Seven sorts of sucking XML

as told by $madam$

Blumsoft Code Gardening, October 7, 2008

Outline

- ✗ memory consumption
- ✗ still needs manual processing
- ✗ . . . which can be awkward
- ✗ . . . and you risk dying of boreDOM
- ✓ XPath may help a lot
- ✗ . . . but that may be slow

- ✓ zero-overhead, real-time
- ✗ you are not in control
- ✗ you need to accumulate context yourself
- ✗ it is hard to catch invalid documents
- ✓ . . . but comes handy for ad-hoc filtering

```
public void processDocument(XmlPullParser xpp)
  throws XmlPullParserException, IOException
{
  int eventType = xpp.getEventType();
  do {
    if (eventType == xpp.START_DOCUMENT)
      System.out.println("Start␣document");
    else if (eventType == xpp.END_DOCUMENT)
      System.out.println("End␣document");
    else if (eventType == xpp.START_TAG)
      processStartElement(xpp);
    else if (eventType == xpp.END_TAG)
      processEndElement(xpp);
    else if (eventType == xpp.TEXT)
      processText(xpp);
    eventType = xpp.next();
  } while (eventType != xpp.END_DOCUMENT);
}
```

- ✓ produces ready-to-use data structures
- ✗ not for strongly typed languages
- ✗ may not be trivial to get it right

```
<config logdir="/var/log/foo/" debugfile="/tmp/foo.debug">
  <server name="sahara" osname="solaris" osversion="2.6">
    <address>10.0.0.101</address>
    <address>10.0.1.101</address>
  </server>
  <server name="gobi" osname="irix" osversion="6.5">
    <address>10.0.0.102</address>
  </server>
  <server name="kalahari" osname="linux"
    osversion="2.0.34">
    <address>10.0.0.103</address>
    <address>10.0.1.103</address>
  </server>
</config>
```

```
{
  'logdir'      => '/var/log/foo/',
  'debugfile'   => '/tmp/foo.debug',
  'server'      => {
    'sahara'       => {
      'osversion'   => '2.6',
      'osname'      => 'solaris',
      'address'     => [ '10.0.0.101', '10.0.1.101' ]
    }, 'gobi'      => {
      'osversion'   => '6.5',
      'osname'      => 'irix',
      'address'     => '10.0.0.102'
    }, 'kalahari' => {
      'osversion'   => '2.0.34',
      'osname'      => 'linux',
      'address'     => [ '10.0.0.103', '10.0.1.103' ]
    }
  }
}
```

```
my $t = XML::Twig->new( # The twig will include
  twig_roots => {       # just the root and the
    'section/title' => \&print_n_purge, # selected
    'annex/title'   => \&print_n_purge  # titles.
  });
$t->parsefile('doc.xml');

sub print_n_purge
{
  my ($t, $elt) = @_;

  print $elt->text(); # Print the text
                      # (including sub-elements).
  $t->purge();        # Frees the memory.
}
```

- ✓ less memory-hungry than DOM
- ✓ has more context than SAX
- ✗ twig processing is like DOM
- ✗ convenience comes at a price

- `myxml_begin()`
- `myxml_process()`
- `myxml_finish()`

- `myxml_get_attr()`
- `myxml_node_is()`
- `myxml_get_text()`

- `myxml_node_started()`
- `myxml_node_closed()`
- `myxml_ignored_node_started()`
- `myxml_ignored_text_node_started()`

```
void parse_event_start(struct myxml_st *myxml,
  XML_Char const *node, const XML_Char *const *atts)
{
  XML_Char const *state = myxml_get_state(myxml);
  struct events_parsing_st *self = myxml->udata;

  if (state == NULL || state == NODE_PEOPLE) {
    if (myxml_node_starts_if(myxml, node, NULL,
                             NODE_GUY, NULL)) {
      /* ... */
      return;
    }
  } else if (state == NODE_GUY) {
    if (myxml_text_node_starts_if(myxml, node, NULL,
                                  NODE_NAME, NULL))
      return;
  }
  myxml_ignored_node_started(myxml);
}
```

```
class Reader {
public: /* Constructors */
  Reader(CSink *sink);

public: /* Public methods */
  CObject *parse(CObject *cargo = NULL);

  static bool is_start_tag(CString const *tag);
  static bool is_end_tag(CString const *tag);
  static bool is_text(CString const *tag);
  static void verify_parent(CString const *parent, ...);

protected: /* Unimplemented interfaces */
  virtual CObject *event(
    CString const *tag, CObject *cargo,
    CString const *key, CObject *value) = 0;
  /* ... */
};
```

```
class CTreeBuilder:
  public gslib::gsxml::Reader
{
  /* ... */
protected: /* Implemented interfaces */
  virtual gslib::CObject *event(
    gslib::CString const *tag, gslib::CObject *cargo,
    gslib::CString const *key, gslib::CObject *value);
  /* ... */
};
```

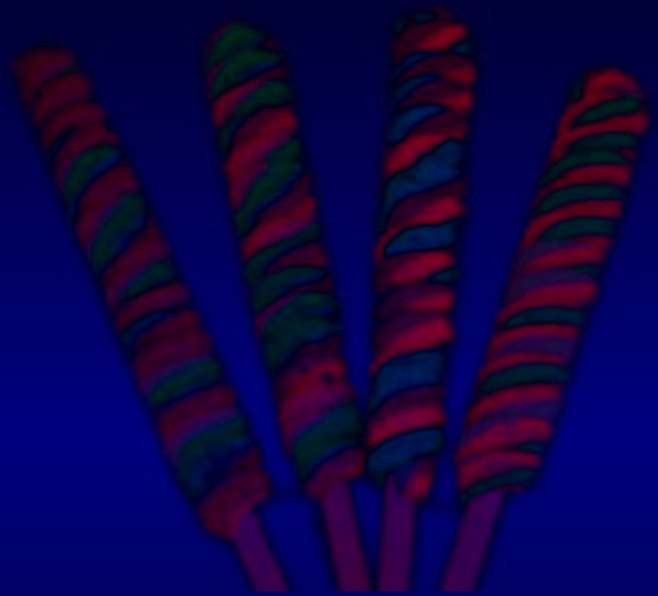
```
/*
 * FYI:
 * patag := parent's tag
 * pargo := parent's cargo
 */
cargo = event(tag, pargo, Start,  patag);

cargo = event(tag, cargo, attr1,  val1);
cargo = event(tag, cargo, attr2,  val2);
...
cargo = event(tag, cargo, Text,   text1);
cargo = event(tag, cargo, child,  value);

cargo = event(tag, cargo, Text,   text2);
...
cargo = event(tag, cargo, Finish, pargo);
```

Conclusion

KTHXBYE



- There's More Than One Way To Do It.
- . . . which is limited by our language of choice
- . . . but why not make the choice the other way around?