OpenEdge® ABL and XML - Essentials

Author: Raghuraman Kadambi
www.ultirags.com | www.skilglobal.com
raghu@ultirags.com | info@skilglobal.com

Contents

Introduction to XML

Definition

EXtensible Markup Language (XML) is a software and hardware independent tool for storing and transporting data.

That say's it all! XML are flat files designed to carry data - with focus on what data is.

History

XML history begins with the development of Standardised Generalised Markup Language (SGML) in the 1970's by Charles Goldfarb along with Ed Mosher and Ray Lorie while working at IBM (Anderson, 2004).

SGML is a language used to specify mark-up languages such as Hype Text Markup Language (HTML) or XML. The purpose of SGML was to create vocabularies which could be used to mark up documents with structural tags. It was thought at the time, that certain machine readable documents should remain machine readable for perhaps decades.

HTML remains popular even today as a presentation technology and is considered unsuitable as a data storage format.

SGML was too complicated to be used to store or transport data. This gave in birth to XML.

XML bridges this gap by being both human and machine readable, while being flexible enough to support platform and architecture independent data interchange.

XML Basics

My first experience with XML was in the year 1999. Since then the primary usage never seem to have changed; data exchange!

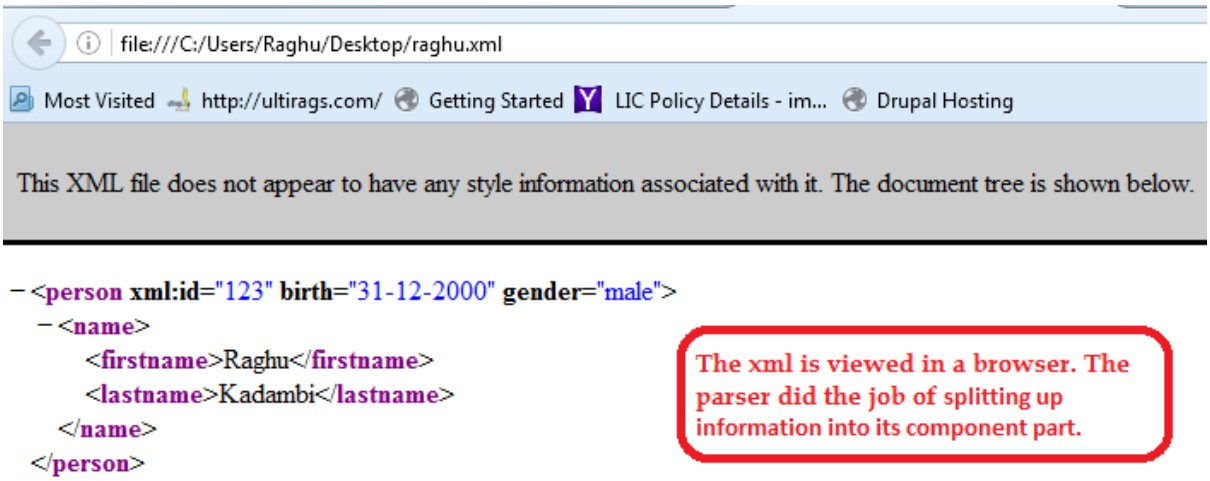
Over the years, XML has gained popularity primarily in the exchange of data; in other words, transport of data.

XML document generally, has an extension .xml.

```
<person xml:id="123" birth="31-12-2000" gender="male">
  <name>
    <firstname>Raghu</firstname>
    <lastname>Kadambi</lastname>
  </name>
</person>
```

Modern day browsers have built-in XML parsers parses (parsing is the act of splitting up information into its component part). The example above as:

1. Element <person> identified with Attribute xml:id (predefined type 'ID') containing "123" and Attribute birth containing "31-12-2000" and Attribute gender containing "male" containing the following additional information
2. Element <name> containing the following additional information
3. Element <firstname> containing text 'Raghu' followed by the following information
4. Element <lastname> containing text 'Kadambi'



Any modern development platform supplies the parser for you without you needing to do anything extra. Unless you are building your own software, there is no need to worry about finding an XML parser. This tutorial does cover the parsers. Also, this tutorial does not cover any example of using SAX parser technique.

