

Drupal and SimpleXML

XML parsing

- Very well defined, well known standards
- Excellent support ...
- ... in most other languages
- Long-standing PHP weakness
(up to version 4)

two main models

SAX

Simple API for XML

- Stream - based xml parsing
- Event - driven API
- Uni-directional

DOM

Document Object Model

- Full object representation of an XML document
- Supports writing documents
- Same system used by browsers, javascript, etc (i.e. “known quantity”)

Other key concepts

- Namespaces: Allow extending existing XML documents
- XPath: selector syntax for addressing parts of a document
- XSLT: “template” language for transforming XML documents into other formats

DOM in PHP

- <http://php.net/domxml>
- Traditionally required external libraries and additional PHP compile flags
- Has been removed from PHP core in PHP5 (available as a PECL module)
- not really an option for Drupal core

SAX in PHP

- Available in PHP 4 & PHP 5 (`--with-xml`)
- Closest “built-in” support
 - (hence what we use in Drupal)
- <http://php.net/xml>

SAX example

```
function aggregator_parse_feed(&$data, $feed) {  
  global $items, $image, $channel;  
  
  // Unset the global variables before we use them:  
  unset($GLOBALS['element'], $GLOBALS['item'], $GLOBALS['tag']);  
  $items = array();  
  $image = array();  
  $channel = array();  
  
  // parse the data:  
  $xml_parser = drupal_xml_parser_create($data);  
  xml_set_element_handler($xml_parser, 'aggregator_element_start', 'aggregator_element_end');  
  xml_set_character_data_handler($xml_parser, 'aggregator_element_data');  
  
  if (!xml_parse($xml_parser, $data, 1)) {  
    watchdog('aggregator', 'The feed from %site seems to be broken, due to an error "%error" on line %line.',  
      array('%site' => $feed['title'], '%error' => xml_error_string(xml_get_error_code($xml_parser)), '%line'  
=> xml_get_current_line_number($xml_parser)), WATCHDOG_WARNING);  
    drupal_set_message(t('The feed from %site seems to be broken, because of error "%error" on line %line.',  
      array('%site' => $feed['title'], '%error' => xml_error_string(xml_get_error_code($xml_parser)), '%line'  
=> xml_get_current_line_number($xml_parser))), 'error');  
    return 0;  
  }  
  xml_parser_free($xml_parser);  
}
```

SAX example (cont.)

```
function aggregator_element_start($parser, $name, $attributes) {
    global $item, $element, $tag, $items, $channel;

    switch ($name) {
        case 'IMAGE':
        case 'TEXTINPUT':
        case 'CONTENT':
        case 'SUMMARY':
        case 'TAGLINE':
        case 'SUBTITLE':
        case 'LOGO':
        case 'INFO':
            $element = $name;
            break;
        case 'ID':
            if ($element != 'ITEM') {
                $element = $name;
            }
        case 'LINK':
            if (!empty($attributes['REL']) && $attributes['REL'] == 'alternate') {
                if ($element == 'ITEM') {
                    $items[$item]['LINK'] = $attributes['HREF'];
                }
                else {
                    $channel['LINK'] = $attributes['HREF'];
                }
            }
            break;
        case 'ITEM':
```

SAX example (cont)

```
function aggregator_element_end($parser, $name) {
    global $element;

    switch ($name) {
        case 'IMAGE':
        case 'TEXTINPUT':
        case 'ITEM':
        case 'ENTRY':
        case 'CONTENT':
        case 'INFO':
            $element = '';
            break;
        case 'ID':
            if ($element == 'ID') {
                $element = '';
            }
        }
    }
}
```

SAX example (cont)

```
function aggregator_element_data($parser, $data) {
    global $channel, $element, $items, $item, $image, $tag;
    $items += array($item => array());
    switch ($element) {
        case 'ITEM':
            $items[$item] += array($tag => '');
            $items[$item][$tag] .= $data;
            break;
        case 'IMAGE':
        case 'LOGO':
            $image += array($tag => '');
            $image[$tag] .= $data;
            break;
        case 'LINK':
            if ($data) {
                $items[$item] += array($tag => '');
                $items[$item][$tag] .= $data;
            }
            break;
        case 'CONTENT':
            $items[$item] += array('CONTENT' => '');
            $items[$item]['CONTENT'] .= $data;
            break;
        case 'SUMMARY':
            $items[$item] += array('SUMMARY' => '');
            $items[$item]['SUMMARY'] .= $data;
            break;
        case 'TAGLINE':
        case 'SUBTITLE':
            $channel += array('DESCRIPTION' => '');
            $channel['DESCRIPTION'] .= $data;
            break;
    }
}
```

SAX drawbacks

- Awkward maintain state across events
- Requires prior knowledge of the document
- Maybe not so simple (lots to code)

Enter SimpleXML

SimpleXML

- DOM-based processing - including xpath
- **Included** in PHP5
- And, um, it's simple

