XML: Tools and Extensions

Web Programming

Uta Priss
ZELL, Ostfalia University

2013

Outline

Tree-based parser

- Parses the whole XML document.
- Returns a data structure of 'nodes' representing elements, attributes, text content and other components.
- Has API of functions for searching and manipulating the tree.
- The Document Object Model (DOM) is a standard API.

Pros and cons of tree-based parsers

Pros:

- Easier to use.
- DOM is language neutral (Java, PHP, Perl, ...).
- Some DOM modules support XPath.

Cons:

- Portability issues (even DOM APIs differ).
- Memory requirements: 10-30 times of original document.

Stream-based parser

- Sends the data to the program in a stream of 'events'.
- The program needs some handler or 'callback' functions.
- SAX (the Simple API for XML) is a standard object-oriented API.

Pros and cons of stream-based parsers

Pros:

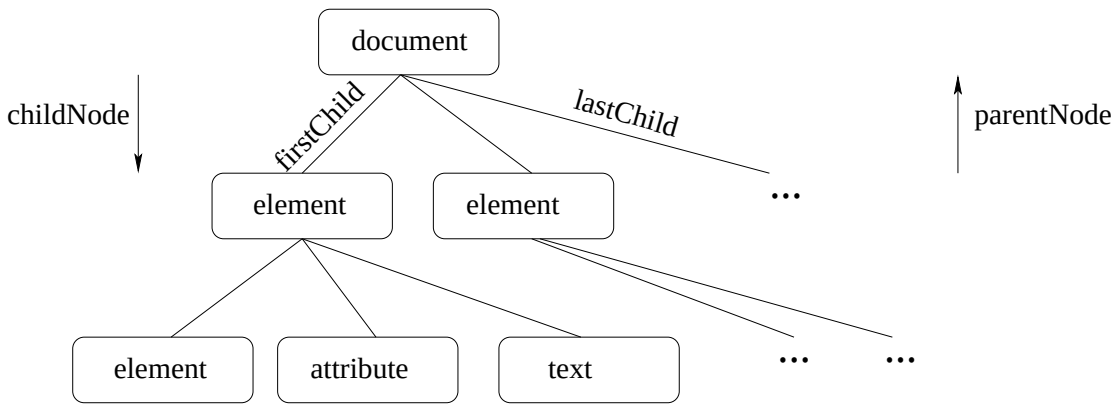
- Should be more efficient (but not all parsers are).
- SAX APIs are very portable.
- SAX encourages a very modular coding style.
- SAX also has non-XML applications.

Cons:

- Programmers that are new to XML, may find it more difficult to use.
- Whether the document is well-formed (and error free) is determined at the end of the parse.

DOM - Document Object Model

In DOM, the XML document is a tree of nodes:



Nodes

- node->nodeName, node->nodeValue
- node->nodeType (element, attribute, text, ...)
- node->childNodes is a list of other Nodes
- node->attributes is a list of keys and values
- node->parentNode, node->firstChild, node->lastChild
  node->previousSibling and node->lastSibling are other Nodes

Iterate over nodes

```
while (node) {
      do something with node
      node = node->nextSibling;
}
```

Or

```
arrayOfChildren = someElement->childNodes;
foreach node in arrayOfChildren {
      do something with node
}
```

Elements

- Elements are a subclass of Nodes.
- Can be retrieved via tagName or ID.
- element.tagName
- element.getAttribute('...')

DOM in PHP

```
$doc = new DOMDocument();
$doc->load("movies.xml");
echo $doc->validate();
echo $doc->doctype->name;
$topElem = $doc->documentElement;
foreach ($topElem->childNodes AS $item) {
...
```

SAX - Simple API for XML

Stream-based, event-based, serial parsing.

- Event: startElement
- Event: endElement
- Event: character data

At each event: callback, do something.
Use global variables to remember state.

In PHP: reading a file

Either line by line:

```
$fp = fopen($file, "r");
while ($data = fread($fp, 4096)) {
      xml_parse($xml_parser, $data, feof($fp));
}
```

Or, for short files:

```
$filecontent = implode('', file('movies.xml'));
xml_parse($xml_parser, $filecontent);
```

In PHP: setting up a parser

```
$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser,
      "startElement", "endElement");
xml_set_character_data_handler($xml_parser,"dataHandler");

xml_parse($xml_parser, $filecontent);
xml_parser_free($xml_parser);
```

Also available:

```
xml_parse_into_struct($xml_parser,
      $filecontent, $vals, $index);
```

In PHP: the event handlers

```
function startElement($parser, $name, $attrs) {
      global $depth;
      for ($i = 0; $i < $depth; $i++) {echo " "; }
      echo "$name: ";
      $depth++;
}
function endElement($parser, $name) {
      global $depth;
      $depth--;
      echo "<p>";
}
function dataHandler($parser,$data) {
      echo "$data";
}
```

SAX's internal data structure

```
[1] => Array (
       [tag] => MOVIE
       [type] => open
       [level] => 2
       [attributes] => Array ( [ID] => 5 )
       [value] =>
)
[2] => Array (
       [tag] => TITLE
       [type] => complete
       [level] => 3
       [value] => The Quest )
```

Data binding

Another form of XML processing is data binding.

- XML converted into programming language data structure.
- Data serialisation (similar to using non-XML serialisation: YAML and JSON).

Examples:
JAXB (Java Architecture for XML Binding); XMLBeans;
Hibernate;

Data binding in PHP

SimpleXML converts XML into an object:

```
<?php
if (file_exists('movies.xml')) {
      $xml = simplexml_load_file('movies.xml');
      print_r($xml);
}?>
```

And converts it back into XML:

```
print $xml->asXML();
```

But: SimpleXML cannot deal with namespaces.

When to use what

- If only partial information required from XML file
  $\Rightarrow$ use DOM or SAX
- If the file is large
  $\Rightarrow$ use the line by line processing option in SAX
- Simple data serialisation
  $\Rightarrow$ use data binding

Other XML processing

- libxml2: C library for XML parsing (used by PHP5)
- expat: stream-oriented XML parser (used by PHP4)
  one of the first open-source parsers; by James Clark
- SDO (Service Data Objects)
  single framework for heterogeneous data:
  relational databases, XML, Web Services, etc
- WDDX (Web Distributed Data eXchange)
  language-, platform- and transport-neutral
  XML data encoding, precursor of SOAP and Web Services