Abteilung für Wirtschaftsinformatik

Processing XML Documents with SAX Using BSF4ooRexx

2013 International Rexx Symposium

RTP, North Carolina

Prof. Dr. Rony G. Flatscher

Overview

- Markup-Language
  - Basics
  - XML
- SAX-Parsing
  - Principles
  - Using ooRexx for listeners
  - Examples
- Roundup

Terms, 1
(Markup Languages)

- Tag
  - Enables one to use tags to embrace regular text
    - Opening tag (a.k.a. start tag)

      `<some_tag_name>`
    - Closing tag (a.k.a. end tag)

      `</some_tag_name>`
  - Allows for analyzing text, by noticing which parts of a text are surrounded ("embraced") by which tags
  - "Element"
    - The sequence "opening tag", text, "closing tag"

Terms, 2 (Markup Languages)

- Document Type Definition (DTD)
  - Defines the tags and their attributes, if any
    - Name (identifier) of the tag
    - Attributes for tags
    - "Content model"
      - Nesting of tags and the allowed sequence of tags
        - **Hierarchical structure !**
      - Allows to determine how many times an element may occur
  - "Instance" of a DTD
    - A document with text that got marked-up according to the rules defined in a DTD
    - A document that has been checked whether the DTD rules were applied correctly is named a "valid" document

Terms, 3 (Markup Languages)

- HTML
  - A markup language for the WWW
    - HTML-Browser
      - Parses a document marked up according to HTML
      - Formats the text, depending on the used tags
  - DTD
    - Version 4.01: three variants defined
    - SGML-based, hence it is possible to
      - Use any case for the tags and attribute names
      - Some closing tags may be omitted if the end tags can be determined by the rules set forth in the DTD
      - It is possible to define exclusions

Terms, 4
(Markup Languages)

- XML
  - A slightly simplified version of SGML
    - Allows the definition of DTDs for markup languages
      - Since 2002 an alternative got introduced in the form of "XML Schema": `http://www.w3c.org`
    - Tag and attribute names must be written in exact case
    - End tags must be always given
    - Attribute values can now be enclosed within apostrophes/single quotes (') in addition to double quotes (")
    - It is possible to explicitly denote empty elements

```
<some_tag_name/>
```

Terms, 5
(Markup Languages)

