Creating an XML File with PeopleTools using the XMLDoc object

PT 8.50+
Randall Groncki

Introduction

PeopleTools has several methods to create an XML file for internal or external consumption. We can use PeopleTools to quickly generate the data, enrich and shape it before sending it on to the destination.

This document will discuss creating an XML File using the XMLDoc object in PeopleCode.

The process in brief:

1. Create and load a rowset
2. Enrich the data in that rowset
3. Create and XMLDoc object and load the data
4. Use one of the XMLDoc's built-in methods to generate an XML string.

For demonstration purposes, our premise is creating an XML file for a BI Publisher report. However, there are other situations requiring XML Files. The XMLDoc is extremely flexible and powerful enough to generate the XML file as required by the target specification.

XML File Creation

Premise for XML File Creation Example

Demonstration premise: A letter to all employees summarizing their current job and showing all training taken through the company. This is a complex data set with a parent/child relationship. Each employee will get a separate page in the report. For each employee, there will be zero or more training entries. See Appendix A for the class populating this data structure. The class returns this Rowset object:

Root

Employee Record: X_EE_RPT_LTR_VW (Parent Record)

- EMPLID
- NAME
- DEPTID
- LOCATION
- JOBCODE
- COMPRATE
- CHANGE_PCT
- LOCATION_DESCR
- JOB_DESCR
- COMPANY_DESCR

Training Record: X_TRAINING_VW (Child Record)

- EMPLID
- COURSE_START_DT
- COURSE
- SESSION_NBR
- COURSE_TITLE
- COURSE_END_DT
- ATTENDANCE
- TRAINING_REASON
- COURSE_GRADE
- PREREQ_MET
- DEPTID
- BUSINESS_UNIT

XMLDoc method of creating an XML File

The XMLDoc object is PeopleTools utility to manipulate XML Files in their native format. One of these objects methods is to generate a formatted XML string from the current XMLDoc structure. The developer creates an XMLDoc structure and adds "Nodes" to the document to create tags and structure. The level the node is attached to determines the structure of the XML document.

Though this code may look similar to a freehand method, realize that the object is maintaining the structure. The developer is not concerned with end tags and completing parent child structures for the document. As each node is added, the value of that node is set.

When all the data has been moved to the XMLDoc object, use the "GenFormattedXmlString" method of the object to create the entire XML file in one string. Then write that string to file.

See appendix F for the result XML File.

```
import PSXP_RPTDEFNMANAGER:*;
import X_BI_PUB_PCODE:LoadTestData;

Local XmlDoc &inXMLDoc;
Local XmlNode &childNode, &textNode, &midNode, &rowNode;
Local File &XML_File;

Local X_BI_PUB_PCODE:LoadTestData &LoadTestData = create
X_BI_PUB_PCODE:LoadTestData();
&RS_Employee = &LoadTestData.LoadTestDataSet();

/* Create & Load XMLDoc */
&inXMLDoc = CreateXmlDoc("<?xml version='1.0'?><root/>");

For &i = 1 To &RS_Employee.ActiveRowCount
   &rowNode = &inXMLDoc.DocumentElement.AddElement("EMPLOYEE_DATA");
   &X_EE_RPT_LTR_VW_REC = &RS_Employee(&i).GetRecord(Record.X_EE_RPT_LTR_VW);

   For &f = 1 To &X_EE_RPT_LTR_VW_REC.FieldCount
      &childNode = &rowNode.AddElement(&X_EE_RPT_LTR_VW_REC.GetField(&f).Name);
      &childNode.NodeValue = String(&X_EE_RPT_LTR_VW_REC.GetField(&f).Value);
   End-For; /* record fields */

   &RS_Training = &RS_Employee(&i).GetRowset(Scroll.X_TRAINING_VW);
   For &j = 1 To &RS_Training.ActiveRowCount
      &X_TRAINING_VW_REC = &RS_Training(&j).GetRecord(Record.X_TRAINING_VW);
      &midNode = &rowNode.AddElement("TRAINING_DATA");

      For &f = 1 To &X_TRAINING_VW_REC.FieldCount
         &childNode = &midNode.AddElement(&X_TRAINING_VW_REC.GetField(&f).Name);
         &childNode.NodeValue = String(&X_TRAINING_VW_REC.GetField(&f).Value);
      End-For; /* record fields */
   End-For; /* training rowset */
End-For; /* employee rowset */

&Xml_String = &inXMLDoc.GenFormattedXmlString();

&XML_File = GetFile("XMLDOC.xml", "W", "UTF8");
&XML_File.WriteLine(&Xml_String);

/* save file name and path for publishing */
&XML_Filename_path = &XML_File.Name;
&XML_File.Close();
```

XML File Generation Considerations

Concurrency: Generating multiple versions of the same report simultaneously

PeopleSoft by its very nature may have multiple, different users attempting to create the same report at the same time. Ensure that your code accounts for this probability by making the file names unique per each instance of the report. One idea to accomplish this is to append the User or Employee ID to the file creating a unique file name.

