Advanced XML with PHP 5

March 14, 2007

Robert Richards

rrichards@ctindustries.net

http://www.cdatazone.org/talks/quebec_2007/workshop.zip

Agenda

- Introduction to Terms and Concepts
- Libxml
- SimpleXML
- SDO
- DOM
- XMLReader
- XMLWriter
- XSL

XML Namespaces

- An XML Namespace is a collection of names identified by a URI.
- They are applicable to elements and attributes.
- Namespaces may or may not be associated with a prefix.
  - xmlns:rob="urn:rob"
  - xmlns=http://www.example.com/rob
- Attributes never reside within a default namespace.
- It is illegal to have two attributes with the same localname and same namespace on the same element.

XML Namespace Example

```
<order num="1001">
  <shipping>
    <name type="care_of">John Smith</name>
    <address>123 Here</address>
  </shipping>
  <billing>
    <name type="legal">Jane Doe</name>
    <address>456 Somewhere else</address>
  </billing>
</order>
```

XML Namespace Example

```
<order num="1001" xmlns="urn:order"
                  xmlns:ship="urn:shipping"
                  xmlns:bill="urn:billing">
  <ship:shipping>
    <ship:name type="care_of">John Smith</ship:name>
    <ship:address>123 Here</ship:address>
  </ship:shipping>
  <bill:billing>
    <bill:name type="legal">Jane Doe</bill:name>
    <bill:address>456 Somewhere else</bill:address>
  </bill:billing>
</order>
```

Illegal Namespace Usage

```
<order num="1001" xmlns="urn:order"
                  xmlns:order="urn:order"
                  xmlns:ship="urn:order">
  <shipping ship:type="fed_ex" type="fed_ex">
    <name ship:type="care_of"
          order:type="legal">John Smith</ship:name>
  </ship:shipping>
</order>
```

Both "type" attributes are bound to same namespace which is not valid!

Illegal Namespace Usage

```
<order num="1001" xmlns="urn:order"
                  xmlns:order="urn:order"
                  xmlns:ship="urn:order">
  <shipping ship:type="fed_ex" type="fed_ex">
    <name ship:type="care_of"
          order:type="legal">John Smith</ship:name>
  </ship:shipping>
</order>
<!-- attributes on shipping element are valid ! -->
```

Reserved Namespaces and Prefixes

- The prefix xml is bound to http://www.w3.org/XML/1998/namespace.
- The prefix xmlns is bound to http://www.w3.org/2000/xmlns/.
- Prefixes should also not begin with the characters xml.

Schemas and Validation

- Validation insures an XML document conforms to a set of defined rules.
- Multiple mechanisms exist to write document rule sets:
  - Document Type Definition (DTD)
  - XML Schema
  - RelaxNG

Document Type Definition (DTD)

validation/courses-dtd.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE courses [
    <!ELEMENT courses (course+)>
    <!ELEMENT course (title, description, credits, lastmodified)>
    <!ATTLIST course cid ID #REQUIRED>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT description (#PCDATA)>
    <!ELEMENT credits (#PCDATA)>
    <!ELEMENT lastmodified (#PCDATA)>
]>
<courses>
    <course cid="c1">
        <title>Basic Languages</title>
        <description>Introduction to Languages</description>
        <credits>1.5</credits>
        <lastmodified>2004-09-01T11:13:01</lastmodified>
    </course>
</courses>
```

DTD and IDs

validation/course-id.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE courses [
    <!ATTLIST course cid ID #REQUIRED>
]>
<courses>
    <course cid="c1">
        <title xml:id="t1">Basic Languages</title>
        <description>Introduction to Languages</description>
    </course>
    <course cid="c2">
        <title xml:id="t3">French I</title>
        <description>Introduction to French</description>
    </course>
</courses>
```

XML Schema

validation/course.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="courses">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="course" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="title" type="xsd:string"/>
              <xsd:element name="description" type="xsd:string"/>
              <xsd:element name="credits" type="xsd:decimal"/>
              <xsd:element name="lastmodified" type="xsd:dateTime"/>
            </xsd:sequence>
            <xsd:attribute name="cid" type="xsd:ID"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

RelaxNG

validation/course.rng

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
        datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <start>
    <element name="courses">
      <oneOrMore>
        <element name="course">
          <attribute name="cid"><data type="ID"/></attribute>
          <element name="title"><text/></element>
          <element name="description"><text/></element>
          <element name="credits"><data type="decimal"/></element>
          <element name="lastmodified"><data type="dateTime"/></element>
        </element>
      </oneOrMore>
    </element>
  </start>

