

SXML: S-expression eXtensible Markup Language

by Sterling Stuart Stein

Manipulating XML

- Many ways to manipulate XML
 - PHP
 - JavaScript
 - XSLT
- Many annoyances
 - Print statements (is it well-formed?)
 - Lack of safety (Stray ampersands?)
 - Verbose commands

PHP example

Echo:

```
<?php
  echo '<tag attr="val">';
  echo 'data';
  echo '</tag>';
?>
```

DOM:

```
<?php
  $doc = new DomDocument('1.0');
  $root = $doc->createElement('tag');
  $root->set_attribute("attr", "val");
  $doc->appendChild($root);
  $value = $doc->createTextNode('data');
  $root->appendChild($value);
  echo $doc->dump_mem();
?>
```

JavaScript is similar.

XSLT

Can have literals:

```
<xsl:template match="/">  
  <tag attr="val">  
    <xsl:text>data</xsl:text>  
  </tag>  
</xsl:template>
```

Hard to add attributes:

```
<tag>  
  <xsl:attribute name="attr">  
    <xsl:text>val</xsl:text>  
  </xsl:attribute>  
</tag>
```

XSLT 2

Verbose function call syntax: navitem("val1", "val2");

```
<xsl:call-template name="navitem">
  <xsl:with-param name="param1" select="'val1'" />
  <xsl:with-param name="param2" select="'val2'" />
</xsl:call-template>
```

Hard to extend: grouping

```
<xsl:key name="group-by-surname" match="contact" use="surname" />
<xsl:template match="records">
  <xsl:for-each select="contact[count(. | key('contacts-by-surname', surname)[1]) =
1]">
  <xsl:sort select="surname" />
  <xsl:value-of select="surname" />,<br />
  <xsl:for-each select="key('group-by-surname', surname)">
  <xsl:sort select="forename" />
  <xsl:value-of select="forename" /> (<xsl:value-of select="title" />)<br />
  </xsl:for-each>
</xsl:for-each>
</xsl:template>
```

S-expressions

- Stands for symbolic expression
- Fully parenthesized parse tree
- Usually uses prefix notation
 - $1+2*3 = (+ 1 (* 2 3))$
- Used to represent data and code in LISP

S-expressions and XML

- S-expressions and XML represent parse trees
- LISP can modify S-expressions
- Therefore, use LISP on XML = SXML
- Goal is to have uniform interfaces
- Mapping from XML to S-expressions:

```
<tag attr="val">          ("tag" (("attr"."val"))
  Data                    "Data"
  <inside />              ("inside" ())
</tag>                    )
```

Features

Quasiquotes make having executable snippets easy:

```
`("div" (("id"."nav"))
, (navitem "link" "dest")
, @morenodes
)
```

Similar to:
<?php ...code... ?>
but safer.

Attributes can be dealt with the same as everything else:

```
`("div"
  (("class" ., (if (here) "active" "inactive")))
)
```

Features 2

- Has transforms that are XSLT-like
- Has sorting by multiple criteria and grouping
- Has path matching
 - Currently far inferior to XSLT
 - Planning on path like: `table/tr = (* "table" "tr")`
- Will have SQL interface
 - Also specified as S-expressions through macros
 - Possibly with type safety to protect from SQL injection attacks
 - Results returned as SXML trees

Implementation

- Implemented as a library for Bigloo Scheme
- Allows for native compiling
 - Runs fast
- Safe for CGI
 - String lengths checked by language
 - SQL checked by macros
- Avoids destructive functions
 - Personal preference
 - Not suitable for huge documents

Thank you!

Any questions?
Any suggestions?