKNOWN("COPY")
proxy~DISPATCH("COPY") /* seit ooRexx 3.1 */
```

Beispiele, Hinweise

- Object Rexx wird mit zahlreichen OLE/ActiveX Beispielen ausgeliefert, die in folgendem Verzeichnis zu finden sind, z.B. auf einem deutschen Windows meistens unter:

```
?\Programme\ObjRexx\SAMPLES\OLE
```

- Auf den folgenden Folien werden diese Beispiele der Entwickler von Object Rexx dargestellt und erläutert !
- Achtung!
  - Es werden hier nicht alle mitgelieferten Beispiele dargestellt, bitte studieren Sie die Unterverzeichnisse und probieren Sie die Programme einfach aus (sie verändern nichts dauerhaft auf Ihrem Computer) !

...\OLE\APPS\SAMP01.REX

```
/* create an object for IE */
myIE = .OLEObject~New("InternetExplorer.Application")

myIE~Width = 454
myIE~Height = 232

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
```

...\OLE\APPS\SAMP02.REX

```
WshShellObj = .OLEObject~New("WScript.Shell")

WshEnv = WshShellObj~Environment
Say "Operating system:" WshEnv["OS"]
Say "You have" WshEnv["NUMBER_OF_PROCESSORS"] "processor(s) of",
    WshEnv["PROCESSOR_ARCHITECTURE"] "architecture in your system."

Say "The following directories represent special folders on your system:"
Do Folder Over WshShellObj~SpecialFolders
  Say "   " Folder
End

Say "Creating a shortcut for NOTEPAD.EXE on your Desktop..."
Desktop = WshShellObj~SpecialFolders("Desktop")
ShortCut = WshShellObj~CreateShortcut(Desktop || "\Shortcut to Notepad.lnk")
ShortCut~TargetPath = "%WINDIR%\notepad.exe"
ShortCut~Save

WshShellObj~Popup("Processing of REXX script has finished!")
```

...\OLE\APPS\SAMP03.REX

```
WshNetObj = .OLEObject~New("WScript.Network")

Say "Computer Name:" WshNetObj~ComputerName
Say "User Domain:" WshNetObj~UserDomain
Say "User Name:" WshNetObj~UserName

Say "The following network drives are currently mapped:"
MappedDrives = WshNetObj~EnumNetworkDrives
Do i=0 To MappedDrives~Count/2 - 1
  Say "   Drive" MappedDrives[i*2] "is mapped to" MappedDrives[i*2 + 1]
End

Say "The following network printers are currently connected:"
Printers = WshNetObj~EnumPrinterConnections
Do i=0 To Printers~Count/2 - 1
  Say "   Port" Printers[i*2] "is connected to" Printers[i*2 + 1]
End
```

...\OLE\APPS\SAMP09.REX

```
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

/* save sheet in default TEMP directory */
Worksheet~SaveAs( value("TEMP",,ENVIRONMENT)"\demo.xls")
excelObject~Quit
```

...\OLE\APPS\SAMP10.REX

```
fsObject = .OLEObject~new("Scripting.FileSystemObject")
allDrives = fsObject~drives
if allDrives = .NIL then do
  say "The object did not return information on your drives!"
  exit 1
end

do i over allDrives
  info = i~DriveLetter "-"
     /* show the DriveType in human-readable form */
  j = i~DriveType
  select
     when j=1 then do
            info = info "Removable"
          end
    when j=2 then do
            info = info "Fixed"
          end
    when j=3 then do
           info = info "Network"
          end
    when j=4 then do
            info = info "CD-ROM"
          end
    when j=5 then do
            info = info "RAM Disk"
          end
    otherwise
          info = info "Unknown"
  end

     /* append the ShareName for a network drive... */
  if j=3 then info = info i~ShareName
     /* ...and the VolumeName for the other ones */
  else if i~IsReady then info = info i~VolumeName
  say info
end
```

...\OLE\APPS\SAMP11.REX

