Evaluating SAX

Author: Mihai Rotaru

Date: 1 Dec 2011

SAX, the Simple API for XML, is another take on parsing XML files. But, as opposed to the DOM approach, a SAX parser will not re-construct the entire document tree in memory, but instead operates in an event-driven manner. The SAX API allows clients to register callback functions for the items they are interested in (which can be text nodes, element nodes, processing instructions or comments). When the SAX parser will encounter these items, an event will be triggered and the callback function (if any are registered) will be called, allowing the client application to take action. Another event will be triggered when an end of any of those XML features is encountered, and eventual functions called.

Another difference between SAX and DOM is that there is no standard for SAX, and therefore the API is not guaranteed to be the same for all parsers. In practice, differences between various implementations are quite significant, requiring users to familiarize themselves with the particular flavor of SAX which the parser employs.

For example, a SAX parser is described by Microsoft's ISAXXMLReader (http://msdn.microsoft.com/en-us/library/aa924174.aspx); callback functions can be set using the ISAXContentHandler::startElement method. But fortunately other implementations outside COM ugliness are available for C++; one of the most popular choices is Xerces (http://xerces.apache.org/index.html), by Apache. It can be used like this:

```
void foobar_sax_handler::startElement(const XMLCh* const name,
AttributeList& attributes)
{
char* m_name = XMLString::transcode(name);
cout << "Element encountered: "<< m_name << endl;
XMLString::release(&m_name);
}
```

This assumes that we have defined a class named `foobar_sax_handler`, which inherits from `HandlerBase` and has a `startElement` method. We can set this class to handle XML events with the `setDocumentHandler` method of the `SaxParser` class. Our customized parser can then be invoked using the `parse( xml_file )` method of the SaxParser class, where xml_file represents the name of an XML file.

When the parse method is called, the `startElement` method will be called for each element, printing it's name. Of course, inside this method we can perform more specific actions, for example only printing or modifying certain elements.

SAX bindings are also available for other languages as well, for example the jssaxparser (http://code.google.com/p/jssaxparser/) is for JavaScript; a SAX parser is also available for Java (javax.xml.parsers.SAXParser). The API varies slightly between all these implementations, but the same principles apply; if one is familiar with using SAX with one programming language, switching to a different one will be relatively easy.