

Dominik Stein

# **Wrapping the XML-parser 'expat' in Rexx/ObjectRexx.**

## **RexxExpat**

- An Introduction -

# Table of Contents

**Overview** of XML and expat

**Example** using XML and expat

**RexxExpat** calling expat

**Handlers** setting and invocation

RexxExpat **Special Issues**

**Outlook** and References

Examples

# Overview

## **XML** (eXtended Markup Language)

- standardized meta language to define individual document types
- allows automated processing and easy exchange of data

## **expat**

- conforming, non-validating XML-parser
- in C implemented
- supports individual encodings, namespace processing and external entity parsing

# Example: Using XML and expat

```
<?xml version="4.0" standalone='no' encoding='NativeAustrian' ?>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"  
                SYSTEM "http://www.w3.org/TR/REC-html40/loose.dtd">
```

```
<!-- XML declarations -->
```

```
<!NOTATION php SYSTEM "http://www.php.net/oneservices/php.exe">
```

```
<!NOTATION gif SYSTEM "http://www.dot.com/viewers/gifviewer.exe">
```

```
<!ENTITY logo SYSTEM "/bilder/RexxExpat.gif" NDATA gif>
```

```
<!ENTITY copyright SYSTEM "copyright.html">
```

```
<!ENTITY author "Dominik Stein">
```

**<HTML> <BODY>**

**<H1>**This is a little document to explain RexxExpat's work!**</H1>**

**<IMG SRC='logo' ALIGN=RIGHT />**

Well, an example for an external entity might be the German Umlaut 'ü' and the German ess-zett 'ß' in their escaped form: **&uuml;** and **&szlig;**. They are declared in the HTML-DTD (aren't they?).

**<?php echo("To serve XML documents, do like this\n"); ?>**

**<![CDATA[**

**>>>** This text could be copied from some file ;-). Declaring it as CDATA frees you from converting every language specific character like the German 'ü' and 'ß' to its escaped form ("&uuml;" and "&szlig;")! **<<< ]]>**

Author &author;

**</BODY> </HTML>**

# RexxExpat calling expat

1. create parser
2. initialize callback routines
3. parse and process document (via callback routines)
4. free parser

**Problem:** Separate environments in main program and callback routines

main program (Rexx) ➔ expat (C) ➔ callback routines (Rexx)

# Handler setting

Each handler is defined by three parameters **ProcName**, **EnvName**, **Instore** according to the following pseudo code:

```
if (ISNULLSTR(Instore) || ISZEROLENSTR(Instore) || Instore == 0)
    { ProcName is a file name of a Rexx procedure }
else if (Instore = "macro" || Instore = "MACRO")
    { ProcName is a macro loaded into the macrospace }
else
    { ProcName contains pure Rexx code }
```

# Handler invocation

- all wrapper functions equally **match** the corresponding expat functions
- handler functions **pass through** all arguments from C to Rexx
- callback routines do not **return codes** (few exceptions)

```
RexxStart ( ArgCount,  
            ArgList,  
            userData->Handler[HandlerType]->ProcName,  
            userData->Handler[HandlerType]->Instore,  
            userData->Handler[HandlerType]->EnvName,  
            RXSUBROUTINE,  
            NULL, /* no system exits for dll's available */  
            &result_int,  
            &result_str);
```



# RexxExpat: Special Issues (1)

UnknownEncodingHandler's return code defines **individual encodings**:

```
/*set XML_Encoding structure*/
currIndex = result_str.strptr;
for (i=0; currIndex && i<ENCODINGMAPSIZE
    && currIndex<=(result_str.strptr+result_str.strlength);
    i++, currIndex++) {
    if (isdigit((unsigned char)*currIndex)) /*byte defined*/
        info->map[i] = strtol(currIndex, &currIndex, 0);
    else /*byte left out*/
        info->map[i] = i;
}
/*fill up bytes*/
for (; i<ENCODINGMAPSIZE; i++) info->map[i] = i;
```

# RexxExpat: Special Issues (2)

**Variable pool functions** to exchange data between main program and callback routines:

```
numeric ExpatSetVariablePool( numeric parser,  
                             alphanumeric varnames, ...);
```

```
numeric ExpatGetVariablePool( numeric parser);
```

```
numeric ExpatFreeVariablePool(numeric parser);
```

**Possible obstacle:** insufficient memory due to duplication of variable pool

# Outlook

- implementation of an object 'expat' in ObjectRexx
- Rexx procedure which creates an object model from a given XML-file
- all current expat functions are wrapped
- future extensions to expat can be easily appended