Another method is to append the current date/time stamp to the filename to force uniqueness.

Schema Files

For PeopleTools version 8.48 & 8.49, BI Publisher requires XML Schema files for the correct mapping of tags to a PDF Template. You could include these for RTF (MSWord) templates, but they were not necessary.

The easiest way to generate correct schemas for the generated XML Files is to use an XML File editing utility such as XML Spy or XML Fox. XML Fox is a freeware utility that works very nicely. These utilities can generate schemas from your final XML Files.

Starting with PeopleTools 8.50, BI Publisher does not require a schema file for PDF Template mapping. Only a sample data file is required.

Sample Files for defining BI Publisher Data Sources.

You must provide a Sample XML Data file when creating a new BI Publisher report. A good idea is to execute a version of the XML File generation code using a sample or representative employee. Another good idea is to, for the sample run, modify your code to put the XML Tag names as values in the XML File. This will aid in mapping and formatting: do you have the right field in the right place on your report?

House Keeping

Depending on data policies of the implementations site, it may be a good idea to delete the BI Publisher Source files from the files directory after the report documents have been created. The easiest way to delete the file is to re-open the file after the BI Publisher code is complete, then use the “delete()” method instead of the “close()” method. This will remove the file from the servers.

Appendix A

Data set class for data source creation. This function populates and returns a complex RowSet object.

```
/*****************************************************/
/** GronkWare, Inc                                  **/
/** Randy Groncki   20168-04-16                      **/
/** BI Publisher                                    **/
/** XML File Generation Examples                    **/
/** Contact: Randy.Groncki@GronkWare.com            **/
/*****************************************************/

class LoadTestData
   method LoadTestData();
   method LoadTestDataSet() Returns Rowset;

private

   method LoadEmployeeImage(&Emplid As string) Returns string;

end-class;

method LoadTestData

end-method;

method LoadTestDataSet
   /+ Returns Rowset +/

   Local Rowset &RS_Training, &RS_Employees;
   Local Record &JOB_REC, &LOCATION_TBL_REC, &COMPANY_TBL_REC, &JOBCODE_TBL_REC;
   Local integer &i;

   /* create records */
   &JOB_REC = CreateRecord(Record.JOB);
   &LOCATION_TBL_REC = CreateRecord(Record.LOCATION_TBL);
   &COMPANY_TBL_REC = CreateRecord(Record.COMPANY_TBL);
   &JOBCODE_TBL_REC = CreateRecord(Record.JOBCODE_TBL);

   /* create rowsets */
   &RS_Training = CreateRowset(Record.X_TRAINING_VW); /* child rowset */
   &RS_Employees = CreateRowset(Record.X_EE_RPT_LTR_VW, &RS_Training); /* parent
rowset */

   /* Fill Parent */
   &RS_Employees.Fill("where emplid like 'KU00%' and exists (select 'x' from
ps_training t where t.emplid = fill.emplid)");

   /* Loop through parent rowset for processing on each row */
   For &i = 1 To &RS_Employees.ActiveRowCount

      /* Fill child rowset */
      &RS_Training = &RS_Employees(&i).GetRowset(Scroll.X_TRAINING_VW);
      &RS_Training.Fill("where emplid = :1",
&RS_Employees(&i).X_EE_RPT_LTR_VW.EMPLID.Value);

      /* Get job row for linking other data */
      &JOB_REC.EMPLID.Value = &RS_Employees(&i).X_EE_RPT_LTR_VW.EMPLID.Value;
      &JOB_REC.EMPL_RCD.Value = 0;
      /* get the current effdt & effseq for the EEs job row */
      SQLExec("select %dateout(j.effdt), j.effseq from ps_job j where j.emplid = :1
and j.empl_rcd = :2 and j.effdt = (select max(j2.effdt) from ps_job j2 where
j2.emplid = j.emplid and j2.empl_rcd = j.empl_rcd and j2.effdt <= %datein(:3)) and
j.effseq = (select max(j3.effseq) from ps_job j3 where j3.emplid = j.emplid and
j3.empl_rcd = j.empl_rcd and j3.effdt = j.effdt)", &JOB_REC.EMPLID.Value,
&JOB_REC.EMPL_RCD.Value, %Date, &JOB_REC.EFFDT.Value, &JOB_REC.EFFSEQ.Value);
      &JOB_REC.SelectByKey();
```