</grammar>
```

XPath

- Language to locate and retrieve information from an XML document
- A foundation for XSLT
- An XML document is a tree containing nodes
- The XML document is the root node
- Locations are addressable similar to the syntax for a filesystem

XPath Reference Document

xpath/courses.xml

```
<courses xmlns:t="http://www.example.com/title">

    <course xml:id="c2">
        <t:title>French I</t:title>
        <description>Introduction to French</description>
    </course>

    <course xml:id="c3">
        <t:title>French II</t:title>
        <description>Intermediate French</description>
        <pre-requisite cref="c2" />
        <defns xmlns="urn::default">content</defns>
    </course>

</courses>
```

XPath Location Example

xpath/location.php

Expression:

```
/courses/course/description
/courses/*/description
//description
//description[ancestor::course]
```

Resulting Nodset:

```
<description>Introduction to French</description>
<description>Intermediate French</description>
```

XPath Function Example

xpath/function.php

string(/courses/course/pre-requisite[@cref="c2"]/..)

French II
Intermediate French

content

Reference:

```
<courses xmlns:t="http://www.example.com/title">
    <course xml:id="c3">
        <!-- additional elements containing the textual content ommitted for brevity -->
        <pre-requisite cref="c2" />
    </course>
</courses>
```

XPath and Namespaces

xpath/namespaces.php

//title

Empty NodeSet

//t:title

<t:title>French I</t:title>
<t:title>French II</t:title>

//defns

Empty NodeSet

//*[local-name()="defns"]

<defns xmlns="urn:default">content</defns>

Reference:

```
<courses xmlns:t="http://www.example.com/title">
    <course xml:id="c2">
        <t:title>French I</t:title>
    </course>

    <course xml:id="c3">
        <t:title>French II</t:title>
        <defns xmlns="urn::default">content</defns>
    </course>
</courses>
```

PHP and XML

- PHP 5 introduced numerous interfaces for working with XML
- The libxml2 library (http://www.xmlsoft.org/) was chosen to provide XML support
- The sister library libxslt provides XSLT support
- I/O is handled via PHP streams
- ext/domxml was removed in favor of ext/dom

XML Entensions for PHP 5

- ext/libxml
- ext/xml (SAX push parser)
- ext/dom
- ext/simplexml
- ext/xmlreader (pull parser)
- pecl/sdo (SCA_SDO)
- ext/xmlwriter
- ext/xsl
- ext/wddx
- ext/soap

Libxml

- Contains common functionality shared across extensions.
- Defines constants to modify parse time behavior.
- Provides access to streams context.
- Allows modification of error handling behavior for XML based extensions.

Libxml: Parser Options

LIBXML_NOENT	Substitute entities with replacement content
LIBXML_DTDLOAD	Load subsets but do not perform validation
LIBXML_DTDATTR	Create defaulted attributes defined in DTD
LIBXML_DTDVALID	Loads subsets and perform validation
LIBXML_NOERROR	Suppress parsing errors from libxml2
LIBXML_NOWARNING	Suppress parser warnings from libxml2
LIBXML_NOBLANKS	Remove insignificant whitespace on parsing
LIBXML_XINCLUDE	Perform XIncludes during parsing
LIBXML_NOCDATA	Merge CDATA nodes in Text nodes
LIBXML_NONET	Disable network access when loading

Libxml: Error Handling

bool libxml_use_internal_errors ([bool use_errors])

void libxml_clear_errors (void)

LibXMLError libxml_get_last_error (void)

array libxml_get_errors (void)

Libxml: LibXMLError

Class: LibXMLError

Properties (Read-Only):

- (int) level
- (int) code
- (int) column
- (string) message
- (string) file
- (int) line

LibXMLError::code Values:

- LIBXML_ERR_NONE
- LIBXML_ERR_WARNING
- LIBXML_ERR_ERROR
- LIBXML_ERR_FATAL

LibXMLError Example

libxml/error.php

```
<?php
/* Regular Error Handling */
$dom = new DOMDocument();
$dom->loadXML('<root>');