```
/* Get stock price from IBM internet page with MS IE and OLE */

Explorer = .OLEObject~new("InternetExplorer.Application")
/* uncomment the next line if you want to see what is happening */
-- Explorer~Visible = .true
Explorer~Navigate("http://www.ibm.com/investor/")

/* wait for browser to load the page */
/* if the page is not loaded by then, an error will occur */
call SysSleep 5

/* obtain text representation of the page */
doc = Explorer~document           -- DOM document
body = doc~body                   -- get BODY
textrange = body~CreateTextRange  -- get TextRange
text = textrange~Text             -- get the contents

/* extract stock price information */
parse var text . "Current price:" stockprice "0d0a"x .
if stockprice = "" then stockprice = "<could not read stock price>"

/* end Explorer */
Explorer~quit

say "IBM stocks are at" stockprice"."
```

Excerpt from: ...\OLE\APPS\SAMP12.REX

```
/* instantiate an instance of the Internet Explorer */
myIE = .watchedIE~new("InternetExplorer.Application","WITHEVENTS")

myIE~visible = .true
myIE~navigate("http://www.ibm.com/")

/* wait for the OnQuit event of the browser to change */
/* the !active attribute of the REXX object to false  */
myIE~!active = .true
do while myIE~!active = .true
  call syssleep(2)
end
```

```
::CLASS watchedIE SUBCLASS OLEObject

/* ... Cut ... Lines deleted, please lookup the original file in your installation ! */

/* this is an event of the Internet Explorer */
::METHOD TitleChange
  use arg Text
  say "The title has changed to:" text

/* this is an event of the Internet Explorer */
::METHOD OnQuit
  self~!active = .false    -- terminates the waiting loop in main code

::METHOD !active ATTRIBUTE -- store the active attribute
```

...\OLE\APPS\SAMP14.REX

```
-- Initialize string to database path.
strDB = "c:\temp\newdb.mdb"

-- Create new instance of Microsoft Access.
appAccess = .OLEObject~new("Access.Application")

-- Open database in Microsoft Access window.
appAccess~NewCurrentDatabase(strDB)

-- Get Database object variable.
dbs = appAccess~CurrentDb

-- Create new table.
tdf = dbs~CreateTableDef("Contacts")

-- Create field in new table.

/* Please note how to access the constant.
   Microsoft documentation and the MS OLEViewer output
   these constants as dbText, dbBinary, etc. - the type library
   however prints them as DB_TEXT, DB_BINARY, etc.. Unless
   documentation is found why the names should be translated,
   the OLE code will *NOT* convert the names. */
fld = tdf~CreateField("CompanyName", appAccess~getConstant("db_Text"), 40)

-- Append Field and TableDef objects.
tdf~Fields~Append(fld)
dbs~TableDefs~Append(tdf)

appAccess~quit
```

...\OLE\ADSI\ADSI1.REX

```
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)
-- since summer 2006 with ooRexx 3.1 the following is preferable:
say left("Class:",10) myComputer~dispatch("class")

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

...\OLE\ADSI\ADSI2.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")     -- get ComputerName
UserID       = value("USERNAME",,"ENVIRONMENT" )        -- get UserName

userObject = .OLEObject~GetObject("WinNT://"||ComputerName||"/"||UserID||",user")

/* using the object property */
say "The full name for" UserID "is" userObject~FullName

/* using the standard get method for ADSI objects */
say "The full name for" UserID "is" userObject~Get("FullName")

say "Would you like to rename the full name (y/n)?"
pull answer

if answer = "Y" then do
  say "New full name:"
  parse pull answer

  /* set the property                                                  */
  /* as an alternative, the property can also be set with the standard put */
  /* method of ADSI objects:                                           */
  /* userObject~Put("FullName",answer)                                 */
  userObject~FullName=answer