Data set function for data source creation (Continued)

```
      /* retrieve specific location data for description in report */
      &LOCATION_TBL_REC.SETID.Value = &JOB_REC.SETID_LOCATION.Value;
      &LOCATION_TBL_REC.LOCATION.Value = &JOB_REC.LOCATION.Value;
      &LOCATION_TBL_REC.SelectByKeyEffDt(%Date);
      &RS_Employees(&i).X_EE_RPT_LTR_VW.LOCATION_DESCR.Value =
&LOCATION_TBL_REC.DESCR.Value;

      /* retrieve specific company data for description in report */
      &COMPANY_TBL_REC.COMPANY.Value = &JOB_REC.COMPANY.Value;
      &COMPANY_TBL_REC.SelectByKeyEffDt(%Date);
      &RS_Employees(&i).X_EE_RPT_LTR_VW.COMPANY_DESCR.Value =
&COMPANY_TBL_REC.DESCR.Value;

      /* retrieve specific jobcode data for description in report */
      &JOBCODE_TBL_REC.SETID.Value = &JOB_REC.SETID_JOBCODE.Value;
      &JOBCODE_TBL_REC.JOBCODE.Value = &JOB_REC.JOBCODE.Value;
      &JOBCODE_TBL_REC.SelectByKeyEffDt(%Date);
      &RS_Employees(&i).X_EE_RPT_LTR_VW.JOB_DESCR.Value = &JOBCODE_TBL_REC.DESCR.Value;

      /* get employee image */
      &RS_Employees(&i).X_EE_RPT_LTR_VW.X_EMPLOYEE_PHOTO.Value =
%This.LoadEmployeeImage(&JOB_REC.EMPLID.Value);

   End-For;

   Return &RS_Employees;
end-method;

method LoadEmployeeImage
   /+ &Emplid as String +/
   /+ Returns String +/

   Local File &Image_File;
   Local string &Base64String, &NewFileName, &FQ_Filename_path;
   Local integer &retcode;

   &NewFileName = %UserId | %Datetime | ".jpg";
   &Image_File = GetFile(&NewFileName, "W");
   &FQ_Filename_path = &Image_File.Name;
   &Image_File.Close();

   &retcode = GetAttachment("record://X_EPHOTO_VW", &Emplid, &FQ_Filename_path);

   If &retcode < 2 Then
      &Image_File = GetFile(&FQ_Filename_path, "R", %FilePath_Absolute);
      &Base64String = &Image_File.GetBase64StringFromBinary();
      &Image_File.Close();
   End-If;

   /* delete file */
   &Image_File = GetFile(&FQ_Filename_path, "R", %FilePath_Absolute);
   &Image_File.Delete();

   Return &Base64String;

end-method;
```

Appendix B

XMLDoc XML File Example

```
<?xml version="1.0"?>
<root>
        <EMPLOYEE_DATA>
                <EMPLID>KU0001</EMPLID>
                <NAME>Douglas Lewis</NAME>
                <DEPTID>ADMIN</DEPTID>
                <LOCATION>KUNY00</LOCATION>
                <JOBCODE>700005</JOBCODE>
                <COMPRATE>21666.666667</COMPRATE>
                <CHANGE_PCT>0</CHANGE_PCT>
                <LOCATION_DESCR>Corporation Headquarters</LOCATION_DESCR>
                <JOB_DESCR>President & CEO</JOB_DESCR>
                <COMPANY_DESCR>Global Business Institute</COMPANY_DESCR>
                <TRAINING_DATA>
                        <EMPLID>KU0001</EMPLID>
                        <COURSE_START_DT>2005-01-13</COURSE_START_DT>
                        <COURSE>M2005</COURSE>
                        <SESSION_NBR>0001</SESSION_NBR>
                        <COURSE_TITLE>course with 2000 hours</COURSE_TITLE>
                        <COURSE_END_DT/>
                        <ATTENDANCE>E</ATTENDANCE>
                        <TRAINING_REASON/>
                        <COURSE_GRADE/>
                        <PREREQ_MET>N</PREREQ_MET
                        ><DEPTID>ADMIN</DEPTID>
                        <BUSINESS_UNIT>GBIBU</BUSINESS_UNIT>
                </TRAINING_DATA>
        </EMPLOYEE_DATA>
</root>
```