/* New Error Handling */
libxml_use_internal_errors(TRUE);

if (! $dom->loadXML('root')) {
   $arrError = libxml_get_errors();
   foreach ($arrError AS $xmlError) {
        var_dump($xmlError);
   }
} else {
   print "Document Loaded";
}
?>
```

LibXMLError Result

PHP Warning: DOMDocument::loadXML(): Premature end of data in tag root line 1 in Entity, line: 1 in /home/rrichards/workshop/libxml/error.php on line 4

Warning: DOMDocument::loadXML(): Premature end of data in tag root line 1 in Entity, line: 1 in /home/rrichards/workshop/libxml/error.php on line 4

New Error Handling:

```
object(LibXMLError)#2 (6) {
  ["level"]=>  int(3)
  ["code"]=>  int(4)
  ["column"]=>  int(1)
  ["message"]=>  string(34) "Start tag expected, '<' not found"
  ["file"]=>  string(0) ""
  ["line"]=>  int(1)
}
```

Libxml: Stream Context

```
$opts = array(
    'http' => array(
        'user_agent' => 'PHP libxml2 agent',
        'proxy' => 'tcp://localhost:8082',
        'request_fulluri' => TRUE
    )
);

$context = stream_context_create($opts);
libxml_set_streams_context($context);

$doc = DOMDocument::load('http://www.example.org/file.xml');
```

Preparing for Unicode in PHP 6

- XML extensions are unicode ready for PHP 6
  - Both the UTF-8 binary strings (existing behavior) **AND** unicode strings.
  - With Unicode enabled, unicode strings are returned instead of the current UTF-8 binary strings.
- Binary strings are required for Loading and Saving **String** based XML documents
  - Insures original encoding is not lost
  - Does not affect usage with unicode disabled

Forward Compatibility

```
$xml = (binary)"<root>My Document</root>";
$sxe = simplexml_load_string($xml);

$xml = b'<root>MyDocument</root>';
$dom = new DOMDocument();
$dom->loadXML($xml);

$sxe = simplexml_load_string($dom->saveXML());
```

SimpleXML

- Provides simple access to XML documents
- Operates only on elements and attributes
- Contains XPath support
- Allows for modifications to the XML
- Zero copy interoperability with DOM
- Examples for PHP 5.1.6+
- New in PHP 5.1.3
  - Elements and attributes can be added using addChild() and addAttribute() methods.
  - Node names can be retrieved by calling getName().

SimpleXML: Consuming Yahoo WebSearch

simplexml/yahoo_websearch_results.xml

```
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    totalResultsReturned="3" firstResultPosition="1">

  <Result>
    <Title>Zend Technologies - PHP 5 . . .</Title>
    <Summary>titletext = $dom-&amp;gt;createText. . .</Summary>
    <Url>http://www.zend.com/php5/articles/php5-x. . .</Url>
    <ClickUrl>http://uk.wrs.yahoo.com/_ylt=...</ClickUrl>
    <DisplayUrl>www.zend.com/php5/articles/php. . .</DisplayUrl>
    <ModificationDate>1171872000</ModificationDate>
    <MimeType>text/html</MimeType>
    <!-- Abbreviated Content -->
  </Result>

  <!-- Abbreviated Response -->
</ResultSet>
```

SimpleXML: Consuming Yahoo WebSearch

simplexml/yahoo_search.php

```
/* URL to Web Search service */
$url = 'http://api.search.yahoo.com/WebSearchService/V1/webSearch';
/* The query is separate here as the terms must be encoded. */
$url .= '?query='.rawurlencode('php5 xml');
/* Complete the URL adding App ID, limit to 3 results and only English results */
$url .= "&appid=zzz&results=3&language=en";

$sxe = simplexml_load_file($url);

/* Check for number of results returned */
if ((int)$sxe['totalResultsReturned'] > 0) {

   /* Loop through results and print title, url and modification date */
   foreach ($sxe->Result AS $result) {
      print 'Title: '.$result->Title."\n";
      print 'Url: '.$result->Url."\n";
      print 'Mod Date: '.date ('M d Y', (int)$result->ModificationDate)."\n\n";
   }
}
```

SimpleXML: Consuming Yahoo WebSearch

RESULTS

Title: Zend Technologies - PHP 5 In Depth - XML in PHP 5 - What's New?
Url: http://www.zend.com/php5/articles/php5-xmlphp.php
Mod Date: Feb 19 2007

Title: Zend Developer Zone | XML in PHP 5 - What's New?
Url: http://devzone.zend.com/node/view/id/1713
Mod Date: Feb 26 2007

Title: FreshPorts -- textproc/php5-xml
Url: http://www.freshports.org/textproc/php5-xml
Mod Date: Feb 15 2007

SimpleXML: Namespaces

simplexml/sxe_ns_books.php

