m_utl_xml2ds.sas File Reference

Utilities

Utility macro to import a XML file into SAS datasets or tables

---

Description

This program converts an extensible markup file with the XML extension into one or more SAS datasets or database tables. The program uses the XML or XMLV2 libname engine and works much like any other SAS engines. A libname statement is executed to assign a libref named MyXML and specified by the XML or XMLV2 engine. The XMLV2 engine is available since SAS 9.2 on all supported operating environments. For previous SAS versions and/or using MSACCESS as xmltype or providing a xmlmap file, the basic XML engine is used for importing the xml file. In case of using SAS 9.2 or later in combination with AUTOMAP=Y, the XMLV2 engine is used to import the XML file into SAS using an autogenerated xmlmap. This method is able to import nested XML file elements and results to importing into more than one SAS datasets or database tables.

Note

This macro does not support the import of concatenated xml documents since they are not part of the XML standard.

Autors

Paul Alexander Canals y Trocha (paul.canals@gmail.com)

Date

2020-09-07 00:00:00

Version

20.1.09

Link

https://github.com/paul-canals/toolbox

Parameters

Input	help	Parameter, if set (or ?) to print the Help information in the log. In all other cases this parameter should be left out from the macro call.
Input	in_file	Specifies the name and location of the extensible markup language file with a XML extension. The default value is: _NONE_.
Input	out_ds	Specifies the full LIBNAME.TABLENAME name of the output SAS dataset or database table. The default value is: _NONE_.
Input	engine	Indicator [XML|XMLV2] parameter to specify the engine that is to be used to import the XML file. The default value is: XMLV2.
Input	xmltype	Indicator [GENERIC|MSACCESS|XMLMAP] to specify the XML markup type. When set to GENERIC, the XML file consists of a root (enclosing) element and repeating element instances. It will determine the variable attributes from the data content. When set to MSACCESS, then the XML markup supports a Microsoft Access database. If the Microsoft Access file contains metadata-related information, then the XMLMETA=SCHEMADATA option is to be set to obtain variable attributes from the embedded schema. If there is not an embedded schema, MSACCESS uses default values for attributes. When set to XMLMAP, the XML markup is determined by an XMLMap, which is an XML document that contains specific XMLMap syntax. The XMLMap syntax tells the XML engine how to map the specific XML document structure back into SAS data. To specify the XMLMap in the LIBNAME statement, use the XMLMAP= option. The default value is: GENERIC.
Input	xmlmeta	Indicator [DATA|SCHEMA|SCHEMADATA] to specify whether to import metadata-related information that is included in the input XML document. The Metadata- related information is metadata that describes the characteristics (types, lengths, levels, and so on) of columns within the table markup. When set to DATA, the metadata-related information is ignored and only data content is imported from the input XML document. When set to SCHEMA, data content is ignored and only metadata-related information is included from the the input XML document. When set to SCHEMADATA, both data content and metadata-related information is imported from the input XML document. The default value is: SCHEMADATA.
Input	xmlmap	Optional parameter to specify name and location of the file that contains specific XMLMap syntax with an MAP extension. The syntax tells the XML engine how to interpret the XML markup for importing. The XMLMap syntax itself is XML markup.
Input	automap	Boolean [Y|N] to decide whether to automatically generate an XMLMap file to import an XML document. The XMLMap document contains specific syntax that describes how to interpret the XML markup into a SAS dataset or database table, variables (columns), and observations (rows). XMLMap syntax is generated by analyzing the structure of the specified XML file. The default value is: N.
Input	xmlencoding	Optional parameter to override the default SAS dataset encoding for the input XML file. If an encoding value contains a hyphen, enclose the value in quotation marks. Use this option with caution. If you are unfamiliar with character sets, encoding methods, or translation tables, do not use this option without proper technical advice.
Input	debug	Boolean [Y|N] parameter to provide verbose mode information. The default value is: N.

Returns

- Output SAS dataset or database table.

