XMLutils Documentation

November 17, 2007

Contents

1 What is XMLutils? 1

2 XPath expressions 1

3 Available functions 2

3.1 XMLopenfile("filenamestr") 2
3.2 XMLclosefile(fileID, toSave) 2
3.3 XMLsavefile(fileID) 2
3.4 XMLelemlist(fileID) 3
3.5 XMLstrFmXpath(fileID,"xpathStr","namespacestr","optionsStr") . 3
3.6 XMLwaveFmXpath(fileID,"xpathStr","namespaceStr","optionsStr") 4
3.7 XMLsetNodeStr(fileID,"xpathStr","namespaceStr","contentStr") . 5
3.8 XMLlistAttr(fileID,"xpathStr","nsStr") 6
3.9 XMLsetAttr(fileID,"xpathStr","nsStr","attNameStr","valStr") . . . 7

4 Licencing 7

1 What is XMLutils?

At the moment XMLutils is an exploration of how to read (and in the future write) data from XML files via an XOP. There is no guarantee that any of the functionality that currently exists will remain in the same form as the package evolves. This is because we are constantly learning. This will make the XOP an pre-alpha release. It uses the libxml2 library, `http://xmlsoft.org`

2 XPath expressions

A description of XPath is outside the scope of this documentation, partially because the authors don't fully understand it. For a tutorial please see `http://www.w3schools.com/xpath/default.asp`. The XPath expressions are evaluated to create a nodeset. The nodes in this nodeset are then output in different ways, e.g. as a string, or in a wave. Xpath expressions are quite powerful and can be used to extract data from many places in the document. As I said, look at the tutorial.

3 Available functions

3.1 XMLopenfile("filenamestr")

Arguments

- FILENAMESTR - a string containing the path to the XML file of interest.

Usage

```
variable fileID = ("/Users/andrew/air_h2o.xml")
```

Output returns a variable which is a reference to the XML file held in memory, each time you want to access the XML file you must use this FILEID variable (it is an integer from >=1).

3.2 XMLclosefile(fileID, toSave)

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*). If FILEID = -1 then all the currently xml files are closed, and optionally saved.
- TOSAVE - whether fileID gets saved (toSave = 1) or not (toSave = 0).

Usage

```
//doesn't save
xmlclosefile(fileID,0)
//does save
xmlclosefile(fileID,1)
```

Output The XML files referred to by fileID is optionally saved then closed. If fileID=-1 then all currently openfiles are closed.

3.3 XMLsavefile(fileID)

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*).

Usage

```
xmlsavefile(fileID)
```

Output The XML files referred to by fileID is saved.

3.4 XMLelemlist(fileID)

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*).

Usage

```
xmlelemlist(fileID)
```

Output W_ELEMENTLIST - a textwave with 3 columns. The first column contains the full path of each of the nodes in the XML file. The second column contains the namespace associated with each of the element nodes. The third column contains the namespace prefix associated with each namespace.

3.5 XMLstrFmXpath(fileID,"xpathStr","namespacestr","optionsStr")

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*).
- XPATHSTR - a string with the XPath to be evaluated
- NAMESPACESTR - a string for registering prefixes and namespaces for use with the XPath expression. This should take the form *"prefix1=namespace1 prefix2=namespace2"* (i.e. separated by whitespace).
- OPTIONSSTR - a string to pass options to the function.

Usage These commands extract the intensities node from the air_h2O.xml file, as a string:

```
//first open the file and get a reference
variable fileID = xmlopenfile("/users/Andrew/air_h2O.xml")
//now get the XMLstring using the fileID
print XMLstrfmXpath(fileID, "//xrdml:intensities/text()",
"xrdml=http://www.xrdml.com/XRDMeasurement/1.0",
"" )
```

The xpathStr "*//xrdml:intensities/*text()" means get the content of all text nodes in the *intensities* element node from the *xrdml* namespace. *Please note, all XPath expressions are case-sensitive.*

The namespaceStr "*xrdml=http://www.xrdml.com/XRDMeasurement/1.0*" registers the *http://www.xrdml.com/XRDMeasurement/1.0* namespace with the *xrdml* prefix. If we had used a different prefix, we would have to have used the same one in the xpathStr. Some XML files (nodes) do not have namespaces associated with them, so there is no need to register a namespace, in which case you can use "" as the namespaceStr. If you want to know the namespace associated with a specific node then use the *xmlelementlist()* function (in XML each node can have a different namespace).

If there is more than 1 node that matches the xpathStr expression, then the content of that node is appended to the string, for example:

```
variable fileID = xmlopenfile("/Users/andrew/cd_catalog.xml")
//assumes you have already loaded catalog_CD.xml into fileID
print xmlstrfmxpath(fileID,
"//TITLE/text()","","")
```

results in:

Empire Burlesque Hide your heart Greatest Hits Still got the blues Eros One night only Sylvias Mother Maggie May Romanza When a man loves a woman Black angel 1999 Grammy Nominees For the good times Bi g Willie style Tupelo Honey Soulsville The very best of Stop Bridge of Spies Private Dancer Midt om natten Pavarotti Gala Concert The dock of the bay Picture book Red Unchain my heart

If you only wanted the first title you could've used:

```
print xmlstrfmxpath(fileID,
"//CD[1]/TITLE/text()","","")
```

If you wanted the all content of all the nodes you could use:

```
print xmlstrfmxpath(fileID,
"//CD/*/text()","","")
```

All of the content of the 1st node:

```
print xmlstrfmxpath(fileID,
"//CD[1]/*/text()","","")
```

Output A string containing the output is returned.