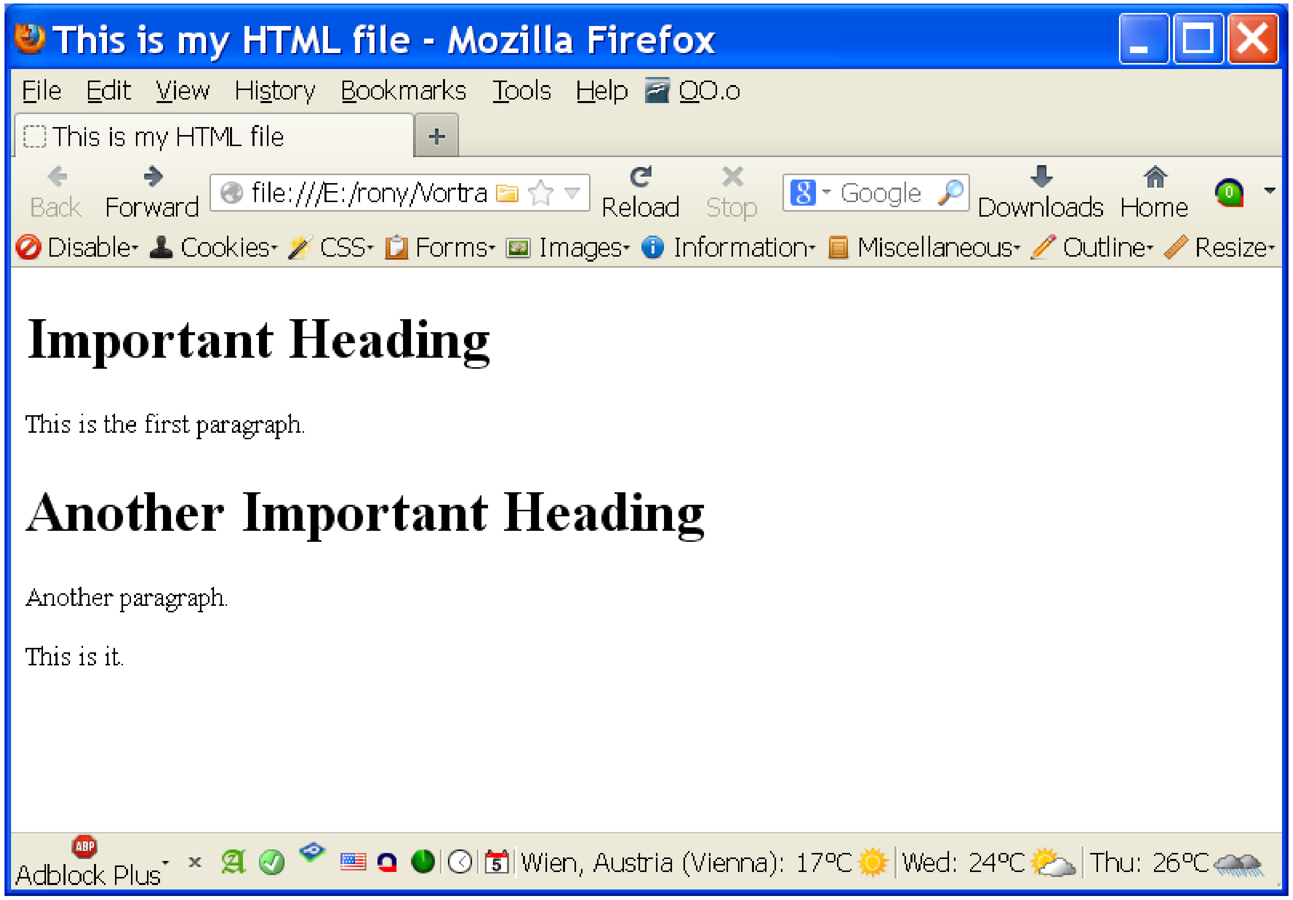
- XML DTDs can be omitted
  - A matching DTD can be always inferred, if the document is "well formed":
    - All tags must be nested
    - Tags must not overlap
    - Start tags must have matching end tags
- Structure is always independent of the formatting!
  - Cascading Style Sheets (CSS)
    - Allows to define formatting (layout) rules for elements
    - It is possible to define specific formatting (layout) rules for elements with attributes that have specific values or depending on the sequence of the elements

Terms (XHTML)

- Text, marked up in XHTML

```
<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.</p>
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.</p>
     <p id="a9876">This <span class="verb">is</span> it.</p>
  </body>
</html>
```

XHTML Text Rendered in Firefox


This is my HTML file - Mozilla Firefox
File Edit View History Bookmarks Tools Help OO.o
This is my HTML file
Back Forward file:///E:/rony/Vortra Reload Stop Google Downloads Home
Disable· Cookies· CSS· Forms· Images· Information· Miscellaneous· Outline· Resize·
Adblock Plus Wien, Austria (Vienna): 17°C Wed: 24°C Thu: 26°C

Important Heading

This is the first paragraph.

Another Important Heading

Another paragraph.