You could learn more about XML in the following http://www.w3schools.com/xml/

OpenEdge® ABL and XML - Essentials

Here are few things that I have worked in the past with OpenEdge® ABL and XML:

1. *Create XML using temp-table* - We should be able to create an xml file out of the data in my temp-table (would use temp-table filed name as tags and attributes and the fields value as data).
2. *Read an XML into a temp-table* - We should be able to read the data in an XML file into a temp-table (would use temp-table filed name as tags and attributes and the fields value as data).
3. *Create XML* [using Document Object Model - DOM] - We should be able to create an xml file without temp-table.
4. *Read an XML* [using Document Object Model - DOM]- We should be able to read the data in an XML file without temp-table.

While I have worked in all the four situations, my personal favoutires are the first two. Nothing like working with the temp-tables ☺.

Let's look at each of these in this tutorial. For learning purpose, this tutorial uses sports2000 database.

Create XML using temp-table

Code:

xml-write-1.txt - Notepad

File Edit Format View Help

```
/*
* Program to create an xml from a temp-table using
* the WRITE-XML method
* Original source: http://documentation.progress.com
* © Progress Software Corporation, USA
*/
DEFINE VARIABLE cTargetType     AS CHARACTER NO-UNDO.
DEFINE VARIABLE cFile           AS CHARACTER NO-UNDO.
DEFINE VARIABLE lFormatted      AS LOGICAL   NO-UNDO.
DEFINE VARIABLE cEncoding       AS CHARACTER NO-UNDO.
DEFINE VARIABLE cSchemaLocation AS CHARACTER NO-UNDO.
DEFINE VARIABLE lWriteSchema    AS LOGICAL   NO-UNDO.
DEFINE VARIABLE lMinSchema      AS LOGICAL   NO-UNDO.
DEFINE VARIABLE lRetOK          AS LOGICAL   NO-UNDO.

DEFINE TEMP-TABLE ttCust NO-UNDO LIKE Customer.

/* Code to populate the temp-table  */
for each customer
  no-lock:
  create ttcust.
  buffer-copy customer to ttcust.
  release ttcust.
end.

ASSIGN
  cTargetType     = "file"
  cFile           = "ttcust.xml"
  lFormatted      = TRUE
  cEncoding       = ?
  cSchemaLocation = ?
  lWriteSchema    = TRUE
  lMinSchema      = FALSE.

lRetOK = TEMP-TABLE ttCust:WRITE-XML(cTargetType,
                                     cFile,
                                     lFormatted,
                                     cEncoding,
                                     cSchemaLocation,
                                     lWriteSchema,
                                     lMinSchema).
```

XML file name

Write the schema of the temp-table

WRITE-XML method creates the ttcust.xml with the schema of the temp-table along with the data

