

Use Saxon to query details from FileMaker's Save as XML files

Did you know that you can use our [Saxon](#) functions to run XPath Queries against the XML file that FileMaker wrote with the Save as XML script step?

Use the menu command or write yourself a script to run every night to use Save as XML... in the Tools menu and write a new XML file.

Query BaseTables

So let's start with the xml file and it starts like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<FMSaveAsXML version="2.2.3.2" Source="22.1.1" File="Assets.fmp12"
  UUID="BAFECB9B-E017-4706-977D-7262EEE9E358" locale="English" Has_DDR_INFO="False">
```

You could run a simple query to list the
`string-join(//BaseTable/@name, '')`

This may give you a list of the base tables, e.g.:

Assets
Vendors
Employees
Assignments

You may notice that we put in the return directly in the call. To make this work in a calculation, we can use the ¶ symbol:

```
Set Field [ Saxon XSLT::Result ;
  MBS( "Saxon.XPathQuery"; Saxon XSLT::XML; "string-join(//BaseTable/@name, '¶')" ) ]
```

Find Table Occurrences

Next we may find all the table occurrences and sum them up as a list with arrows showing the name of the underlying base table. Here is the XPathQuery to do this:

```
for $to in //TableOccurrence
return concat(
  $to/@name, ' -> ',
  //BaseTable[@id=$to/BaseTableSourceReference/
BaseTableReference/@id]/@name
)
```

As you see this is basically a for each loop going over the list of TableOccurrence nodes found in the XML. For each entry found, it concatenates the name of the table with the looked up base table name. As you see we match the base tables by the id given in the TableOccurrence definition. Saxon executes this lookup as basically another loop to go over all the BaseTableReference objects and figuring out which one has the matching identifier.

This may show a list of texts like this:

```
"Assets -> Assets"
"Vendors -> Vendors"
"Employees -> Employees"
"Assignments -> Assignments"
```

We may use the string-join function again to make it one string:

```
string-join(for $to in //TableOccurrence
return concat(
  $to/@name, ' -> ',
  //BaseTable[@id=$to/BaseTableSourceReference/
BaseTableReference/@id]/@name
), '
')
```

or all in one line as a calculation:

```
MBS( "Saxon.XPathQuery"; Saxon XSLT::XML;
  "string-join(for $to in //TableOccurrence return
concat($to/@name, ' -> ',
  //BaseTable[@id=$to/BaseTableSourceReference/
BaseTableReference/@id]/@name), '¶')" )
```

This is all one line. We wrapped it for the blog post.

List fields

Let's check the field catalog and find all the fields with their name and ids. The `//FieldsForTables/FieldCatalog/ObjectList/Field` query finds all Field objects. The long query makes sure we don't get the Field entries of a different kind inside the layouts. Once we have a field, we can follow the structure up to find the BaseTableReference and get the table id there. This id can then be used to match the base table and get the name.

```
string-join(
  for $f in //FieldsForTables/FieldCatalog/ObjectList/Field
  return concat(
    $f/@name,
    ' with id ',
    $f/@id, ' is inside table ', //BaseTable[@id=$f/../../
BaseTableReference/@id]/@name, ' with id ', $f/../../
BaseTableReference/@id
  ),
  '
')
```

I know that you could use `"$f/../../BaseTableReference/@name"` directly to also get the name, but I like to show how to reference to another entity here.

The output looks like this:

```
Name with id 1 is inside table Assets with id 130
Description with id 2 is inside table Assets with id 130
...
```

Find scripts

You may apply the query from above to get the script names:

```
string-join(//ScriptCatalog/Script[not(@isFolder)]/@name, '')
```

or as script:

```
Set Field [ Saxon XSLT::Result ; MBS( "Saxon.XPathQuery"; Saxon XSLT::XML; "string-join(//ScriptCatalog/Script[not(@isFolder)]/@name, '¶')" ) ]
```

As you see, we check the isFolder attribute to make sure we skip folders.

Or you like folders? Then you can find them by looking for scripts that have the isFolder attribute:

```
string-join(//ScriptCatalog/Script[@isFolder]/@name, '')
```

You may find plenty of more things to do with our [Saxon](#) functions in FileMaker.

On the end this is similar to what tools like [Cross Check](#) or [Inspector Pro](#) do. They run lots of XSLT transformations on the XML from the Database Design Reports or Save as XML.