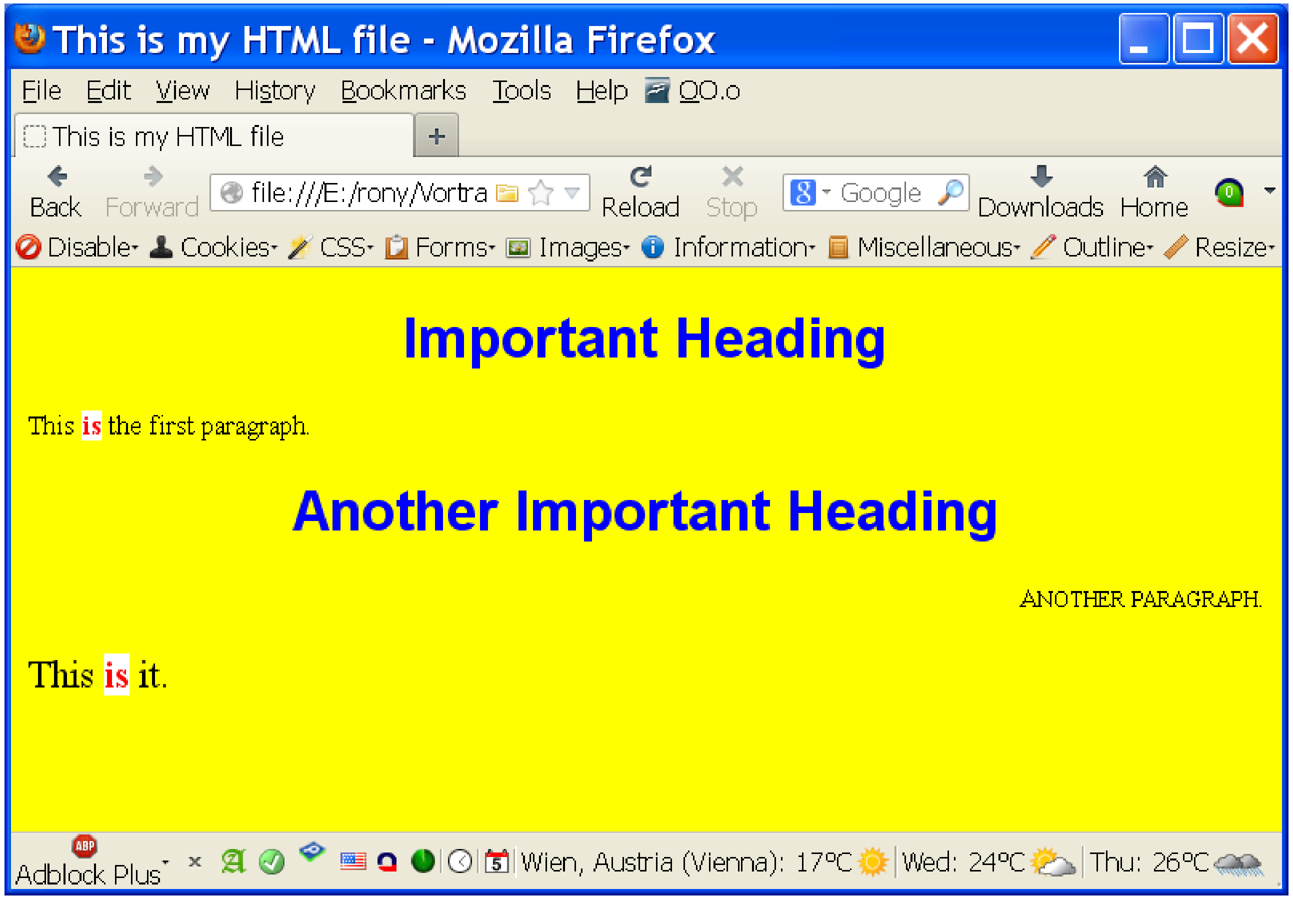
This is it.

Example: Linking a Cascading Style Sheet (CSS)

- Text, marked up in XHTML

```
<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.</p>
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.</p>
     <p id="a9876">This <span class="verb">is</span> it.</p>
  </body>
</html>
```

XHTML Text Rendered in Firefox with CSS


This is my HTML file - Mozilla Firefox
File Edit View History Bookmarks Tools Help OO.o
This is my HTML file
Back Forward file:///E:/rony/Vortra Reload Stop Google Downloads Home
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize
Adblock Plus
Wien, Austria (Vienna): 17°C Wed: 24°C Thu: 26°C

Important Heading

This is the first paragraph.