Output:


```
- <ttCust>
  - <xsd:schema>
    - <xsd:element name="ttCust" prodata:proTempTable="true">
      - <xsd:complexType>
        - <xsd:sequence>
          - <xsd:element name="ttCustRow" minOccurs="0" maxOccurs="unbounded">
            - <xsd:complexType>
              - <xsd:sequence>
                <xsd:element name="CustNum" type="xsd:int" nillable="true" prodata:format=">>>>9" prodata:label="Cust Num" prodata:help="Please enter a customer number."/>
                <xsd:element name="Country" type="xsd:string" nillable="true" default="USA" prodata:format="x(20)" prodata:help="Please enter a country."/>
                <xsd:element name="Name" type="xsd:string" nillable="true" prodata:format="x(30)" prodata:help="Please enter a name."/>
                <xsd:element name="Address" type="xsd:string" nillable="true" prodata:format="x(35)" prodata:help="Please enter an address."/>
                <xsd:element name="Address2" type="xsd:string" nillable="true" prodata:format="x(35)" prodata:help="Please enter an address."/>
                <xsd:element name="City" type="xsd:string" nillable="true" prodata:format="x(25)" prodata:help="Please enter a city."/>
                <xsd:element name="State" type="xsd:string" nillable="true" prodata:format="x(20)" prodata:help="Please enter standard state abbreviation."/>
                <xsd:element name="PostalCode" type="xsd:string" nillable="true" prodata:format="x(10)" prodata:label="Postal Code" prodata:help="Please enter the appropriate Postal Code."/>
                <xsd:element name="Contact" type="xsd:string" nillable="true" prodata:format="x(30)" prodata:help="Please enter a contact."/>
                <xsd:element name="Phone" type="xsd:string" nillable="true" prodata:format="x(20)" prodata:help="Please enter a phone number"/>
                <xsd:element name="SalesRep" type="xsd:string" nillable="true" prodata:format="x(4)" prodata:label="Sales Rep" prodata:help="Please Enter a Sales Rep."/>
                <xsd:element name="CreditLimit" type="xsd:decimal" nillable="true" default="1500.0" prodata:format="->,>>>,>>9" prodata:label="Credit Limit" prodata:help="Please enter a Credit Limit." prodata:decimals="2"/>
                <xsd:element name="Balance" type="xsd:decimal" nillable="true" prodata:format="->,>>>,>>9.99" prodata:help="Please enter a balance." prodata:decimals="2"/>
                <xsd:element name="Terms" type="xsd:string" nillable="true" default="Net30" prodata:format="x(20)" prodata:help="Please enter terms"/>
                <xsd:element name="Discount" type="xsd:int" nillable="true" prodata:format=">>9%" prodata:help="Please enter a percentage from 0 to 100."/>
                <xsd:element name="Comments" type="xsd:string" nillable="true" prodata:format="x(80)" prodata:help="Please enter comments."/>
                <xsd:element name="Fax" type="xsd:string" nillable="true" prodata:format="x(20)" prodata:help="Please enter a fax number."/>
                <xsd:element name="EmailAddress" type="xsd:string" nillable="true" prodata:format="x(50)" prodata:label="Email" prodata:help="Please enter an full Internet Email Address."/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    + <xsd:unique name="CustNum" prodata:primaryIndex="true"></xsd:unique>
    </xsd:element>
  + <xsd:annotation></xsd:annotation>
  </xsd:schema>
- <ttCustRow>
    <CustNum>2</CustNum>
    <Country>Finland</Country>
    <Name>Sreedher</Name>
    <Address>Rattipolku 3</Address>
    <Address2/>
    <City>Bangalore</City>
    <State>Uusima</State>
    <PostalCode>45321</PostalCode>
    <Contact>Urpo Leppakoski</Contact>
    <Phone>(603) 532 5471</Phone>
    <SalesRep>DKP</SalesRep>
    <CreditLimit>20000.00</CreditLimit>
    <Balance>437.63</Balance>
    <Terms>Net30</Terms>
    <Discount>35</Discount>
    <Comments>Ship all products 2nd Day Air.</Comments>
    <Fax/>
    <EmailAddress/>
  </ttCustRow>
```


Temp-Table *ttcust* schema

Sample Data - First record

Create XML using temp-table – with XML Node Attribute

Code:

Another example of creating an XML files using the previous technique but with a change in temp-table in a way that some of the data in the records goes into an XML file as an attribute of the field tag.

tt-xml-attribute.txt - Notepad

File Edit Format View Help

```
/*  Sample program - demonstrating addting XML-NODE-ATTRIBUTE to a temp-table */
define variable total_orders        as int       no-undo.
define variable var_target_type     as character no-undo.
define variable var_file            as character no-undo.
define variable var_formatted       as logical   no-undo.
define variable var_encoding        as character no-undo.
define variable var_schema_location as character no-undo.
define variable var_write_schema    as logical   no-undo.
define variable var_min_schema      as logical   no-undo.
/*
* define temp-table for store all customers
* along with their total count of orders placed by a customer
*/
define temp-table tt_customerlist no-undo
  fields tt_custnum     like customer.custnum
  fields tt_custname    like customer.name
  fields tt_city        like customer.city
  fields tt_totalorders as integer
         xml-node-type "attribute"   /* for total orders as attribute */
         xml-node-name "totalorders" /* name of the total orders attribute */.

/* populate temp-table with with the customer data */
for each customer
  no-lock:
 create tt_customerlist.
 assign
  tt_custnum     = customer.custnum
  tt_custname    = customer.name
  tt_city        = customer.city.
  /* calculate the total order placed by each customer */
  for each order no-lock
    where order.custnum = customer.custnum:
    total_orders = total_orders  + 1.
  end.
  assign
    tt_totalorders = total_orders /* assign the total order for a customer */
    total_orders    = 0. /* initialize the order count */
end.

/* assign XML file details */
assign
  var_target_type     = "file"
  var_file            = "tt_customerlist.xml"
  var_formatted       = yes
  var_encoding        = ?
  var_schema_location = ?
  var_write_schema    = no
  var_min_schema      = no.

/* create an XML file of the temp-table with the populated data */
temp-table tt_customerlist:write-xml(var_target_type,
                                     var_file,
                                     var_formatted,
                                     var_encoding,
                                     var_schema_location,
                                     var_write_schema,
                                     var_min_schema).
```

