Dominik Stein

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

RexxExpat

- An Introduction -

Table of Contents

Overview

XML (eXtended Markup Language)

- meta language to define individual document types
- allows automated processing and portable use of documents

expat

- conforming, non-validating XML-parser
- in C implemented
- supports individual encodings, namespace processing and external DTD (document type definitions)

Example: Using XML and expat

```
<?html 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/onlineservices/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 would be the left angle bracket (**&**lt**;**) and the ampersand (**&**amp**;**) in their escaped form: **&**amp**;**lt; and **&**amp**;**amp;. They are declared in the HTML-DTD.

<?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 key character like left angle brackets and ampersands to its escaped form ("<" and "&")! <<<
]]>

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)

```
int  _ExpatExternalEntityRefHandler ( XML_Parser parser,
                                      const XML_Char *context,
                                      const XML_Char *base,
                                      const XML_Char *systemId,
                                      const XML_Char *publicId)

numeric ExternalEntityRefHandler    ( numeric parser,
                                      alphanumeric context,
                                      alphanumeric base,
                                      alphanumeric systemId,
                                      alphanumeric publicId)
```

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

- 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, August 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
```