Another Important Heading

ANOTHER PARAGRAPH.

This is it.

Example of a Cascading Style Sheets (CSS)

Tag

```
h1       { color: blue;
           text-align: center;
           font-family: Arial,sans-serif;
           font-size: 200%; }
```

Tag

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

"class" Attribut

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

"id" Attribut

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

"id" Attribut

```
#a9876   { font-size: large; }
```

SAX (Simple API for XML), 1

- A SAX parser sequentially parses a XML document
- The Java SAX parser interfaces are defined in the package org.xml.sax
- Each time a meaningful piece of characters got parsed, the SAX parser will inform registered listener objects
  - The SAX parser available with Java defines the methods listener objects must implement: org.xml.sax.ContentHandler
  - Each method represents one "SAX event", including the argument a SAX parser supplies to listener objects

SAX (Simple API for XML), 2

- A SAX parser informs registered SAX event listener objects about the following SAX parsing events (in the following order)
  - setDocumentLocator(Locator locator)
  - → **startDocument()**
  - startPrefixMapping(String prefix, String uri)
    - → *skippedEntity(String name)*
    - → ***startElement(****String uri, String localName, String qName, Attributes atts****)***
    - → *ignorableWhitespace(char[] ch, int start, int length)*
    - → ***characters(****char[] ch, int start, int length****)***
    - → ***endElement(****String uri, String localName, String qName****)***
  - endPrefixMapping(String prefix)
  - → **endDocument()**

SAX (Simple API for XML), 3

- The interface org.xml.sax.ErrorHandler defines the methods a SAX/DOM error listener must implement
  - error(SAXParseException exception)
  - fatalError(SAXParseException exception)
  - warning(SAXParseException exception)
- org.xml.sax.SAXParseException has the following methods
  - getCause() returns a Throwable Java object representing the cause
  - getException() returns an embedded exception, if any
  - getMessage() returns a string with the detailed error message
  - toString() returns a string representation of the SAXParseException

Defining a SAX Listener in ooRexx

- Create an ooRexx listener class
  - For each SAX event you wish to process, create an ooRexx method by the same name and fetch the arguments, if any, using USE ARG
  - If SAX events are intentionally not handled, then define a method named UNKNOWN, such that Rexx does not raise a condition
- Create an ooRexx listener object from it
- Create a Java object that embeds the ooRexx listener object
  - BSFCreateRexxProxy(rexxListenerObject,[slotArg],interfaceName[,...])
  - interfaceName denotes the Java interface name which methods the Rexx listener object handles
    - It is possible to denote more than one Java interface, if the Rexx listener object is able to handle all methods defined by them!

"code01.rxj ": Extract Text from any XHTML Document The ooRexx Program (Main Program), 1

```
/* purpose: demonstrate how to extract the text from a xhtml file using SAX */
parse arg xmlFileName

rexxObject=.saxHandler~new -- create a Rexx SAX handler object
   -- wrap up the Rexx SAX handler as a Java object
javaProxy=BSFCreateRexxProxy(rexxObject,,"org.xml.sax.ContentHandler")

   -- create a Java SAX parser object and register our content handler object
parser=bsf.loadClass("org.xml.sax.helpers.XMLReaderFactory")~createXMLReader
parser~setContentHandler(javaProxy) -- set the content handler for this parser

eh=.errorHandler~new        -- create an error handler Rexx object
   -- wrap up the Rexx error handler as a Java object
javaEH=BsfCreateRexxProxy(eh, , "org.xml.sax.ErrorHandler")
parser~setErrorHandler(javaEH)    -- set the error handler for this parser

parser~parse(xmlFileName)   -- parse the InputStream, will call back

::requires BSF.CLS          -- get the Java support for ooRexx

::class "SaxHandler"     -- a Rexx content handler ("org.xml.sax.ContentHandler")
... cut ...
::class ErrorHandler     -- a Rexx error handler ("org.xml.sax.ErrorHandler")
... cut ...
```