Output:

file:///D:/OpenEdge/WRK/tt_customerlist.xml

Most Visited http://ultirags.com/ Getting Started LIC Policy Details

This XML file does not appear to have any style information associated w


```
−<tt_customerlist>
  −<tt_customerlistRow totalorders="7">
      <tt_custnum>2</tt_custnum>
      <tt_custname>Sreedher</tt_custname>
      <tt_city>Bangalore</tt_city>
    </tt_customerlistRow>
  −<tt_customerlistRow totalorders="5">
      <tt_custnum>3</tt_custnum>
      <tt_custname>Sreedher</tt_custname>
      <tt_city>Bangalore</tt_city>
    </tt_customerlistRow>
  +<tt_customerlistRow totalorders="2"></tt_customerlistRow>
  +<tt_customerlistRow totalorders="12"></tt_customerlistRow>
  +<tt_customerlistRow totalorders="5"></tt_customerlistRow>
  +<tt_customerlistRow totalorders="4"></tt_customerlistRow>
  +<tt_customerlistRow totalorders="1"></tt_customerlistRow>
  +<tt_customerlistRow totalorders="5"></tt_customerlistRow>
  +<tt_customerlistRow totalorders="7"></tt_customerlistRow>
```


XML Node attribute that is defined as a field in the temp-table.

Read an XML into a temp-table

Code:

This code uses the previously created XML file *ttcust.xml.*

Note: My temp-table schema is same as the schema defined in the XML file.

xml-read-2.txt - Notepad

File Edit Format View Help

```
/*
* Program to read from an xml into a temp-table using the READ-XML method
* Original source: http://documentation.progress.com
* © Progress Software Corporation, USA
*/
DEFINE VARIABLE cSourceType              AS CHARACTER NO-UNDO.
DEFINE VARIABLE cReadMode                AS CHARACTER NO-UNDO.
DEFINE VARIABLE lOverrideDefaultMapping AS LOGICAL   NO-UNDO.
DEFINE VARIABLE cFile                    AS CHARACTER NO-UNDO.
DEFINE VARIABLE cSchemaLocation          AS CHARACTER NO-UNDO.
DEFINE VARIABLE cFieldTypeMapping        AS CHARACTER NO-UNDO.
DEFINE VARIABLE cVerifySchemaMode        AS CHARACTER NO-UNDO.
DEFINE VARIABLE lRetOK                   AS LOGICAL   NO-UNDO.
DEFINE VARIABLE httCust                  AS HANDLE    NO-UNDO.

define variable custHndle as handle no-undo.

DEFINE VARIABLE bh           AS HANDLE NO-UNDO.
DEFINE VARIABLE qh           AS HANDLE NO-UNDO.

custHndle = BUFFER Customer:handle.
CREATE TEMP-TABLE httCust.
httCust:create-like(custHndle).
httCust:TEMP-TABLE-PREPARE("ttCust").

/* Get the buffer handle for the temp-table */
bh = httCust:DEFAULT-BUFFER-HANDLE.
ASSIGN
  cSourceType              = "file"
  cFile                    = "ttcust.xml"
  cReadMode                = "empty"
  cSchemaLocation          = ?
  lOverrideDefaultMapping  = ?
  cFieldTypeMapping        = ?
  cVerifySchemaMode        = ?.
lRetOK = httCust:READ-XML(cSourceType,
                          cFile,
                          cReadMode,
                          cSchemaLocation,
                          lOverrideDefaultMapping,
                          cFieldTypeMapping,
                          cVerifySchemaMode).
/* Run a query to access the TEMP-TABLE */
CREATE QUERY qh.
qh:SET-BUFFERS(bh).
qh:QUERY-PREPARE("FOR EACH ttcust").
qh:QUERY-OPEN().
/* Display the order number and the salesrep name */
REPEAT:
  qh:GET-NEXT().
    IF qh:QUERY-OFF-END THEN LEAVE.
       display
        bh:buffer-field(1):buffer-value
        bh:buffer-field(2):buffer-value
        bh:buffer-field(2):name format "x(30)".
END.
qh:QUERY-CLOSE().
bh:BUFFER-RELEASE().
DELETE OBJECT httCust.
DELETE OBJECT qh.
```

XML file name

READ-XML method reads the xml schema and data and loads into the temp-table *ttcust.*

Create XML [using Document Object Model - DOM]

cust.xml - Sample XML file that the code below would use

cust.xml

```
<?xml version="1.0" ?>
<Customers>
  <Customer CustNum="1" Name="Lift Tours">
    <Country>USA</Country>
    <Address>276 North Drive</Address>
    <Address2></Address2>
    <City>Burlington</City>
    <State>MA</State>
    <PostalCode>01730</PostalCode>
    <Contact>Gloria Shepley</Contact>
    <Phone>(617) 450-0086</Phone>
    <SalesRep>HXM</SalesRep>
    <CreditLimit>66700</CreditLimit>
    <Balance>903.64</Balance>
    <Terms>Net30</Terms>
    <Discount>35</Discount>
    <Comments>This customer is on credit hold.</Comments>
    <Fax></Fax>
    <EmailAddress></EmailAddress>
  </Customer>
</Customers>
```

Code:

xml-create-dom-3.txt - Notepad

File Edit Format View Help

```
/*
* i-outcus.p - Export the Customer table to an xml file - uses DOM
* Original source: http://documentation.progress.com
* © Progress Software Corporation, USA
*/
DEFINE VARIABLE hDoc   AS HANDLE  NO-UNDO.
DEFINE VARIABLE hRoot  AS HANDLE  NO-UNDO.
DEFINE VARIABLE hRow   AS HANDLE  NO-UNDO.
DEFINE VARIABLE hField AS HANDLE  NO-UNDO.
DEFINE VARIABLE hText  AS HANDLE  NO-UNDO.
DEFINE VARIABLE hBuf   AS HANDLE  NO-UNDO.
DEFINE VARIABLE hDBFld AS HANDLE  NO-UNDO.
DEFINE VARIABLE ix     AS INTEGER NO-UNDO.

CREATE X-DOCUMENT hDoc.
CREATE X-NODEREF  hRoot.
CREATE X-NODEREF  hRow.
CREATE X-NODEREF  hField.
CREATE X-NODEREF  hText.

hBuf = BUFFER Customer:HANDLE.

/* Set up a root node */
hDoc:CREATE-NODE(hRoot,"Customers","ELEMENT").
hDoc:APPEND-CHILD(hRoot).
FOR EACH Customer
  WHERE Customer.CustNum < 5:
    hDoc:CREATE-NODE(hRow,"Customer","ELEMENT"). /* create a row node */
    hRoot:APPEND-CHILD(hRow). /* put the row in the tree */
    hRow:SET-ATTRIBUTE("CustNum", STRING(Customer.CustNum)).
    hRow:SET-ATTRIBUTE("Name", Customer.Name).

    /* Add the other fields as tags in the xml */
    REPEAT ix = 1 TO hBuf:NUM-FIELDS:
        hDBFld = hBuf:BUFFER-FIELD(ix).
        IF hDBFld:NAME = "CustNum" OR hDBFld:NAME = "Name" THEN NEXT.
        /* Create a tag with the field name */
        hDoc:CREATE-NODE(hField, hDBFld:NAME, "ELEMENT").

        /* Put the new field as next child of row */
        hRow:APPEND-CHILD(hField).

        /* Add a node to hold field value. The empty string (??) represents the value that will be set later. */
        hDoc:CREATE-NODE(hText, "", "TEXT").

        /* Attach the text to the field */
        hField:APPEND-CHILD(hText).
        hText:NODE-VALUE = STRING(hDBFld:BUFFER-VALUE).
    END.
END.
/* Write the XML node tree to an xml file */
hDoc:SAVE("file","cust.xml").

DELETE OBJECT hDoc.
DELETE OBJECT hRoot.
DELETE OBJECT hRow.
DELETE OBJECT hField.
```