  /* because properties are cached to avoid network calls, changing the    */
  /* properties of an object will only affect the cache at first.          */
  /* the object gets updated with the SetInfo method:                      */
  userObject~SetInfo

  say "updated the full name for" UserID
end
```

...\OLE\ADSI\ADSI3.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

container = .OLEObject~GetObject("WinNT://"ComputerName||"/Administrators")

do member over container~members
  say member~dispatch("class") ":" member~name "[" member~description "]"
end
```

...\OLE\ADSI\ADSI4.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

computerObject = .OLEObject~GetObject("WinNT://"||ComputerName)

computerObject~Filter = .array~of("Group","Service")

/* show only objects of type Group and Service: */
do item over computerObject
  /* avoid calling the CLASS method of the REXX object by using the */
  /* "unknown" mechanism (this calls the CLASS method of "item").   */
  say item~unknown("class",.nil) ":" item~name
  /* above is same as: say item~dispatch("class") ":" item~name */
end
```

...\OLE\ADSI\ADSI5.REX

```
myADS = .OLEObject~GetObject("ADs:")

do namespace over myADS
  say "Domains in" namespace~Name
  do domain over namespace
    if domain \= .nil then
      say "   " domain~name
    else
      say domain
  end
end
```

...\OLE\ADSI\ADSI6.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

myDomain = .OLEObject~GetObject("WinNT://"||ComputerName)
mySchemaClass = .OLEObject~GetObject(myDomain~schema)

say "Properties for the" myDomain~name "object:"
say

if mySchemaClass~container = 1 then do
  say myDomain~name "may contain the following objects:"
  do i over mySchemaClass~Containment
    say "   " i
  end
end
else
  say myDomain~name "is not a container."

say
say "Mandatory properties:"
do i over mySchemaClass~MandatoryProperties
  say "   " i
end

say
say "Optional properties:"
do i over mySchemaClass~OptionalProperties
  say "   " i
End
```

...\OLE\ADSI\ADSI7.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")  -- get ComputerName
computer = .OLEObject~GetObject("WinNT://"||ComputerName)
/* create a new group */
newGroup = computer~Create("group", "REXX-TestGroup")
newGroup~Description = "A test group created with REXX"
newGroup~SetInfo

/* make sure the information in the object cache is up-to-date */
newGroup~GetInfo
say "Created new group" newGroup~Name
say "Description:" newGroup~Description; say
say "Creating 15 users in this group:"
say "User01..User15 with passwords demo01..demo15"
/* create several new users */
do i = 1 to 15
  /* create name and other information */
  userName = "User"right(i,2,'0')
  userFullName = "Demo User Number" i
  userDescription = "A demo user that was created with REXX"
  userPassword = "demo"right(i,2,'0')

  newUser = computer~Create("user", userName)
  newUser~FullName = userFullName
  newUser~Description = userDescription
  newUser~SetPassword(userPassword)
  newUser~SetInfo
  newGroup~Add(newUser~ADsPath)
end
```

...\OLE\ADSI\ADSI8.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")  -- get ComputerName
computer = .OLEObject~GetObject("WinNT://"||ComputerName)

say "Removing the fifteen users..."
do i = 1 to 15
  computer~Delete("user", "User"||right(i,2,'0'))
end

say "Removing the test group..."

computer~Delete("group", "REXX-TestGroup")

say "done"
```

...\OLE\WMI\ACCOUNTS.REX

```
WMIObject = .OLEObject~GetObject("WinMgmts:{impersonationLevel=impersonate}")
userAccounts = WMIObject~InstancesOf("Win32_Account")

do instance over userAccounts
  say
  say "="~copies(16) instance~name "="~copies(16)
  do i over instance~properties_
    say left(i~name":",20,' ') i~value
  end
end
```

...\OLE\WMI\PROCESS.REX

```
WMIObject = .OLEObject~GetObject("winmgmts:{impersonationLevel=impersonate}")

say "Here is a list of currently running processes"

do process over WMIObject~InstancesOf("win32_process")
  say process~processid process~name
end
```