Calls

- m_utl_get_sashelp.sas
- m_utl_print_message.sas
- m_utl_print_mtrace.sas

Usage

Example 1: Show help information:

```
%m_utl_xml2ds(?)
```

Example 2: Step 1 - Export a SAS dataset into a generic XML file:

```
libname MyXML xml "%sysfunc(getoption(WORK))/class.xml"
   xmlmeta=schemadata
   ;

data MyXML.class;
   set SASHELP.class;
run;

libname MyXML clear;
```

Example 2: Step 2 - Import the XML file back into a SAS dataset:

```
%m_utl_xml2ds(
   in_file = %str(%sysfunc(getoption(WORK))/class.xml)
 , out_ds  = WORK.class
 , debug   = Y
   );

proc print data=WORK.class;
run;
```

Example 3: Step 1 - Export a SAS dataset into a Ms. Access type XML file:

```
libname MyXML xml "%sysfunc(getoption(WORK))/class.xml"
   xmlmeta=schemadata
   xmltype=msaccess
   ;

data MyXML.class;
   set SASHELP.class;
run;

libname MyXML clear;
```

Example 3: Step 2 - Import the Ms. Access type XML file into a SAS dataset:

```
%m_utl_xml2ds(
   in_file = %str(%sysfunc(getoption(WORK))/class.xml)
 , out_ds  = WORK.class
 , xmltype = msaccess
 , debug   = Y
   );

proc print data=WORK.class;
run;
```

Example 4: Step 1 - Export a SAS dataset into a Ms. Access type XML file:

```
libname MyXML xml "%sysfunc(getoption(WORK))/class.xml"
   xmlmeta=schemadata
   xmltype=msaccess
   ;

data MyXML.class;
   set SASHELP.class;
run;

libname MyXML clear;
```

Example 4: Step 2 - Import the XML file into a SAS dataset by automap:

```
%m_utl_xml2ds(
   in_file = %str(%sysfunc(getoption(WORK))/class.xml)
 , out_ds  = WORK.class
 , automap = Y
 , debug   = Y
   );

proc print data=WORK.class;
run;
```

Example 5: Step 1 - Created a XML file with nested elements:

```
filename class "%sysfunc(getoption(WORK))/class.xml";

data _null_;
   file class;
   put '<?xml version="1.0" encoding="utf-8" ?>';
   put '<root>';
   put '   <dataroot>';
   put '      <class>';
   put '         <Name>Alfred</Name>';
   put '         <Sex>M</Sex>';
   put '         <Age>14</Age>';
   put '         <Height>69</Height>';
   put '         <Weight>112.5</Weight>';
   put '         <classfit>';
   put '            <predict>126.00617011</predict>';
   put '            <lowermean>116.94168423</lowermean>';
   put '            <uppermean>135.07065599</uppermean>';
   put '            <lower>100.64558742</lower>';
   put '            <upper>151.36675279</upper>';
   put '         </classfit>';
   put '      </class>';
   put '      <class>';
   put '         <Name>Alice</Name>';
   put '         <Sex>F</Sex>';
   put '         <Age>13</Age>';
   put '         <Height>56.5</Height>';
   put '         <Weight>84</Weight>';
   put '         <classfit>';
   put '            <predict>77.268291747</predict>';
   put '            <lowermean>68.906553333</lowermean>';
   put '            <uppermean>85.630030161</uppermean>';
   put '            <lower>52.150311745</lower>';
   put '            <upper>102.38627175</upper>';
   put '         </classfit>';
   put '      </class>';
   put '      <class>';
   put '         <Name>Barbara</Name>';
   put '         <Sex>F</Sex>';
   put '         <Age>13</Age>';
   put '         <Height>65.3</Height>';
   put '         <Weight>98</Weight>';
   put '         <classfit>';
   put '            <predict>111.57975811</predict>';
   put '            <lowermean>105.26025322</lowermean>';
   put '            <uppermean>117.89926301</uppermean>';
   put '            <lower>87.06587649</lower>';
   put '            <upper>136.09363973</upper>';
   put '         </classfit>';
   put '      </class>';
   put '  </dataroot>';
   put '</root>';
run;

data _null_;
   infile class;
   input;
   put _infile_;
run;
```

Example 5: Step 2 - Import the nested XML file into SAS datasets by automap:

```
%m_utl_xml2ds(
   in_file = %str(%sysfunc(getoption(WORK))/class.xml)
 , out_ds  = WORK.class
 , automap = Y
 , debug   = Y
   );

proc print data=WORK.class_root noobs;
run;

proc print data=WORK.class_dataroot noobs;
run;

proc print data=WORK.class noobs;
run;

proc print data=WORK.class_classfit noobs;
run;
```