```
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
    <bk:price>12.99</bk:price>
  </bk:book>

  <book qty="33" xmlns="http://www.example.com/ExteralClassics">
    <name>To Kill a Mockingbird</name>
    <price>10.99</price>
  </book>
</books>

$books = simplexml_load_file('sxe_ns_books.xml');

foreach ($books->book AS $book) {
  if ($book->name)
    print $book->name;
}
```

SimpleXML: Namespaces Results

To Kill a Mockingbird

Question:

What happened to the other book title?

SimpleXML: Namespaces

simplexml/sxe_ns_books_prefix.php

```
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
    <bk:price>12.99</bk:price>
  </bk:book>

  <book qty="33" xmlns="http://www.example.com/ExteralClassics">
    <name>To Kill a Mockingbird</name>
    <price>10.99</price>
  </book>
</books>

$books = simplexml_load_file('sxe_ns_books.xml');

$children = $books->children("http://www.example.com/book");
foreach ($children->book AS $book) {
        print $book->name."\n";
}
```

SimpleXML: Namespaces Results

Grapes of Wrath

Only book elements within the specified namespace are used in the results

SimpleXML: Xpath

simplexml/sxe_xpath_ns.php

```
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
    <bk:price>12.99</bk:price>
  </bk:book>
  <book qty="33" xmlns="http://www.example.com/ExteralClassics" xmlns:pc="urn::pricing">
    <name>To Kill a Mockingbird</name>
    <pc:price>10.99</pc:price>
  </book>
</books>
```

```
$sxe = simplexml_load_file('sxe_ns_books.xml');

$nodelist = $sxe->xpath("//name");
print "Book Title: ".$nodelist[0]."\n";

$nodelist = $sxe->xpath("//bk:name");
print "Book Title: ".$nodelist[0]."\n";
```

SimpleXML: XPath Results

Book Title:

Book Title: Grapes of Wrath

Why doesn't the title "To Kill a Mockingbird" get printed?

SimpleXML: Xpath

simplexml/sxe_xpath_ns_def.php

```
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
    <bk:price>12.99</bk:price>
  </bk:book>
  <book qty="33" xmlns="http://www.example.com/ExteralClassics" xmlns:pc="urn::pricing">
    <name>To Kill a Mockingbird</name>
    <pc:price>10.99</pc:price>
  </book>
</books>
```

```
$sxe = simplexml_load_file('sxe_ns_books.xml');

$sxe->registerXPathNamespace("ec", "http://www.example.com/ExteralClassics");

$nodelist = $sxe->xpath("//ec:name");
print "Book Title: ".$nodelist[0]."\n";
```

Book Title: To Kill a Mockingbird

SimpleXML: XPath Context

simplexml/sxe_xpath_context.php

```
$xml = <<< EOXML
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <book qty="33" xmlns="http://www.example.com/ExteralClassics" xmlns:pc="urn::pricing">
    <name>To Kill a Mockingbird</name>
    <pc:price>10.99</pc:price>
  </book>
</books>
EOXML;

libxml_use_internal_errors(TRUE);

$books = simplexml_load_string($xml);
$books->registerXPathNamespace("pc", "urn::pricing");

$nodelist = $books->xpath("//ec:name");
$pricing = $books->book->xpath("./pc:price");
print "Book Price: $".$pricing[0]."\n";
```

RESULTS

Book Price: $

SimpleXML: XPath Context

simplexml/sxe_xpath_ns_context.php

```
$xml = <<< EOXML
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <book qty="33" xmlns="http://www.example.com/ExteralClassics" xmlns:pc="urn::pricing">
    <name>To Kill a Mockingbird</name>
    <pc:price>10.99</pc:price>
  </book>
</books>
EOXML;
$books = simplexml_load_string($xml);

$book = $books->book[0];          /* GOTCHA TO BE AWARE OF*/
$book->registerXPathNamespace("pc", "urn::pricing");

$pricing = $book->xpath("./pc:price");

print "Book Price: $".$pricing[0]."\n";
```

RESULTS

Book Price: $10.99

SimpleXML: Write some data

```
$data = "<root><top><![CDATA[ Some text]]></top></root>";

/* We can even pass libxml parser options */
$sxe = new SimpleXMLElement($data, LIBXML_NOCDATA);

$sxe->child = 'new child';
$sxe->child2 = 'second child';

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

```
<?xml version="1.0"?>
<root><top> Some text</top><child>new
child</child><child2>second child</child2></root>
```

SimpleXML: Reading Data can Create Data?