3.6 XMLwaveFmXpath(fileID,"xpathStr","namespaceStr","optionsStr")

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*).
- XPATHSTR - a string with the XPath to be evaluated
- NAMESPACESTR - a string for registering prefixes and namespaces for use with the XPath expression. This should take the form *"prefix1=namespace1 prefix2=namespace2"* (i.e. separated by whitespace).
- OPTIONSSTR - a string to pass options to the function.

Usage See the XMLstrFmXpath function for more useage details, they are exactly the same, except for the name of the function.

Output

M_xmlcontent - a 2D matrix textwave. Please see the XMLstrFmXpath for it's usage and output. This function simply parses that output into tokens (using whitespace as a separator). Data from different nodes in the file are put into successive columns in the wave. For example if the XPath expression matches 4 different nodes, then there will be 4 columns. The total number of rows will be equal to number of tokens from the element with the biggest content. Creating a numeric wave from this is simple:
make/n=(dimsize(M_xmlcontent,0),dimsize(M_xmlcontent,1)) numbers = str2num(M_xmlcontent)
It is unlikely that all the nodes will have the same number of tokens, so there will be blank entries at the end that will transfer to NaN if you use *str2num*.

W_xmlcontentnodes - a 1D textwave containing the path to each of the nodes that match the XPath expression.

3.7 XMLsetNodeStr(fileID,"xpathStr","namespaceStr","contentStr")

Arguments

- fileID - the fileID for the XML file (see *xmlopenfile*).
- xpathStr - a string with the XPath to be evaluated
- namespaceStr - a string for registering prefixes and namespaces for use with the XPath expression. This should take the form *"prefix1=namespace1 prefix2=namespace2"* (i.e. separated by whitespace).
- contentStr - a string for setting the contents of a text node.

Usage All the nodes in the file that match the XPath expression are found (there may be more than 1) and their contents replaces by *contentStr*. This can be a dangerous operation if you don't set the Xpath correctly.

For example.
<book>
<title>wally</title>
<cost>10</cost>
</book>

Say you wanted to put some text directly after <book>, you would have to use the XPath *//book/text()[1]*, which would give:
<book>
textStr

<title>wally</title>
<cost>10</cost>
</book>
If you used *//book/text()* you would get:
<book>
textStr
<title>wally</title>
textStr
<cost>10</cost>
textStr
</book>
Because there are children textnodes of book at each of those places. However, the most dangerous operation would be to use *//book*, this would result in:
<book>
textStr
</book>

Output The XML file is changed. However, it is not saved until *XMLsavefile* is used.

3.8 XMLlistAttr(fileID,"xpathStr","nsStr")

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*).
- XPATHSTR - a string with the XPath to be evaluated
- NSSTR - a string for registering prefixes and namespaces for use with the XPath expression. This should take the form *"prefix1=namespace1 prefix2=namespace2"* (i.e. separated by whitespace).

Usage All the nodes in the file that match the XPath expression are found (there may be more than 1) and their attributes and values found and listed.
For example:

```
XMLlistAttr(fileID,"//xrdml:kAlpha1",
"xrdml=http://www.xrdml.com/XRDMeasurement/1.0")
```

Output A three column text wave called M_LISTATTR is produced. The first column contains the path structure to each of the nodes matched by the XPath expression, the second contains the attribute name, the third contains the value of the attribute.

3.9 XMLsetAttr(fileID,"xpathStr","nsStr","attNameStr","valStr")

Arguments

- FILEID - the fileID for the XML file (see *xmlopenfile*).
- XPATHSTR - a string with the XPath to be evaluated
- NSSTR - a string for registering prefixes and namespaces for use with the XPath expression. This should take the form *"prefix1=namespace1 prefix2=namespace2"* (i.e. separated by whitespace).
- ATTNAMESTR - a string that contains the name of the attribute you would like to change.
- VALSTR - a string containing the new value of the attribute

Usage All the nodes in the file that match the XPath expression are found (there may be more than 1), all the nodes in this nodeset that have the attribute name *attributeNameStr* have their value changed to *valueStr*.

For example:

```
XMLsetAttr(fileID,"//xrdml:kAlpha1",
"xrdml=http://www.xrdml.com/XRDMeasurement/1.0","foo","bar")
```

will create the attribute *foo* which has the value *bar* in the *kAlpha1* element node, e.g.

```
<kAlpha1 foo=''bar''/>
```

If the attribute doesn't exist, it is created.

Output If successful *fileID* is changed, but is not saved until xmlsavefile is used.

4 Licencing

This package uses the libxml2 library from http://xmlsoft.org/ `http://xmlsoft.org/`, which is written by Daniel Veillard. The libxml2 package is released under the MIT licence: