XML
DOM and SAX Parsers

By
Omar RABI

Introduction to parsers

- The word *parser* comes from compilers
- In a compiler, a parser is the module that reads and interprets the programming language.

Introduction to Parsers

- In XML, a parser is a software component that sits between the application and the XML files.

Figure 1: architecture of an xml program

Introduction to parsers

- It reads a text-formatted XML file or stream and converts it to a document to be manipulated by the application.

Well-formedness and validity

- Well-formed documents respect the syntactic rules.
- Valid documents not only respect the syntactic rules but also conform to a structure as described in a DTD.

Validating vs. Non-validating parsers

- Both parsers enforce syntactic rules
- only validating parsers know how to validate documents against their DTDs

Tree-based parsers

- These map an XML document into an internal tree structure, and then allow an application to navigate that tree.
- Ideal for browsers, editors, XSL processors.

Event-based

- An event-based API reports parsing events (such as the start and end of elements) directly to the application through callbacks.
- The application implements handlers to deal with the different events

Event-based vs. Tree-based parsers

- Tree-based parsers deal generally small documents.
- Event-based parsers deal generally used for large documents.

Event-based vs. Tree-based parsers

- Tree-based parsers are generally easier to implement.
- Event-based parsers are more complex and give hard time for the programmer

What is DOM?

- The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents.
- It defines the logical structure of documents and the way a document is accessed and manipulated

Properties of DOM

- Programmers can build documents, navigate their structure, and add, modify, or delete elements and content.
- Provides a standard programming interface that can be used in a wide variety of environments and applications.
- structural isomorphism.

DOM Identifies

- The interfaces and objects used to represent and manipulate a document.
- The semantics of these interfaces and objects - including both behavior and attributes.
- The relationships and collaborations among these interfaces and objects.

What DOM is not!!

- The Document Object Model is not a binary specification.
- The Document Object Model is not a way of persisting objects to XML or HTML.
- The Document Object Model does not define "the true inner semantics" of XML or HTML.

What DOM is not!!

- The Document Object Model is not a set of data structures, it is an object model that specifies interfaces.
- The Document Object Model is not a competitor to the Component Object Model (COM).

DOM into work

```
<?xml version="1.0"?>
<products>
    <product>
        <name>XML Editor</name>
        <price>499.00</price>
    </product>
    <product>
        <name>DTD Editor</name>
        <price>199.00</price>
    </product>
    <product>
        <name>XML Book</name>
        <price>19.99</price>
    </product>
    <product>
        <name>XML Training</name>
        <price>699.00</price>
    </product>
</products>
```

DOM into work

Figure 1: Building the tree of objects

DOM levels: level 0

- DOM Level 0 is a mix of Netscape Navigator 3.0 and MS Internet Explorer 3.0 document functionalities.

DOM levels: DOM 1

- It contains functionality for document navigation and manipulation.

i.e.: functions for creating, deleting and changing elements and their attributes.

DOM level 1 limitations

- A structure model for the internal subset and the external subset.
- Validation against a schema.
- Control for rendering documents via style sheets.
- Access control.
- Thread-safety.
- Events

DOM levels: DOM 2

- A style sheet object model and defines functionality for manipulating the style information attached to a document.
- Enables of the traversal on the document.
- Defines an event model.
- Provides support for XML namespaces

DOM levels: DOM 3

- Document loading and saving as well as content models (such as DTD's and schemas) with document validation support.
- Document views and formatting, key events and event groups