Example 5: Step 3 - Join the SAS datasets from Step 2 into one result dataset:

```
proc sql noprint;
   create table WORK.examined as
   select c.Name
        , c.Sex
        , c.Age
        , c.Height
        , c.Weight
        , d.predict
        , d.lowermean
        , d.uppermean
        , d.lower
        , d.upper
     from WORK.class_root as a
        , WORK.class_dataroot as b
        , WORK.class as c
        , WORK.class_classfit as d
    where a.root_ORDINAL = b.root_ORDINAL
      and b.dataroot_ORDINAL = c.dataroot_ORDINAL
      and c.class_ORDINAL = d.class_ORDINAL
    ;
quit;

proc print data=WORK.examined noobs;
run;
```

Example 6: Step 1 - Created a XML file with structure and some data:

```
filename nhl_xml "%sysfunc(getoption(WORK))/nhl.xml";

data _null_;
   file nhl_xml;
   put '<?xml version="1.0" encoding="iso-8859-1" ?>';
   put '<NHL>';
   put '   <CONFERENCE> Eastern';
   put '      <DIVISION> Southeast';
   put '         <TEAM name="Thrashers"  abbrev="ATL" />';
   put '         <TEAM name="Hurricanes" abbrev="CAR" />';
   put '         <TEAM name="Panthers"   abbrev="FLA" />';
   put '         <TEAM name="Lightning"  abbrev="TB"  />';
   put '         <TEAM name="Capitals"   abbrev="WSH" />';
   put '      </DIVISION>';
   put '   </CONFERENCE>';
   put '   <CONFERENCE> Western';
   put '      <DIVISION> Pacific';
   put '         <TEAM name="Stars"   abbrev="DAL" />';
   put '         <TEAM name="Kings"   abbrev="LA"  />';
   put '         <TEAM name="Ducks"   abbrev="ANA" />';
   put '         <TEAM name="Coyotes" abbrev="PHX" />';
   put '         <TEAM name="Sharks"  abbrev="SJ"  />';
   put '      </DIVISION>';
   put '   </CONFERENCE>';
   put '</NHL>';
run;

data _null_;
   infile nhl_xml;
   input;
   put _infile_;
run;

filename nhl_xml clear;
```

Example 6: Step 2 - Created a XML map file using XML map v1.0 format:

```
filename nhl_map "%sysfunc(getoption(WORK))/nhl.map";

data _null_;
   file nhl_map;
   put '<?xml version="1.0" ?>';
   put '<SXLEMAP version="1.2">';
   put '   <TABLE name="TEAMS">';
   put '      <TABLE-PATH syntax="XPATH">/NHL/CONFERENCE/DIVISION/TEAM</TABLE-PATH>';
   put '         <COLUMN name="NAME">';
   put '            <PATH>/NHL/CONFERENCE/DIVISION/TEAM/@name</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>30</LENGTH>';
   put '         </COLUMN>';
   put '         <COLUMN name="ABBREV">';
   put '            <PATH>/NHL/CONFERENCE/DIVISION/TEAM/@abbrev</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>3</LENGTH>';
   put '         </COLUMN>';
   put '         <COLUMN name="CONFERENCE" retain="YES">';
   put '            <PATH>/NHL/CONFERENCE</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>10</LENGTH>';
   put '         </COLUMN>';
   put '         <COLUMN name="DIVISION" retain="YES">';
   put '            <PATH>/NHL/CONFERENCE/DIVISION</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>10</LENGTH>';
   put '         </COLUMN>';
   put '   </TABLE>';
   put '</SXLEMAP>';
run;

data _null_;
   infile nhl_map;
   input;
   put _infile_;
run;

filename nhl_map clear;
```

Example 6: Step 3 - Import the XML file into a SAS dataset by XML map:

```
%m_utl_xml2ds(
    in_file = %sysfunc(getoption(WORK))/nhl.xml
  , out_ds  = WORK.teams
  , engine  = XML
  , xmltype = XMLMAP
  , xmlmap  = %sysfunc(getoption(WORK))/nhl.map
  , debug   = Y
    );

proc print data=WORK.teams noobs;
run;
```

Copyright