18. Extracting and Using XBRL Information

Ultimately, the entire reason for going through the effort of tagging or structuring financial information is so that the information can be reused.

XBRL can facilitate this, cost effectively. That is why XBRL is being adopted. And not only can the information be reused, it can be automatically reused, meaning that a computer, on its own, can be fed the information and the computer application can do what it needs with the information. This is as opposed to requiring humans to intervene and do something before the information can be reused.

Now, this is not to say that you want computers making all the decisions, but they can make some.

In this next section we will briefly look into extracting information from an XBRL instance document for reuse. The focus of this book is to create XBRL, but we did want to touch on extracting so the creators of XBRL instance documents will understand a bit about the step of the process after their step.

18.1. Overview

Extracting specific pieces of data from XBRL instance documents is fairly straight forward and easy. The reason for this ease is because this is what XBRL was built to achieve.

XBRL gave up a few things to get here. One of the things that it gave up was the content model, or rather XBRL has a very shallow content model. This makes querying and extracting XBRL a bit easier.

18.2. Comparability of Information

Many who extract information from XBRL instance documents will desire to compare what they extract with other information from other XBRL instance documents using automated processes.

XBRL has the features and functionality to achieve automated reuse of instance information, including comparison with other XBRL information. Comparability is less a technical issue, and more a domain and human issue. Comparability can be achieved at many different levels. The level of comparison will be decided by the creators and consumers of instance information.

There are many obstacles which must be overcome to compare financial information. Whether they are overcome will depend on the desires of users and consumers of information to overcome the obstacles.

18.3. Extracting Using XML versus Using XBRL Processor

XML processors have no knowledge of XBRL beyond what is in the XML Schema document. The primary reason for the XML Schema document is to enforce the validity of the structure of the XML.

Property structured XML is necessary for extracting XBRL data, but it is not sufficient. XBRL has additional functionality built in which XML parsers do not understand. The XBRL conformance suite has over 400 tests, each which an XBRL processor must "pass" an property interpret XBRL functionality specified in the XBRL specification.

The following is a brief summary of XBRL extraction issues which might trip up an XML parser-type approach to extracting and using XBRL information:

- What do you do when you run into two XBRL concepts in an instance document which are exactly the same? Same concept, same concept, same units, same decimals, and either the exact same value or maybe a different value? The XBRL specification explains what to do. FRIS forbids duplicate items and tuples. In financial reporting, duplicate items and duplicate tuples make no sense. XBRL processors which support FRIS will detect this.
- How does an XML parser you know if the data in an XBRL instance document properly adds up per the calculation linkbase of a taxonomy? Well, it could, but basically, the XML parser would have to have all the functionality to do this built on top of the XML parser. XBRL processors are required to know how to do this in order to be deemed a fully compliant XBRL processor.

The differences between an XML parser and XBRL processor are discussed in the comparison of XML and XBRL earlier in this document.

The bottom line is that XBRL processors understand XBRL, all of XBRL. They exist for the specific task of working with XBRL, that is all they do. Using an XML parser to extract information is very possible, and very useful in many cases; but in other cases, an XBRL processor will probably be required.

18.4. Example of Extracting Data

In this section we will look at the process of grabbing one piece of XBRL instance document data using an XML parser. The purpose is to briefly discuss how to do it and point out a few things to watch for. The purpose of this is to help those who desire to extract XBRL information into, say, an Excel model used for analysis.

This is by no means a comprehensive explanation of extracting information, you would, again, need to re-write all the code needed by an XBRL processor to cover everything. This is only to get you started.

18.4.1. Data to be Extracted

The following is one fact value from an XBRL instance document in the "Extract" files included on the CD:

```
<ci:Land contextRef="I-2003" unitRef="U-Monetary" decimals="INF">5347000</ci:Land>
```

The following is a simple XPath statement which extracts a single piece of data from an XBRL Instance document:

```
<xsl:value-of select="/xbrli:xbrl/ifrs-gp:Land[@contextRef='I-2003']" />
```

XML processors have no knowledge of XBRL beyond what is in the XML Schema document. The primary reason for the XML Schema document is to enforce the validity of the structure of the XML.