Weitere Links, 1

- Rexx Language Association (RexxLA)

  http://www.RexxLA.org/

  - Zahlreiche weiterführende Links
- Object Rexx bezogene OLE/ActiveX URL

  http://pragmaticlee.safedataisp.net/

  - Lee Peedin's ActiveX für Object Rexx Seite
- OLE-/Active-X Query Tool in Object Rexx
  - Die folgende Datei beinhaltet alle Erklärungen und URLs, die Sie benötigen:

    http://wi.wu-wien.ac.at/rgf/rexx/orx15/readme.html

    - Mit Microsoft's Internet Explorer (Version 6.0 oder höher) die Datei "**`rgf_oleinfo.hta`**" öffnen (eine DHTML Anwendung in Object Rexx)
    - Möglicherweise müssen Sie die Sicherheitsstufe im Internet Explorer für lokale Anwendungen auf "sehr niedrig" setzen
      - Extras -> Internetoptionen ... -> Sicherheit -> Stufe anpassen ...

Weitere Links, 2

- Microsoft (!) Skriptsammlung für Object Rexx

  http://www.microsoft.com/technet/scriptcenter/scripts/rexx/default.mspx

- Microsoftsammlung von administrativen Scripts für Windows
  - Beschreibt selbstzerlegende Windows Hilfedatei

    http://download.microsoft.com/download/.NetEnterpriseServer/Utility/1.0/NT5XP/EN-US/netscrpt.exe

    - Visual Basic Script Sammlung für verschiedenste Verwaltungsaufgaben
    - Einfach auf Object Rexx übertragbar
      - ersetzen Sie den Punkt in Visual Basic Script Programmen mit dem Object Rexx Nachrichtenoperator (die Tilde): ~

- Skripting Buch für Windows-Administratoraufgaben

  http://www.microsoft.com/mspress/books/6417.asp

  - Buch, dessen Inhalt auch auf Windows XP anwendbar ist

DHTML: Übersicht

- Begriffe
- DOM/DHTML
- Beispiele

Begriffe, 1 (Auszeichnungssprachen, Markup Languages)

- Tag (Marke)
  - Ermöglicht Marken anzugeben, die Texte umschließen
    - Öffnende Marke (englisch: opening tag)

      `<some_tag_name>`
    - Schließende Marke (englisch: closing tag)

      `</some_tag_name>`
  - Ermöglicht das Analysieren von Texten, indem festgestellt werden kann, welche Textteile von welchen Marken umschlossen sind
  - "Element"
    - Die Abfolge (Sequenz) "opening tag", Text, "closing tag"

Begriffe, 2 (Auszeichnungssprachen, Markup Languages)

- Dokumententypdefinition, Document Type Definition (DTD)
  - Definiert Marken (Tags) und ihre Attribute
    - Bezeichner (Namen) für Marken (Tags)
    - Attribute für Marken (Tags)
    - "Content model"
      - Verschachtelung von Marken und die erlaubte Ineinanderreihung
        - **Hierarchische Struktur !**
      - Ermöglicht das Festlegen, wie oft ein Element auftreten darf
  - "Instanz" einer DTD
    - Ein Dokument, dessen Text entsprechend der DTD-Regeln ausgezeichnet (englisch: "to mark up") wurde
    - Ein Dokument, das daraufhin überprüft wurde, dass die DTD-Regeln beachtet befolgt wurden, wird als "valide" ("validated ") bezeichnet

Begriffe, 3 (Auszeichnungssprachen, Markup Languages)

- HTML
  - Eine Auszeichnungssprache für das WWW
    - HTML-Browser
      - Zerlegt (englisch: to parse) ein HTML-ausgezeichnetes Dokument
      - Formatiert den Text, abhängig von den verwendeten Marken
  - DTD
    - Version 4.01: drei Variationen definiert
    - SGML-basiert, z.B.
      - Ermöglicht es, die Namen von Marken (Tags) und Attributen unabhängig von ihrer Groß- und Kleinschreibung anzugeben
      - Manche Endemarken (end-tags) können augelassen werden, wenn man sie aufgrund der DTD-Regeln eindeutig einsetzen kann
      - Es ist möglich, Ausschließungsregeln zu definieren

Begriffe, 4 (Auszeichnungssprachen, Markup Languages)

- XML
  - Eine leicht vereinfachte Version von SGML
    - Erlaubt das Definieren von DTDs für Auszeichnungssprachen (englisch: markup languages)
      - Ermöglicht seit 2002 die alternative Definition in Form von XML Schemas, standardisiert von: `http://www.w3c.org`
    - Namen müssen exakt in der definierten Groß- und Kleinschreibung angegeben werden
    - Endemarken (englisch: end tags) müssen immer angegeben werden
    - Werte von Attribute können neben doppelten Anführungszeichen auch in Apostrophen eingeschlossen werden
    - Es ist möglich, leere Elemente ausdrücklich anzugeben

```
<some_tag_name/>
```

Begriffe, 5 (Auszeichnungssprachen, Markup Languages)

- XML DTDs können ausgelassen werden
  - DTD kann hergeleitet werden, wenn es "wohlgeformt" (englisch: "well-formed") ist
    - Alle Marken (tags) müssen ineinandergeschachtelt sein
    - Marken dürfen nicht überlappen
    - Öffnende Marken müssen passende Endemarken aufweisen
- Struktur ist unabhängig vom Aufbereiten (Formatieren)!
  - Cascading Style Sheets (CSS)
    - Ermöglichen die Definition von Formatierungsregeln für Elemente
    - Möglich, spezifische Formatierungsregeln für Elemente anzugeben, bei denen Attribute bestimmte Werte aufweisen, oder der Position/Reihenfolge von Elementen innerhalb von anderen!

Beispiel (HTML)

- Text, in HTML ausgezeichnet

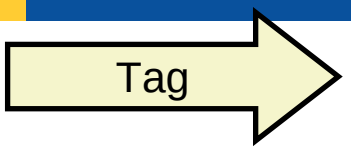
```
<html>
  <head>
     <title>This is my HTML file</title>
  </head>
  <body>
     <h1>Important Heading</h1>
     <p>This <span class="verb">is</span> the
        first paragraph.
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.
     <p id="9876">This <span class="verb">is</span> it.
  </body>
</html>
```

Beispiel: kaskadierendes Stylesheet einbinden

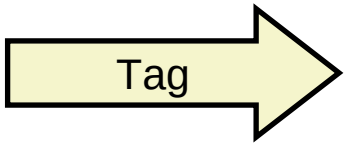
- Text, in HTML ausgezeichnet

```
<html>
  <head>
     <title>This is my HTML file</title>
     <link rel="stylesheet" type="text/css" href="example2.css">
  </head>
  <body>
     <h1>Important Heading</h1>
     <p>This <span class="verb">is</span> the
        first paragraph.
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.
     <p id="9876">This <span class="verb">is</span> it.
  </body>
</html>
```

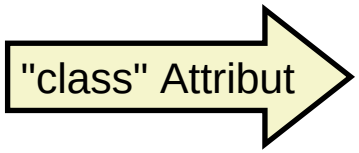
Beispiel eines Cascading Style Sheets



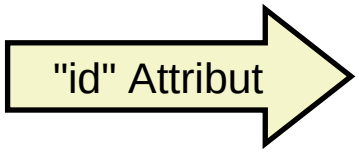

```
H1      { color: blue;
          text-align: center;
          font-family: Arial,sans-serif;
          font-size: 200%; }
```



