SAX

An API for reading and creating XML files. Examples for XML-Parsers with a SAX interface are Crimson and Xerces.

Get more information at http://www.saxproject.org.

"SAX? Events? I'm confused!"

I'll try to explain

Tree-view of a document

An XML document, as you might know, can be seen as a "tree" of "nodes". Take a look at a small example document:

```
<doc>
    <title>Hello</title>
    Humtidumpti dumpti dump...
    <special>some text</special>
</doc>
```

This document can be seen as a tree of nodes:

```
The root
  |
  +- element "doc"
       |
       +- element "title"
       |    |
       |    +- text "Hello"
       |
       +- text "Humptidumpti dumpti dump"
       |
       +- element "special"
            |
            +- text "some text"
```

...where "some text" belongs to the element "special" that belongs to the element "doc" etc.

Enter SAX: the event-view of a document

Now, for a computer to understand this tree view of the document, it has to basically parse the whole document.

Some people thought "That's inconvenient! Why can't we handle the document in a sequential manner instead?". So they invented (particulary a guy named David Meggison invented) SAX, the *S*imple (or "Streaming") *A*pi (for) *X*ML documents.

"An XML document", SAX says, "can be seen not only as a tree, but as a number of 'commands' or 'events'."

The example document above could be expressed as:

```
SAXEvent: "document begins"
SAXEvent: "element 'doc' begins"
SAXEvent: "element 'title' begins"
SAXEvent: "now comes the text 'Hello'"
SAXEvent: "element 'title' ends"
SAXEvent: "now comes the text 'Humtidumpti dumpti dump...'"
SAXEvent: "element 'special' begins"
SAXEvent: "now comes the text 'some text'"
SAXEvent: "element 'special' ends"
SAXEvent: "element 'doc' ends"
SAXEvent: "document ends"
```

How this is utilized in Cocoon pipelines

Nowadays there are XSLT engines that can do the trick of transforming XML documents in a streams based fashion. So when a Pipeline in Cocoon handles documents, it does so using a bunch of SAX-handling components.

The pipeline starts with a Generator that knows how to compute or read **something** (for example parsing an XML document) and from that generate a stream of SAX events ("document begins" etc). Then there might be a bunch of other components that modifies the stream of SAX events, for instance using XSLT or some other transforming algorithm.

In the end there is the Serializer that listens to the SAX events and translates them to a binary data stream. This transformation to binary data can be as simple as "translating" SAX events to text (the usual form of an XML document), or as complex as producing PDF from XSL:FO or a JPEG image from SVG.

Example pipeline

When the XMLGenerator looks at an XML file and sees "<doc>" it fires the SAX event "element 'doc' begins".

A Transformer might take it from here, perhaps instructed to transform every "<doc>" element into a "<document>" element. So when this Transformer recieves the "element 'doc' begins", it fires the SAX event "element 'document' begins".

In the other end of the pipeline, when the serializer recieves the SAX event "element 'document' begins", it converts it to the text "<document>".

Clear as crystal, don't you think?

Readers' comments

Yes, absolutely. Thank you so much. We (the users) really need articles like this - gabridome

bbbbbbbbut, this is fantastic.. the possibilities!! - gina