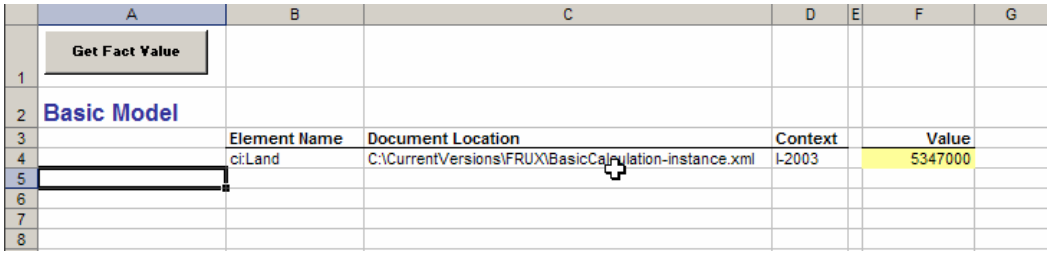
What we are going to do is the following:

1. Build a basic Excel model into which we desire to automatically enter XBRL data,
2. Write a brief Excel VBA macro to go get that data.
3. Populate the model using the code.

Realize that this is pretty basic, but this is really for those who are curious about how this extraction process works, not for developers who wish to implement XBRL applications. This is intentionally pretty basic.

18.4.2. Excel Model

This is a screen shot of our Excel model into which the data from the XBRL instance document will be populated:



Again, this is a very, very simple model which tries to focus on showing someone how to extract one piece of data from an XBRL instance document. Extracting more items should then be easy for someone who understands a little about writing VBA macros and creating "Do" loops or other mechanisms to extract more than one fact value.

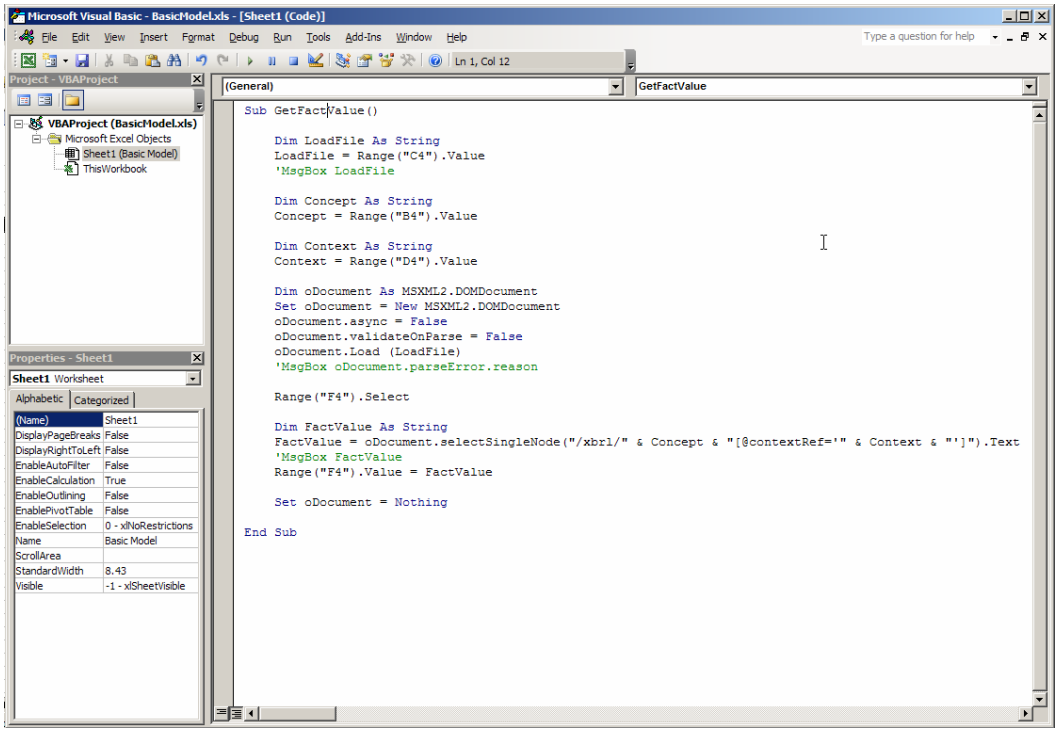
Using this macro is very simple:

1. Open the Excel spreadsheet.
2. Press the button "Get Fact Value".
3. Change the "Element Name", the "Document Location" and/or the "Context".
4. Press the button again.

Believe it or not, this code can be used to extract ANY fact value from ANY XBRL instance document ANYWHERE IN THE WORLD! Of course, the document has to be available either on your hard drive or on the Internet.

18.4.3. Code to Perform Extraction

The following is a screen shot from the Microsoft Visual Basic editor opened from Excel. The code you see is all that you need to extract fact values from an XBRL instance document:



We will now examine the code. Note that this code does not have any error checking built into it, the goal is clarity of understanding the code, not writing the most efficient code which will detect all possible errors. Clarity of the big picture is what we are after here.

The first thing you need to do in order to use an XML parser is to reference the XML parser API. To do this (we are assuming you have Microsoft Internet Explorer 6.0 installed) from within the VBA Editor select "Tools", "References". Then scroll down the list and check "Microsoft XML, v4.0" from the list.

The following explains the code:

- First we define a few variables to hold the name of the document to load, concept to go look for, context of the concept to go look for.
- We then load the XML/XBRL document.
- We then use a function of the XML parser to select the value of a specific XML element, we pass the XPath to that element.
- We grab the data and populate the Excel spreadsheet.

That is all there is to it. However, there is a lot that needs to be considered, we will discuss this in the next section.

18.4.4. Considerations

Although extracting data, as you can see, is quite simple, there is a lot to be considered if you are building complex applications (or even simple macros) to extract and use fact values from an XBRL document.

- Realize that you have grabbed the data, for example "ci:Land" from an instance document, but what exactly does that data mean? You either have to know what it means, or have the taxonomy.

- The contextRef has been used to grab the data, which is fine if you know what the contextRef will be. If not, you have to go read the actual context information.
- Unless you write additional code you are not sure if the instance documents is valid per the XBRL specification, FRTA, FRIS, or if the calculations are in fact valid. You can use the data at your own risk, but you also may want to be sure all of these things are in fact valid prior to using the data, it is up to the user of the data.
- XBRL processors do a lot of additional work, far, far more than can be covered here. Realize what you gain or loose by using or not using an XBRL processor. Remember, ignorance is bliss!

18.4.5. Summary

So in conclusion, extracting XBRL data is easy, but to actually use the data in an automated process, you have to really understand that what you are getting is correct, or you have to validate it prior to using it.

18.5. XQuery

XQuery is another part of the XML family. XQuery has its pros and cons, but we wanted to point it out as a potentially useful tool for extracting and using XBRL data. It has its places.

18.6. Relational Databases

Lots of people know how to use relational databases. SQL has been a round quite some time. XBRL was built to be easily loaded into an SQL database, again, the content model is fairly flat. Particularly if the taxonomy has only items in it (no tuples), then using relational database to store XBRL data is child's play.

Data may, or may not, exist in an XBRL format for long. Many, many sophisticated analysis applications exist. What may occur is simply using XBRL to transport data from creator to consumer, then the data is transformed into an in-house format, such as a relational database. From there, how to use the data is very understandable. But, you still have to consider the data validation issues: is the data correct. XBRL has features for checking the data's validity. If you remove the XBRL format, you loose these features.

18.7. Summary

Although the basic process of extracting data from an XBRL instance document, there is really a lot to consider if you want to use the data, particularly if you want to use the data in an automated process.