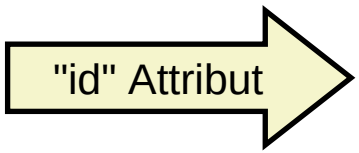

```
body    { background-color: yellow;
          font-family: Times, Avantgarde;
          font-size: small; }
```




```
.verb   { background-color: white;
          color: red;
          font-weight: 900; }
```




```
#xyz1   { font-variant: small-caps;
          text-align: right; }
```



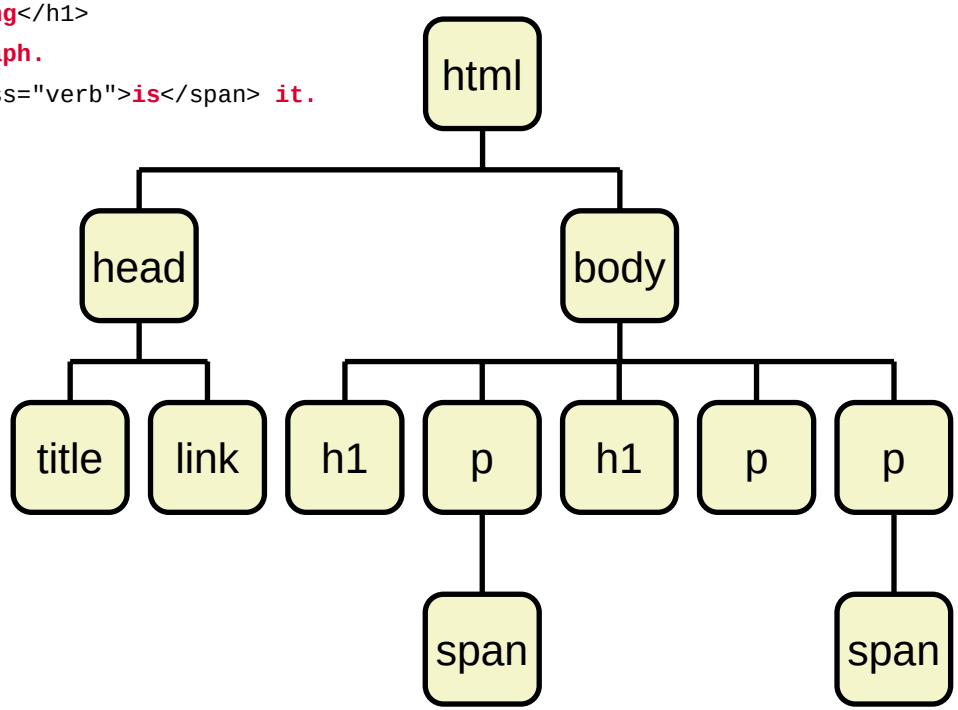

```
#9876   { font-size: large; }
```


Document Object Model (DOM)

- Zerlegen (englisch: parsing) einer HTML/XML-Datei
  - Erzeugt einen Zerlegungsbaum (parse tree), in dem die Elemente die Knoten sind
    - Jeder HTML-Browser muss dies mit HTML-Dateien tun!
- Programmschnittstellen (Application Programming Interface) für
  - Erzeugen, Abfragen, Ändern und Löschen von Knoten im Baum
    - Beinhaltet auch die Attribute der Marken !
  - Abfangen von Ereignissen, während mit dem Zerlegungsbaum gearbeitet wird
    - Benutzergenerierte Ereignisse als Folge von Maus- oder Tastenaktionen
    - Applikationsgenerierte Ereignisse wie "Dokument geladen" (englisch: "document loaded")

Document Object Model (DOM)
Beispiel

```
<html>
  <head>
     <title>This is my HTML file</title>
     <link rel="stylesheet" type="text/css" href="example2.css">
  </head>
  <body>
     <h1>Important Heading</h1>
     <p>This <span class="verb">is</span> the
        first paragraph.
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.
     <p id="9876">This <span class="verb">is</span> it.
  </body>
</html>
```



