



Orchard: A New API for XML

Matt Sergeant and Ken MacLeod
The Perl XML Community



Background

What is

- DOM is a way to model trees
- SAX is a way to model streams

What is wrong

- "Language independent APIs" aren't natural
 - ◆ Leads to development of "natural" APIs: JDOM, XML::Simple
- Fixed intersection with XML Information Set
 - ◆ Editors need more, common XML needs far less
- PEEK and POKE

What is different

- Data models more language independent
 - ◆ DOM and SAX say, "come to me"
 - ◆ Orchard says, "I'll come to you"
- Progressive disclosure, scalable intersection
 - ◆ Documents, Elements, Attributes, and Text
- Orchard merges features of SAX and DOM
- Emphasis on application-specific data models



Common XML in Orchard

XML nodes

- Document
 - ◆ root, contents
- Element
 - ◆ name, namespace_uri, prefix, local_name, attributes, contents
- Attribute
 - ◆ name, namespace_uri, prefix, local_name, value
- Characters
 - ◆ data

- Properties are case and underscore mapped
 - ◆ NamespaceURI <-> namespace-uri <-> namespace_uri

XML nodes example -- Source XML

```
<book xmlns:dc="http://purl.org/dc/elements/1.1/"  
      xmlns:my="http://bookseller.biz/">  
  <dc:title>Catching Roadrunners</dc:title>  
  <dc:author>Wile E. Coyote</dc:/author>  
  <my:info my:price="$34.85" my:in_stock="true"/>  
</book>
```

XML nodes reading example -- Perl

```
use Orchard qw{namespace};
use Orchard::XML;

my $DC = namespace("http://purl.org/dc/elements/1.1/");
my $MY = namespace("http://bookseller.biz/");

my $book = Orchard::XML->load('book.xml');

my $title = $book->get_elements_by_tag_name($DC->title);
my $author = $book->get_elements_by_tag_name($DC->author);
my $info = $book->get_elements_by_tag_name($MY->info);

print "Title:  $title\n";
print "Author: $author\n";
print "Price:  " . $info->{Attributes}{$MY->price} . "\n";
```

XML nodes writing example -- Python

```
from Orchard import namespace
from Orchard.XML import *

DC = namespace("http://purl.org/dc/elements/1.1/")
MY = namespace("http://bookseller.biz/")

doc = Orchard.XML.Document()
root = doc.create_element('book')

title = doc.create_element(DC.title)
title.contents = "Catching Roadrunners"

author = doc.create_element(DC.author)
author.contents = "Wile E. Coyote"

info = doc.create_element(MY.info)
info.attributes[MY.price] = "$34.95"
info.attributes[MY.in_stock] = "true"

root.contents.append(title, author, info)

print root.to_string()
```

SAX

- Passes nodes as parameter
- Features govern node properties
- Reduced need for callbacks and interfaces
- Scalable, dynamic interface
- Pull
 - ◆ `next_event()`, `peek_event()`, `skip_events()`

Beyond XML

Types of data modeling

- lists -- scheme, lisp
- rows and columns -- RDBs, spreadsheets
- tuples, arcs -- RDF/Topic Maps, logic
- nodes, objects -- common languages, Groves, and Orchard
- XML is just one set of nodes in Orchard
 - ◆ RSS, SVG, SlideShow, CDDb, MPEG, RDF, ...
- RDF and Groves (Orchard) are complementary
 - ◆ RDF can be very loose and much more flexible
 - ◆ Groves are simpler in that respect
 - ◆ Groves and RDF are interoperable

Common node behaviors

- Introspection
- Iteration, navigation
- Namespaced property names
- Underlying storage, load/save
- XPath, XSLT

RSS example -- Python

```
from Orchard import namespace
import Orchard.RSS

channel = \
    Orchard.RSS.load('http://MonkeyFist.com/rss1.php3')

DC = namespace("http://purl.org/dc/elements/1.1/")

print "Site:           " + channel.title
print "URL:            " + channel.link
print "Description:    " + channel.description
print "Copyright:      " + channel[DC.rights]
print "Language:       " + channel[DC.language]
print "Publisher:      " + channel[DC.publisher]
print
print "Items:"

for item in channel.items:
    print "  Title:           " + item.title
    print "  Link:              " + item.link
    print "  Description:      " + item.description
    print "  Link creator:    " + item[DC.creator]
    print "  Link date:       " + item[DC.date]
    print
```

RSS example -- Perl

```
use Orchard qw{namespace};
use Orchard::RSS;

my $channel = \
    Orchard::RSS->load('http://MonkeyFist.com/rss1.php3');

my $DC = namespace('http://purl.org/dc/elements/1.1/');

print "Site:           " + $channel->{Title}           . "\n";
print "URL:           " + $channel->{Link}             . "\n";
print "Description:   " + $channel->{Description}      . "\n";
print "Copyright:    " + $channel->{$DC->rights}       . "\n";
print "Language:     " + $channel->{$DC->language}     . "\n";
print "Publisher:    " + $channel->{$DC->publisher}    . "\n";
print "\n";
print "Items:\n";

foreach my $item ($channel->{Items}) {
    print "  Title:           " + $item->{Title}           . "\n";
    print "  Link:           " + $item->{Link}             . "\n";
    print "  Description:    " + $item->{Description}      . "\n";
    print "  Link creator:   " + $item->{$DC->creator}     . "\n";
    print "  Link date:     " + $item->{$DC->date}         . "\n";
    print "\n";
}
```

What's next

- XPath and XSLT on any node tree
- DOM level 2 and 3 on any node tree
- Vertical development
 - ◆ More node sets, transforms
- Horizontal development
 - ◆ language bindings, storage drivers, common behaviors
- Cross-compatibility with RDF
- Node set schema languages
- DOM and SAX interoperability, conformance testing

Resources and contact

- <http://Orchard.SourceForge.net/>
- <http://www.w3.org/DOM/>
- <http://www.Megginson.com/SAX/>

- Ken MacLeod <ken@bitsko.slc.ut.us>
- Matt Sergeant <matt@axkit.com>