SimpleXML example

```
$result = simplexml_load_string($data);
```

Example file

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0"
  xmlns:media="http://search.yahoo.com/mrss/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  >
<channel>
  <title>Photos from walkah</title>
  <link>http://www.flickr.com/photos/walkah/</link>
  <description></description>
  <pubDate>Thu, 23 Aug 2007 18:45:53 -0800</pubDate>
  <lastBuildDate>Thu, 23 Aug 2007 18:45:53 -0800</lastBuildDate>
  <item>
    <title>rights of passage...</title>
    <link>http://www.flickr.com/photos/walkah/1217848261/</link>
    <description>&lt;p&gt;&lt;a href="http://www.flickr.com/people/walkah/"&gt;walkah&lt;/a&gt; posted a photo</p></description>
    <pubDate>Thu, 23 Aug 2007 18:45:53 -0800</pubDate>
    <dc:date.Taken>2007-08-23T16:26:04-08:00</dc:date.Taken>
    <author>nobody@flickr.com (walkah)</author>
    <guid isPermaLink="false">tag:flickr.com,2004:/photo/1217848261</guid>
    <media:content url="http://farm2.static.flickr.com/1150/1217848261_61198a9c8a_m.jpg"
      type="image/jpeg"
      height="192"
      width="240"/>
    <media:title>rights of passage...</media:title>
    <media:text type="html">&lt;p&gt;&lt;a href="http://www.flickr.com/people/walkah/"&gt;walkah&lt;/a&gt; po</p></media:text>
    <media:thumbnail url="http://farm2.static.flickr.com/1150/1217848261_61198a9c8a_s.jpg" height="75" width="75" />
    <media:credit role="photographer">walkah</media:credit>
    <media:category scheme="urn:flickr:tags">ears camryn piercedears</media:category>
  </item>
```

SimpleXML example

```
$xml = simplexml_load_file('flickr.xml');  
foreach ($xml->channel->item as $item) {  
    print $item->title . "<br />";  
}
```

XPath example

```
$xml = simplexml_load_file('flickr.xml');  
foreach ($xml->xpath('/rss/channel/item/title') as $title) {  
    print $title . "<br />";  
}
```

SimpleXML

+

drupal_http_request() ftw

simple webservices

```
$result = drupal_http_request('http://api.flickr.com/services/feeds/photos_public.gne?id=44124266211@N01');
if ($result->code == 200) {
  $xml = simplexml_load_file($result->data);
  foreach ($xml->channel->item as $pic) {
    print $pic->title . '<br />';
  }
}
```

When SimpleXML isn't
simple

namespaces

Example file

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0"
  xmlns:media="http://search.yahoo.com/mrss/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  >
<channel>
  <title>Photos from walkah</title>
  <link>http://www.flickr.com/photos/walkah/</link>
  <description></description>
  <pubDate>Thu, 23 Aug 2007 18:45:53 -0800</pubDate>
  <lastBuildDate>Thu, 23 Aug 2007 18:45:53 -0800</lastBuildDate>
  <item>
    <title>rights of passage...</title>
    <link>http://www.flickr.com/photos/walkah/1217848261/</link>
    <description>&lt;p&gt;&lt;a href=&quot;http://www.flickr.com/people/walkah/&quot;&gt;walkah&lt;/a&gt; posted a photo</p></description>
    <pubDate>Thu, 23 Aug 2007 18:45:53 -0800</pubDate>
    <dc:date.Taken>2007-08-23T16:26:04-08:00</dc:date.Taken>
    <author>nobody@flickr.com (walkah)</author>
    <guid isPermaLink="false">tag:flickr.com,2004:/photo/1217848261</guid>
    <media:content url="http://farm2.static.flickr.com/1150/1217848261_61198a9c8a_m.jpg"
      type="image/jpeg"
      height="192"
      width="240"/>
    <media:title>rights of passage...</media:title>
    <media:text type="html">&lt;p&at:&lt;a href=&quot;http://www.flickr.com/people/walkah/&quot;&at:walkah&lt;/a&at: no</p></media:text>
    <media:thumbnail url="http://farm2.static.flickr.com/1150/1217848261_61198a9c8a_s.jpg" height="75" width="75" />
    <media:credit role="photographer">walkah</media:credit>
    <media:category scheme="urn:flickr:tags">ears camryn piercedears</media:category>
  </item>
```

Namespaces example

```
$result = drupal_http_request('http://api.flickr.com/services/feeds/photos_public.gne?id=44124266211@N01');
if ($result->code == 200) {
  $xml = simplexml_load_data($result->data);
  foreach ($xml->channel->item as $pic) {
    $media = $pic->children('http://search.yahoo.com/mrss/');
    $thumb = $media->thumbnail->attributes();
    print '<a href="' . check_url($pic->link) . '" title="' . check_plain($pic->title) . '">';
    print '';
    print '</a>';
  }
}
```

maybe it's not so bad

XML Writing

XML Writing example

```
$xml = simplexml_load_file('flickr.xml');  
$xml->channel->addAttribute('pants', 'off');  
print $xml->asXML();
```

```
<?xml version="1.0" encoding="utf-8"?>  
<rss xmlns:media="http://search.yahoo.com/mrss/" xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0">  
  <channel pants="off">  
    <title>Photos from walkah</title>  
    <link>http://www.flickr.com/photos/walkah/</link>  
    <description/>  
    <pubDate>Thu, 23 Aug 2007 18:45:53 -0800</pubDate>  
    <lastBuildDate>Thu, 23 Aug 2007 18:45:53 -0800</lastBuildDate>  
  </channel>  
</rss>
```

Advantages for Drupal

- Shorter, cleaner code (think cleaner aggregator.module, etc)
- Manipulatable DOM structure (think hook_xml_alter)
- XML Writing (think node_feed, etc)

Thanks, PHP5