Programmanweisungen zu HTML/XML hinzufügen, 1

- <script>-Marke (Tag)

```
<script language="Object Rexx"> -- <![CDATA[
      ... Rexx code ...
    -- ]]>
</script>
```

  - oder:

```
<script language="Object Rexx" src="file.rex"></script>
```

- "on..."-Attribute

```
<some_tag onclick="call beep 90,90" language="Object Rexx">
```

  - Man kann als öffentlich (Schlüsselwort PUBLIC) definierte Rexx Routinen direkt aufrufen
    - "**this**"
      - MS IE Objekt, die das Element repräsentiert, für das das Ereignis ausgelöst wurde
      - Kann als Argument für die aufgerufene Routine benutzt werden

Programmanweisungen zu HTML/XML hinzufügen, 2

- Microsoft Internet Explorer (MSIE)
  - Windows Script Host
  - Stellt das OLE-Objekt "window" von selbst zur Verfügung
    - Ein COM Objekt zur Implementierung von DOM
      - "DHTML": dynamic HTML
    - Auch alle OLE-Objekt-Eigenschaften von "window" werden von MSIE eingebracht, z.B. "document"
      - Repräsentiert ein HTML/XML-Dokument
      - Verfügt über alle Funktionen/Methoden, um z.B. **all(e)** Knoten (nodes) oder alle **tables** (Tabellen) oder Elemente eines bestimmten Typs (d.h. mit demselben Namen für eine Marke (englisch: tagname)
      - Ermöglicht das Hinzufügen, Ändern und Löschen von Elementen
  - Kontrolliert die Ausführung der eingebetteten Skriptprogramme
    - Extrahieren, Starten der entsprechenden Programmanweisungen

Programmanweisungen zu HTML/XML hinzufügen, 3

- Programmanweisungen können überall in den Dokumenten eingestreut werden
  - Ausführung erfolgt in der Reihenfolge, in der die Anweisungen im Dokument aufgefunden wurden (englisch: document order)
  - Öffentliche Routinen können anschließend von *überall* aus aufgerufen werden
  - Programmanweisungen, die in Ereignisattributen angegeben sind, werden dann ausgeführt, wenn das Ereignis eintritt (englisch: "event fires")

Beispiel: HTML-Datei mit eingestreuten Rexx Programmanweisungen, 1

```
<html>
  <head>
     <script language="Object Rexx">
        document~writeln("<title>Produced by Rexx #1</title>")
     </script>
  </head>
  <body>
     <script language="Object Rexx">
        document~writeln("It is:<em>" date("s") time()"</em>, isn't it?")
     </script>
  </body>
</html>
```

Beispiel: HTML-Datei mit eingestreuten Rexx Programmanweisungen, 2

```
<html>
  <head>
     <script language="Object Rexx">
        document~writeln("<title>Produced by Rexx # 2</title>")
        ::routine info public -- to be called later on
           use arg o
           window~alert("this=["o~tagName"] innerText=["o~innerText"]")
           window~alert("this=["o~tagName"] innerHtml=["o~innerHtml"]")
           window~alert("this=["o~tagName"] outerHtml=["o~outerHtml"]")
           tmpStr=""
           do item over document~all  -- iterate over all elements
              tmpStr=tmpStr item~tagName
           end
           call alert "elements:" tmpStr
     </script>
  </head>
  <body onclick="call info this" language="Object Rexx">
     <script language="Object Rexx">
        document~writeln("It is:<em>" date("s") time()"</em>, isn't it?")
     </script>
  </body>
</html>
```

Sicherheitsüberlegungen

- MSIE
  - "Sandbox"
  - Object Rexx benutzt seinen eigenen "Security Manager", um z.B.
    - Den Zugriff auf die Umgebung zu verbieten
    - Skripte können faktisch auf das Loginverzeichnis beschränkt werden, etc.
- Sichere, lokale Ausführung
  - Benennen Sie die Dateiendung von "html" auf "hta" um
    - "HTML Application"

DHTML: Zusammenfassung

- HTML/XML Dateien
  - Hierarchisch (Zerlegungsbaum)
  - Abfragen, Hinzufügen, Ändern und Löschen von Elementen des Zerlegungsbaums
  - Programmanweisung mit Ereignissen verknüpfen
- DOM
  - Document Object Model, W3C
  - DHTML
    - Microsofts Implementierung von DOM
    - Unvollständig, proprietäre Erweiterungen
- Kann für die Erstellung von GUIs und Ausdrucken eingesetzt werden!