ISSUE IS RESOLVED IN PHP 5.2.1+

```
$sxe = new SimpleXMLElement("<root />");

foreach ($sxe->childr AS $child) {
   var_dump($child);
}
print $sxe->asXML();
```

????? RESULTS ?????

```
<?xml version="1.0"?>
<root><childr/></root>
```

SimpleXML: Advanced Editing

simplexml/editing.php

```
$data = array(array('title'=>'Result 1', 'descript'=>'Res1 description'),
              array('title'=>'Result 2', 'descript'=>'description of Res2'),
              array('title'=>'Result 3', 'descript'=>'This is result 3'));

class webservice extends simpleXMLElement {
   public function appendElement($name, $value=NULL) {
      $count = (isset($this->{$name}))?count($this->{$name}):0;
      if ($count) {
         $this->{$name}[$count] = $value;
      } else {  /* An issue requires first child  be created without offset */
         $this->{$name} = $value;
      }
      return $this->{$name}[$count];
} }

$rest = simplexml_load_string('<results num="0" />', 'webservice');
$rest['num'] = count($data);

foreach ($data AS $result_item) {
   $result = $rest->appendElement('result');
   $result->appendElement('title', $result_item['title']);
   $result->appendElement('description', $result_item['descript']);
}

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

SimpleXML: Advanced Editing Results

```
<?xml version="1.0"?>
<results num="3">
    <result>
        <title>Result 1</title>
        <description>Res1 description</description>
    </result>
    <result>
        <title>Result 2</title>
        <description>description of Res2</description>
    </result>
    <result>
        <title>Result 3</title>
        <description>This is result 3</description>
    </result>
</results>
```

SimpleXML: Advanced Editing PHP 5.1.3

simplexml/editing_php513.php

```
$data = array(array('title'=>'Result 1', 'descript'=>'Res1 description'),
              array('title'=>'Result 2', 'descript'=>'description of Res2'),
              array('title'=>'Result 3', 'descript'=>'This is result 3'));

$rest = simplexml_load_string('<results num="0" />');
$rest['num'] = count($data);

foreach ($data AS $result_item) {
   $result = $rest->addChild('result');
   $result->addChild('title', $result_item['title']);
   $result->addChild('description');
   $result->description = $result_item['descript'];
}

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

SimpleXML: Removing data

remove_data.php

```
<?php
$results = simplexml_load_file('editing_php513.xml');

/* Delete title from first result element */
unset($results->result->title);

/* Delete the 2nd result element - WORKS in PHP 5.1.3+ */
unset($results->result[1]);

print $results->asXML();
?>
```

SimpleXML: Removing data

RESULTS

```
<?xml version="1.0"?>
<results num="3">
  <result>
    <description>Res1 description</description>
  </result>
  <result>
    <title>Result 3</title>
    <description>This is result 3</description>
  </result>
</results>
```

SDO (SDO_DAS_XML)

- Provides simple access to XML documents
- Works in a similar fashion to SimpleXML
- Operates only on elements and attributes
- Contains XPath support
- Allows for modifications to the XML
- Requires XML Schema
- Stricter than SimpleXML
- Maps XML to Data Types

SDO: Consuming Yahoo ContextSearch

sdo/yahoo_search_results.xml

```
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    totalResultsReturned="3" firstResultPosition="1">

  <Result>
    <Title>Zend Technologies - PHP 5 . . .</Title>
    <Summary>titletext = $dom-&amp;gt;createText. . .</Summary>
    <Url>http://www.zend.com/php5/articles/php5-x. . .</Url>
    <ClickUrl>http://uk.wrs.yahoo.com/_ylt=...</ClickUrl>
    <ModificationDate>1171872000</ModificationDate>
    <MimeType>text/html</MimeType>
    <!-- Abbreviated Content -->
  </Result>

  <!-- Abbreviated Response -->
</ResultSet>
```