"code01.rxj ": Extract Text from any XHTML Document The ooRexx Program (the ooRexx Classes), 2

```
... cut ...

parser~parse(xmlFileName)   -- parse the InputStream, will call back

::requires BSF.CLS          -- get the Java support for ooRexx

::class "SaxHandler"     -- a Rexx content handler ("org.xml.sax.ContentHandler")

::method characters         -- the callback method for characters (text)
  use arg textCharArray, start, length -- arguments from the Java SAX parser
  say pp(.bsf~new("java.lang.String", textCharArray, start, length)~toString)

::method unknown            -- intercept all other messages to avoid runtime error

::class ErrorHandler     -- a Rexx error handler ("org.xml.sax.ErrorHandler")

::method unknown         /* handles "warning", "error" and "fatalError" events */
  use arg methName, argArray  -- arguments from the Java SAX parser
  exception=argArray[1] /* retrieve SAXException argument             */
  .error~say(methName":" -
           "line="exception~getLineNumber",col="exception~getColumnNumber":" -
            pp(exception~getMessage))
```

"code01.rxj ": Extract Text from any XHTML Document Running the ooRexx Program, 3

```
f:\>rexx code01_text.rxj example2.html
[This is my HTML file]
[
     ]
[Important Heading]
[
     ]
[This ]
[is]
[ the
        first paragraph.]
[
     ]
[Another Important Heading]
[
     ]
[Another paragraph.]
[
     ]
[This ]
[is]
[ it.]
[
  ]
```

```
<!-- example2.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "DTD/xhtml1-transitional.dtd">
<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.</p>
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.</p>
     <p id="a9876">This <span class="verb">is</span> it.</p>
  </body>
</html>
```

"code02.rxj ": List Elements in Document Order The ooRexx Program (Main Program), 1

```
/* purpose: demonstrate how to list the element names in document order */
parse arg xmlFileName

rexxObject=.saxHandler~new -- create a Rexx SAX handler object
   -- wrap up the Rexx SAX handler as a Java object
javaProxy=BSFCreateRexxProxy(rexxObject,,"org.xml.sax.ContentHandler")

   -- create a Java SAX parser object and register our content handler object
parser=bsf.loadClass("org.xml.sax.helpers.XMLReaderFactory")~createXMLReader
parser~setContentHandler(javaProxy) -- set the content handler for this parser

eh=.errorHandler~new        -- create an error handler Rexx object
   -- wrap up the Rexx error handler as a Java object
javaEH=BsfCreateRexxProxy(eh, , "org.xml.sax.ErrorHandler")
parser~setErrorHandler(javaEH)    -- set the error handler for this parser

parser~parse(xmlFileName)   -- parse the InputStream, will call back

::requires BSF.CLS          -- get the Java support for ooRexx

::class "SaxHandler"       -- a Rexx content handler ("org.xml.sax.ContentHandler")
... cut ...
::class ErrorHandler       -- a Rexx error handler ("org.xml.sax.ErrorHandler")
... cut ...
```

Changes only in this class!

"code02.rxj" : List Elements in Document Order The ooRexx Program (the ooRexx Classes), 2

```
... cut ...

parser~parse(xmlFileName)  -- parse the InputStream, will call back

::requires BSF.CLS         -- get the Java support for ooRexx

::class "SaxHandler"    -- a Rexx content handler ("org.xml.sax.ContentHandler")

::method startElement      -- the callback method for characters (text)
  use arg   , localName
  say pp(localName)

::method unknown           -- intercept all other messages to avoid runtime error

::class ErrorHandler    -- a Rexx error handler ("org.xml.sax.ErrorHandler")

::method unknown        /* handles "warning", "error" and "fatalError" events */
  use arg methName, argArray  -- arguments from the Java SAX parser
  exception=argArray[1] /* retrieve SAXException argument          */
  .error~say(methName":" -
           "line="exception~getLineNumber",col="exception~getColumnNumber":" -
            pp(exception~getMessage))
```

