XML reader models: SAX versus XML pull parser

In your experience with XML, you might have heard of SAX (the Simple API for XML), and you might have heard the term "XML pull parser." These are both ways to parse large XML files without using a lot of memory resources. They don't let you modify or generate XML; these "XML readers" let you read through a large XML file efficiently, using a small amount of memory.

The two models of read-only XML parser

The difference is simple: SAX uses callbacks and pull parsers don't. The two models are also described as being push versus pull:

- **SAX model:** after you set up callbacks, the SAX component *pushes* document events to you
- **pull parser model:** you *pull* data from the document using the pull parser component

With pull parsing, you gather information from the file as you request it, which gives you the flexibility of conveniently changing the kinds of things you're looking for as you go. With the event-driven SAX model you configure your callbacks and then set the parser in motion.

"XML Reader" confusion

Well anyway, *I* was confused. The Microsoft .NET XmlReader is a pull parser, whereas the XML Standard API XMLReader is the interface of SAX2.

In both cases, the core feature of an "XML Reader" is that unlike DOM (the Document Object Model) an XML reader does not load the entire document into memory at once, but instead reads the file sequentially in a read-only forward-only manner allowing the client to gather the desired information.

Pull parser design beats SAX (soundly)

Just re-read the descriptions of the two models. The pull parser model reads like a straight-forward service component. You call its method to do something and it does it. The SAX model turns that logic on its head. With SAX, you have to set up methods so that it can call your methods when things happen. SAX was developed in the early stages of XML and despite its enormous drawbacks it has been institutionalized in the XML industry.

The pull parser model is more flexible and dramatically easier to work with. You don't need to do any setup and there is much less you need to know to use it. Check out CMarkup's C++ XML reader which uses the pull parser model. For example, you can grab a piece of information from a megabyte XML file in under a tenth of a second with negligible memory usage:

```
CMarkup xmlreader;
xmlreader.Open( "largeXMLfile.xml", MDF_READFILE );
str sVal = xmlreader.FindGetData( "//record[@id='7643']" );
xmlreader.Close();
```

Pulling a specific value is uncommon; more often you will be pulling numerous records for calculating statistics, importing into a database, or generating a report.

"XML Writer"

The corresponding component to generate a large XML file without first building the entire document in memory, is called an "XML Writer". The XML is pushed to file in a forward-only write-only manner. CMarkup can also be used as a C++ XML writer. With XML writer models there is no split between designs like there is with the XML reader push and pull models.

Posted 24 Mar 2009. Questions, comments? Contact info@firstobject.com.