SDO: Consuming Yahoo ContextSearch

sdo/yahoo_search.php

```
/* URL to Web Search service */
$url = 'http://search.yahooapis.com/WebSearchService/V1/contextSearch';
$url .= '?query='.rawurlencode('php5 xml')."&appid=zzz&results=3&language=en&context=internet";

$xsd = "http://search.yahooapis.com/WebSearchService/V1/WebSearchResponse.xsd";
$xmldas = SDO_DAS_XML::create($xsd);

$document = $xmldas->loadFile($url);

$sxe = $document->getRootDataObject();

/* Check for number of results returned */
if ((int)$sxe['totalResultsReturned'] > 0) {

   /* Loop through results and print title, url and modification date */
   foreach ($sxe->Result AS $result) {
      print 'Title: '.$result->Title."\n";
      print 'Url: '.$result->Url."\n";
      print 'Mod Date: '.date ('M d Y', (int)$result->ModificationDate)."\n\n";
   }
}
```

SDO: Consuming Yahoo ContextSearch

sdo/yahoo_search2.php

```
/* URL to Web Search service */
$url = 'http://search.yahooapis.com/WebSearchService/V1/contextSearch';
$url .= '?query='.rawurlencode('php5 xml')."&appid=zzz&results=3&language=en&context=internet";

$xsd = "http://search.yahooapis.com/WebSearchService/V1/WebSearchResponse.xsd";
$xmldas = SDO_DAS_XML::create($xsd);

$document = $xmldas->loadFile($url);

$sxe = $document->getRootDataObject();

/* Check for number of results returned */
if ($sxe->totalResultsReturned > 0) {

   /* Loop through results and print title, url and modification date */
   foreach ($sxe->Result AS $result) {
      print 'Title: '.$result->Title."\n";
      print 'Url: '.$result->Url."\n";
      print 'Mod Date: '.date ('M d Y', $result->ModificationDate)."\n\n";
   }
}
```

SDO: Consuming Yahoo ContextSearch

RESULTS

Title: Zend Technologies - PHP 5 In Depth - XML in PHP 5 - What's New?
Url: http://www.zend.com/php5/articles/php5-xmlphp.php
Mod Date: Feb 19 2007

Title: Zend Developer Zone | XML in PHP 5 - What's New?
Url: http://devzone.zend.com/node/view/id/1713
Mod Date: Feb 26 2007

Title: Parsing XML using PHP5
Url: http://www.developertutorials.com/tutorials/php/parsing-xml-using-php5-050816/page1.html
Mod Date: Feb 02 2007

SDO: XML Editing

sdo/address.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:addr="urn::addressNS" targetNamespace="urn::addressNS"
      elementFormDefault="qualified">

  <complexType name="AddressType">
    <sequence>
        <element name="street" type="string"/>
        <element name="city" type="string"/>
        <element name="state" type="string"/>
        <element name="zip" type="string"/>
    </sequence>
  </complexType>

  <element name="address" type="addr:AddressType" />

</schema>
```

SDO: XML Editing

sdo/editing.php

```
$xmldas = SDO_DAS_XML::create("address.xsd");
$xdoc = $xmldas->createDocument();
$address = $xdoc->getRootDataObject();

$address->street = "123 My Street";
$address->zip = 12345;
$address->state = 'ME';
$address->city = 'Portland';

print $xmldas->saveString($xdoc, 5);
```

```
<?xml version="1.0" encoding="UTF-8"?>
<address xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:tns="urn::addressNS" xmlns="urn::addressNS">
    <street>123 My Street</street>
    <city>Portland</city>
    <state>ME</state>
    <zip>12345</zip>
</address>
```

SDO: XML Editing

sdo/editing_error.php

```
try {
    $xmldas = SDO_DAS_XML::create("address.xsd");
    $xdoc = $xmldas->createDocument("address");
    $address = $xdoc->getRootDataObject();