"code02.rxj ": List Elements in Document Order Running the ooRexx Program, 3

```
f:\>rexx code02.rxj example2.html
[html]
[head]
[title]
[link]
[body]
[h1]
[p]
[span]
[h1]
[p]
[p]
[span]
```

```
<!-- example2.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "DTD/xhtml1-transitional.dtd">
<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.</p>
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.</p>
     <p id="a9876">This <span class="verb">is</span> it.</p>
  </body>
</html>
```

"code03.rxj" : List Elements in Document Order #2 The ooRexx Program (Main Program), 1

```
/* purpose: demonstrate how to list the element names in document order */
parse arg xmlFileName

rexxObject=.saxHandler~new -- create a Rexx SAX handler object
   -- wrap up the Rexx SAX handler as a Java object
javaProxy=BSFCreateRexxProxy(rexxObject,,"org.xml.sax.ContentHandler")

   -- create a Java SAX parser object and register our content handler object
parser=bsf.loadClass("org.xml.sax.helpers.XMLReaderFactory")~createXMLReader
parser~setContentHandler(javaProxy) -- set the content handler for this parser

eh=.errorHandler~new          -- create an error handler Rexx object
   -- wrap up the Rexx error handler as a Java object
javaEH=BsfCreateRexxProxy(eh, , "org.xml.sax.ErrorHandler")
parser~setErrorHandler(javaEH)    -- set the error handler for this parser

parser~parse(xmlFileName)  -- parse the InputStream, will call back

::requires BSF.CLS          -- get the Java support for ooRexx

::class "SaxHandler"      -- a Rexx content handler ("org.xml.sax.ContentHandler")
... cut ...
::class ErrorHandler      -- a Rexx error handler ("org.xml.sax.ErrorHandler")
... cut ...
```

Changes only in this class!

"code03.rxj ": List Elements in Document Order #2 The ooRexx Program (the ooRexx Classes), 2

```
... cut ...

parser~parse(xmlFileName)   -- parse the InputStream, will call back

::requires BSF.CLS          -- get the Java support for ooRexx

::class "SaxHandler"     -- a Rexx content handler ("org.xml.sax.ContentHandler")

::method init               -- ooRexx constructor
  expose level              -- object attribute (variable)
  level=0                   -- initialize to 0

::method startElement       -- the callback method for characters (text)
  expose level
  use arg   , localName
  say "   "~copies(level) || pp(localName)
  level+=1                  -- increase level by 1

::method endElement
  expose level
  level-=1                  -- decrease level by 1

::class ErrorHandler     -- a Rexx error handler ("org.xml.sax.ErrorHandler")
... cut ...
```

"code03.rxj": List Elements in Document Order #2 Running the ooRexx Program, 3

```
f:\>rexx code03.rxj example2.html
[html]
    [head]
        [title]
        [link]
    [body]
        [h1]
        [p]
            [span]
        [h1]
        [p]
        [p]
            [span]
```

```
<!-- example2.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "DTD/xhtml1-transitional.dtd">
<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.</p>
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.</p>
     <p id="a9876">This <span class="verb">is</span> it.</p>
  </body>
</html>
```