# References

- W3C, Extensible Markup Language (XML) 1.0 (Second Edition), Oct. 6, 2000 (<http://www.w3.org/TR/REC-xml>)
- Kuno Dönhölter, Extensible Markup Language (XML), version 2.0, Sept. 1, 1998 (<http://members.aol.com/xmldoku>)
- xmlparse.h, version 1.2, Thai Open Source Software Center
- Clark Cooper, Using Expat, Sept. 1, 1999, XML.com, O'Reilly & Associates (<http://www.xml.com/pub/1999/09/expat/reference.html>)
- rexxsaa.h, version 1.5, Anders Christensen (/Mark Hessling), The Regina Rexx Interpreter
- Anders Christensen (/Mark Hessling), The Regina Rexx Interpreter, Aug. 14, 2000, (<ftp://ftp.lightlink.com/pub/hessling/Regina/reginapdf22.zip>)
- IBM, Object REXX for Windows NT and Windows95 Programming Guide (Version 1.03), third edition, May 1999 (<ftp://service.boulder.ibm.com/ps/products/ad/obj-xx/rexxpg.zip>)

# RexxExpat – Example (1)

RexxExpatExample.rexx

```
Call      RxFuncAdd 'ExpatLoadFuncs', 'RexxExpat.dll',  
          'ExpatLoadFuncs';
```

```
Call      ExpatLoadFuncs();
```

```
Parser    = ExpatParserCreate();
```

```
Call      ExpatSetElementHandler(Parser,  
          "StartElement.rexx", "", 0,  
          "EndElement.rexx", "", 0);
```

```
pad        = 5; i = 0;
```

```
myfile     = "Interface.xml"; mypos = 0;
```

```
Do while lines(myfile)<>0
```

```
    nextline = linein(myfile)
```

```
    mypos     = STREAM(myfile, 'Command',  
          'Query Position Read');
```

```
    Call      ExpatFreeVariablePool(Parser);
```

```
    Call      ExpatSetVariablePool(Parser, "i", "pad");
```

```
    retstr    = ExpatParse(Parser, nextline,  
          lines(myfile))
```

```
    Call      ExpatGetVariablePool(Parser);
```

```
    If retstr == 0 then say "### ERROR ###"
```

```
    Call      STREAM(myfile, 'Command',  
          'Position ='mypos Read);
```

```
End
```

```
Call      ExpatParserFree(Parser);
```

```
Call      ExpatDropFuncs();
```

## RexxExpat – Example (2)

StartElement.rexx

**Parse Arg** Parser, Name

**Call** RxFuncAdd 'ExpatSetVariablePool',  
'RexxExpat.dll', 'ExpatSetVariablePool';

**Call** RxFuncAdd 'ExpatGetVariablePool',  
'RexxExpat.dll', 'ExpatGetVariablePool';

**Call** ExpatGetVariablePool(Parser);

**Say** Insert(Name, "End", i\*padding+6);

**i** = i + 1;

**Call** ExpatSetVariablePool(Parser, "i");

**Call** RxFuncDrop 'ExpatSetVariablePool'

**Call** RxFuncDrop 'ExpatGetVariablePool'

**Exit**

EndElement.rexx

**Parse Arg** Parser, Name

**Call** RxFuncAdd 'ExpatSetVariablePool',  
'RexxExpat.dll', 'ExpatSetVariablePool';

**Call** RxFuncAdd 'ExpatGetVariablePool',  
'RexxExpat.dll', 'ExpatGetVariablePool';

**Call** ExpatGetVariablePool(Parser);

**i** = i - 1;

**Say** Insert("/Name, "End", i\*padding+6);

**Call** ExpatSetVariablePool(Parser, "i");

**Call** RxFuncDrop 'ExpatSetVariablePool'

**Call** RxFuncDrop 'ExpatGetVariablePool'

**Exit**

# RexxExpat – Data structures

```
/* ExpatHandlers - how many are there? */
```

```
enum EXPATHANDLERTYPES {  
    OnStartElement,  
    OnEndElement,  
    OnCharacterData,  
    [...]  
    NumberOfHandlers};
```

```
/* ExpatHandler - data storage */
```

```
typedef struct TypeDefExpatHandler {  
    PSZ          ProcName;  
    PSZ          EnvName;  
    PRXSTRING    Instore;    /* pointer to array[2] */  
} *EXPATHANDLER;
```

```
/* RexxExpatData - to make it thread-safe */
```

```
typedef struct TypeDefRexxExpatData {  
    /* Expat-Data */  
    XML_Parser    Parser;    /* parser */  
    XML_Encoding  Encoding;  /* encoding */  
    EXPATHANDLER  Handler[NumberOfHandlers];  
                                /* handler */  
  
    /* Rexx-Data */  
    PSHVBLOCK     shvfirst;  /* rexx variable pool */  
} *RexxExpatData;
```