Read an XML [using Document Object Model - DOM]

Code:

xml-read-dom-4.txt - Notepad

File Edit Format View Help

```
/*
* i-incus.p - Import the Customer table from an XML file - uses DOM
* Original source: http://documentation.progress.com
* © Progress Software Corporation, USA
*/
DEFINE VARIABLE hDoc   AS HANDLE  NO-UNDO.
DEFINE VARIABLE hRoot  AS HANDLE  NO-UNDO.
DEFINE VARIABLE hTable AS HANDLE  NO-UNDO.
DEFINE VARIABLE hField AS HANDLE  NO-UNDO.
DEFINE VARIABLE hText  AS HANDLE  NO-UNDO.
DEFINE VARIABLE hBuf   AS HANDLE  NO-UNDO.
DEFINE VARIABLE hDBFld AS HANDLE  NO-UNDO.
DEFINE VARIABLE ix     AS INTEGER NO-UNDO.
DEFINE VARIABLE jx     AS INTEGER NO-UNDO.

/* So we can create new recs */
DEFINE TEMP-TABLE ttCustomer LIKE Customer.
CREATE X-DOCUMENT hDoc.
CREATE X-NODEREF  hRoot.
CREATE X-NODEREF  hTable.
CREATE X-NODEREF  hField.
CREATE X-NODEREF  hText.
hBuf = BUFFER ttCustomer:HANDLE.
/* Read in the file created in i-outcus.p */
hDoc:LOAD("file", "cust.xml", FALSE).
hDoc:GET-DOCUMENT-ELEMENT(hRoot).

/* Read each Customer from the root */
REPEAT ix = 1 TO hRoot:NUM-CHILDREN:
    hRoot:GET-CHILD(hTable, ix).
    CREATE ttCustomer.

    /* Get the fields given as attributes */
    ttCustomer.CustNum = INTEGER(hTable:GET-ATTRIBUTE("CustNum")).
    ttCustomer.Name = hTable:GET-ATTRIBUTE("Name").
    /* Get the remaining fields given as elements with text */
    REPEAT jx = 1 TO hTable:NUM-CHILDREN:
        hTable:GET-CHILD(hField, jx).
        IF hField:NUM-CHILDREN < 1 THEN NEXT.
        /* Skip any null value */
        hDBFld = hBuf:BUFFER-FIELD(hField:NAME).
        hField:GET-CHILD(hText, 1).
        /* Get the text value of the field */
        hDBFld:BUFFER-VALUE = hTEXT:NODE-VALUE.
    END. /* REPEAT jx */
END. /* REPEAT ix */

DELETE OBJECT hDoc.
DELETE OBJECT hRoot.
DELETE OBJECT hTable.
DELETE OBJECT hField.
DELETE OBJECT hText.
/* show data made it by displaying temp-table */
FOR EACH ttCustomer:
    DISPLAY ttCustomer.Name.
END.
```

References, Credits, Trademarks and Copyrights

1. http://www.w3schools.com/xml/xml_whatis.asp
2. Original source: http://documentation.progress.com
3. Progress Software Corporation, USA – OpenEdge® Web Services documentation.
4. Progress®, OpenEdge®, AppServer® & WebSpeed® are trademarks of Progress Software Corporation, USA.
5. "PMP" & "PMI" are registered marks of the Project Management Institute (PMI), USA.
6. All other logo and trademarks referenced in this material are belong to their respective owners.
7. Copyright content belongs to its rightful owners.