Abschließendes DHTML-Beispiel (Ein- und Ausgabe)

```
<head>
    <title>Object Rexx: Process User Input</title>

    <script language="Object Rexx">
        -- queries input value and outputs that string in reversed form
        ::routine work public
          output~innerhtml=reverse(input~value)
    </script>

</head>

<body>
    Please enter some text:<p>

    <input id="input" type="textarea" size="100"> <BR>

    <input type="BUTTON" onclick="call work" language="Object Rexx"
           value="Please Press Me!"><p>

    <p id="output">
</body>
```

Weitere Informationen

- World Wide Web Consortium

  http://www.w3c.org

  http://www.w3c.org/Style/CSS/

  http://www.w3c.org/DOM/

  http://www.w3c.org/MarkUp/ (HTML)

- Microsoft

  http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/dhtml.asp

  http://people.freenet.de/JavaScript/javap00.htm (Javascript-basiert)

- SelfHTML
  - Ausgezeichnete, leicht verständliche Ressourcen zu HTML, CSS, XML, DOM, …!
  - Deutschsprachige Texte und Tutorials
  - URL

    http://de.selfhtml.org/

Problem: Windows XP Service Pack 2 und MSIE

- Windows XP Professional, Service Pack 2, 2004-08-03
  - MSIE kann "aus Sicherheitsgründen" nicht mehr automatisch mit WSH-Skriptsprachen angesteuert werden
    - Gesteuert über einen Eintrag in der Registrierungsdatenbank ("Registry")
    - Object Rexx Beispiele funktionieren womöglich nicht mehr
  - Object Rexx verwendet immer schon die Sicherheitseinstellungen von MSIE
    - Sichere Ausführung zugesichert
- Lösungsmöglichkeiten
  - Dateiendung von ".htm" bzw. ".html" auf ".hta" umbenennen
    - "HTML"-Applications: erhalten uneinegeschränkten Zugriff auf das System
  - Änderung der entsprechenden Registry-Einstellung
    - Erlaubt Active Scripting wieder unter MSIE

Problemlösung: Registry-Änderung für MSIE

```
/* ---rgf, 2004-09-15
   purpose: enable/disable Active Scripting in MSIE

Object Rexx version of:
   From: "mayayana" <mayaXXyana1a@mindYYspring.com>
   Newsgroups: microsoft.public.scripting.wsh
   Subject: Re: Please confirm: No future for Scripting of IE?
   Date: Tue, 17 Aug 2004 15:39:27 GMT
   NNTP-Posting-Date: Tue, 17 Aug 2004 08:39:27 PDT
*/

parse upper arg switch .
bEnable= (switch = "") | (switch = 1) | (switch~left(3) = "YES") | (switch~left(2) = "ON")
if bEnable then iVal=0
           else iVal=1

SH = .oleObject~new("WScript.Shell") -- allows to use registry Windows-style

   -- define local part of the key in the registry pointing to IE
sReg = "Software\Microsoft\Internet Explorer\Main\" || -
       "FeatureControl\FEATURE_LocalMachine_Lockdown\IExplore.exe"

SH~RegWrite("HKLM\"sReg, iVal, "REG_DWORD")
SH~RegWrite("HKCU\"sReg, iVal, "REG_DWORD")

if iVal=0 then say "MSIE enabled for Active Scripting."
          else say "MSIE barred from Active Scripting."
```