"code04.rxj" : List Elements with Text The ooRexx Program (Main Program), 1

```
/* purpose: demonstrate how to list the element names in document order */
parse arg xmlFileName

rexxObject=.saxHandler~new -- create a Rexx SAX handler object
   -- wrap up the Rexx SAX handler as a Java object
javaProxy=BSFCreateRexxProxy(rexxObject,,"org.xml.sax.ContentHandler")

   -- create a Java SAX parser object and register our content handler object
parser=bsf.loadClass("org.xml.sax.helpers.XMLReaderFactory")~createXMLReader
parser~setContentHandler(javaProxy) -- set the content handler for this parser

eh=.errorHandler~new         -- create an error handler Rexx object
   -- wrap up the Rexx error handler as a Java object
javaEH=BsfCreateRexxProxy(eh, , "org.xml.sax.ErrorHandler")
parser~setErrorHandler(javaEH)    -- set the error handler for this parser

parser~parse(xmlFileName)   -- parse the InputStream, will call back

::requires BSF.CLS          -- get the Java support for ooRexx

::class "SaxHandler"      -- a Rexx content handler ("org.xml.sax.ContentHandler")
... cut ...
::class ErrorHandler      -- a Rexx error handler ("org.xml.sax.ErrorHandler")
... cut ...
```

Changes only in this class!

"code04.rxj" : List Elements with Text The ooRexx Program (the ooRexx Classes), 2

```
... cut ...

::class "SaxHandler"    -- a Rexx content handler ("org.xml.sax.ContentHandler")
::method init              -- ooRexx constructor
  expose level             -- establish direct access to attribute
  level=0                  -- initialize to 0

::method startElement      -- the callback method for characters (text)
  expose level             -- establish direct access to attribute
  use arg   , localName
  say "   "~copies(level) || pp(localName)
  level+=1                 -- increase level by 1

::method endElement
  expose level             -- establish direct access to attribute
  level-=1                 -- decrease level by 1

::method characters        -- the callback method for characters (text)
  expose level             -- establish direct access to attribute
  use arg textCharArray, start, length -- arguments from the Java SAX parser
  say "   "~copies(level) "-->" pp(.bsf~new("java.lang.String", textCharArray, -
                                    start, length)~toString)
::class ErrorHandler    -- a Rexx error handler ("org.xml.sax.ErrorHandler")
... cut ...
```

"code04.rxj" : List Elements with Text
Running the ooRexx Program, 3

```
f:\>rexx code04.rxj example2.html
[html]
   [head]
      [title]
          --> [This is my HTML file]
      [link]
   [body]
       --> [
     ]
      [h1]
          --> [Important Heading]
       --> [
     ]
      [p]
          --> [This ]
         [span]
             --> [is]
          --> [ the
        first paragraph.]
       --> [
     ]
      [h1]
          --> [Another Important Heading]
       --> [
     ]
      [p]
          --> [Another paragraph.]
       --> [
     ]
      [p]
          --> [This ]
         [span]
             --> [is]
          --> [ it.]
       --> [
  ]
```

```
<!-- example2.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "DTD/xhtml1-transitional.dtd">
<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.</p>
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.</p>
     <p id="a9876">This <span class="verb">is</span> it.</p>
  </body>
</html>
```

Roundup

- Parsing any XML encoded document possible
  - Using BSF4ooRexx
  - Exploiting Java's functionality for parsing XML documents
- SAX parsing
  - SAX parser defines events
  - SAX parser invokes the respective SAX event method in the registered callback object
  - Concepts quite easy, memory efficient
- Easy to exploit from ooRexx !

Further Information

- World Wide Web Consortium ("W3C")
  - http://www.w3c.org
  - http://www.w3c.org/Style/CSS/
  - http://www.w3c.org/DOM/
  - http://www.w3c.org/MarkUp/ (HTML, XHTML2)
  - http://www.w3.org/QA/2002/04/valid-dtd-list.html (Doctype links)
- SAX specific URLs (as of 2013-05-01)
  - http://www.saxproject.org/ (current project home)
    - http://www.megginson.com/downloads/SAX/ (original creator)
  - http://www.cafeconleche.org/books/xmljava/chapters/index.html (book)
  - http://docs.oracle.com/javase/7/docs/api/org/xml/sax/package-summary.html (Java 7 docs)
- Sample files installed with BSF4ooRexx
  - bsf4oorexx/samples/SAX