An Application of DOM

```
<HTML>
<HEAD>
<TITLE>Currency Conversion</TITLE>
<SCRIPT LANGUAGE="JavaScript" SRC="conversion.js"></SCRIPT>
</HEAD>
<BODY>
<CENTER>
<FORM ID="controls">
File: <INPUT TYPE="TEXT" NAME="fname" VALUE="prices.xml">
Rate: <INPUT TYPE="TEXT" NAME="rate" VALUE="0.95274" SIZE="4"><BR>
<INPUT TYPE="BUTTON" VALUE="Convert" ONCLICK="convert(controls,xml)">
<INPUT TYPE="BUTTON" VALUE="Clear" ONCLICK="output.value='"><BR>
<TEXTAREA NAME="output" ROWS="10" COLS="50" READONLY> </TEXTAREA>
</FORM>
<xml id="xml"></xml>
</CENTER>
</BODY>
</HTML>
```

An Application of DOM

- <xml id="xml"></xml>: defines an XML island.
- XML islands are mechanisms used to insert XML in HTML documents.
- In this case, XML islands are used to access Internet Explorer’s XML parser. The price list is loaded into the island.

An Application of DOM

- The “Convert” button in the HTML file calls the JavaScript function convert(), which is the conversion routine.
- convert() accepts two parameters, the form and the XML island.

An Application for DOM

```
<SCRIPT LANGUAGE="JavaScript" SRC="conversion.js"></SCRIPT>

function convert(form,xmldocument)
{var fname = form.fname.value,
output = form.output,
rate = form.rate.value;
output.value = "";
var document = parse(fname,xmldocument),
topLevel = document.documentElement;
searchPrice(topLevel,output,rate);}

function parse(uri,xmldocument)
{xmldocument.async = false;
xmldocument.load(uri);
if(xmldocument.parseError.errorCode != 0)
alert(xmldocument.parseError.reason);
return xmldocument;}

function searchPrice(node,output,rate)
{if(node.nodeType == 1)
{if(node.nodeName == "price")
output.value += (getText(node) * rate) + "\r";
var children,
i;
children = node.childNodes;
for(i = 0;i < children.length;i++)
searchPrice(children.item(i),output,rate);}}

function getText(node)
{return node.firstChild.data;}
```

An Application of DOM

- nodeType is a code representing the type of the object.
- parentNode is the parent (if any) of current Node object.
- childNode is the list of children for the current Node object.
- firstChild is the Node's first child.
- lastChild is the Node's last child.
- previousSibling is the Node immediately preceding the current one.
- nextSibling is the Node immediately following the current one.
- attributes is the list of attributes, if the current Node has any.

An Application of DOM

- The parse() function loads the price list in the XML island and returns its Document object.
- The function searchPrice() tests whether the current node is an element.

An Application of DOM

- The function searchPrice() visits each node by recursively calling itself for all children of the current node.

An Application for DOM

Figure 2: DOM Application

What is SAX?

- SAX (the Simple API for XML) is an event-based parser for xml documents.
- The parser tells the application what is in the document by notifying the application of a stream of parsing events.
- Application then processes those events to act on data.

SAX History

- SAX 1.0 was released on May 11, 1998.
- SAX is a common, event-based API for parsing XML documents, developed as a collaborative project of the members of the XML-DEV discussion under the leadership of David Megginson.

Why SAX?

- For applications that are not so XML-centric, an object-based interface is less appealing.
- Efficiency: lower level than object-based interfaces

Why SAX?

- Event-based interface consumes fewer resources than an object-based one
- With an event-based interface, the application can start processing the document as the parser is reading it

Limitations of SAX

- With SAX, it is not possible to navigate through the document as you can with a DOM.
- The application must explicitly buffer those events it is interested in.

SAX API

- Parser events are similar to user-interface events such as ONCLICK (in a browser) or AWT events (in Java).
- Events alert the application that something happened and the application might want to react.

SAX API

- Element opening tags
- Element closing tags
- Content of elements
- Entities
- Parsing errors

SAX API

Figure 3: Event-Based Parser

SAX Example

```
<?xml version="1.0"?>
<doc>
  <para>Hello, world!</para>
</doc>
```

SAX example

- start document
- start element: doc
- start element: para
- characters: Hello, world!
- end element: para
- end element: doc
- end document

Conclusion