    $address->city = "Portland";
    $address->company = "Local Thunder";

    print($xmldas->saveString($xdoc, 5));
} catch (SDO_Exception $e) {
    print "Error: ".$e->getMessage()."\n";
}
```

Error: Cannot find property:company

DOM

- Tree based parser
- Allows for creation and editing of XML documents
- W3C Specification with DOM Level 2/3 compliancy
- Provides XPath support
- Provides XInclude Support
- Ability to work with HTML documents
- Zero copy interoperability with SimpleXML
- Replacement for ext/domxml from PHP 4

DOMNode Classes

- DOMDocument
- DOMElement
- DOMAttr
- DOMComment
- DOMDocumentType
- DOMNotation
- DOMEntity
- DOMEntityReference
- DOMProcessingInstruction
- DOMNameSpaceNode
- DOMDocumentFragment
- DOMCharacterData
- DOMText
- DOMCdataSection

Additional Classes

- DOMNodeList
- DOMNamedNodeMap
- DOMXPath
- DOMException
- DOMImplementation

DOM: Sample Document

```
<courses>
    <course cid="c1">
        <title>Basic Languages</title>
        <description>Introduction to Languages</description>
        <credits>1.5</credits>
        <lastmodified>2004-09-01T11:13:01</lastmodified>
    </course>
    <course cid="c2">
        <title>French I</title>
        <description>Introduction to French</description>
        <credits>3.0</credits>
        <lastmodified>2005-06-01T14:21:37</lastmodified>
    </course>
    <course cid="c3">
        <title>French II</title>
        <description>Intermediate French</description>
        <credits>3.0</credits>
        <lastmodified>2005-03-12T15:45:44</lastmodified>
    </course>
</courses>
```

DOM:Document Navigation

dom/navigate-2.php

```
<?php
$dom = new DOMDocument();
$dom->load('course.xml');

$nodelist = $dom->getElementsByTagName('description');

foreach ($nodelist AS $key=>$node) {
   print "#$key: ".$node->nodeValue."\n";
}
?>
```

Results:

```
#0: Introduction to Languages
#1: Introduction to French
#2: Intermediate French
```

DOM: Navigation Optimized

dom/navigate-optimized.php

```
function locateDescription($node) {
  while($node) {
    if ($node->nodeType == XML_ELEMENT_NODE &&
      $node->nodeName == 'description') {
        $GLOBALS['arNodeSet'][] = $node;
        return;
    }
    locateDescription($node->firstChild);
    $node = $node->nextSibling;
  }
}

$dom = new DOMDocument();
$dom->load('course.xml');
$root = $dom->documentElement;
$arNodeSet = array();

locateDescription($root->firstChild);

foreach ($arNodeSet AS $key=>$node) {
  print "#$key: ".$node->nodeValue."\n";
}
```

DOM: Creating a Simple Tree

dom/create_simple_tree.php

```
$doc = new DOMDocument();

$root = $doc->createElement("tree");
$doc->appendChild($root);

$root->setAttribute("att1", "att1 value");

$attr2 = $doc->createAttribute("att2");
$attr2->appendChild($doc->createTextNode("att2 value"));
$root->setAttributeNode($attr2);

$child = $root->appendChild(new DOMElement("child"));

$comment = $doc->createComment("My first Document");
$doc->insertBefore($comment, $root);

$pi = $doc->createProcessingInstruction("php", 'echo "Hello World!"');
$root->appendChild($pi);

$cdata = $doc->createCdataSection("special chars: & < > '");
$child->appendChild($cdata);
```

DOM: Simple Tree Output

```
<?xml version="1.0"?>
<!--My first Document-->
<tree att1="att1 value" att2="att2 value">
  <child><![CDATA[special chars: & < > ']]></child>
<?php echo "Hello World!"?>
</tree>
```

DOM: Modification

Load with User Stream

```
class MyStreamWrapper {
    protected $stream = '<!--My first Document--><tree att1="att1 value" att2="att2 value">
      <child><![CDATA[special chars: & < > ]]></child><?php echo "Hello World!"?></tree>';

    public function stream_open($path, $mode, $options, $opened_path) {
        $this->position = 0;
        $this->len = strlen($this->stream);
        return true;
    }

    public function stream_read($count) {
         $ret = substr($this->stream, $this->position, $count);
         $this->position += strlen($ret);
         return $ret;
    }

    public function stream_close() { }
    public function stream_eof() { return $this->position >= $this->len; }
    public function stream_stat() { return $this->len; }
    public function url_stat($path) { return array($this->len); }
}
stream_wrapper_register('myStream', 'MyStreamWrapper');
```

DOM: Document Modification

dom/modify.php

```
<!--My first Document-->
<tree att1="att1 value" att2="att2 value"><child><![CDATA[special chars: & < > ]]></child>
<?php echo "Hello World!"?>
</tree>

$doc = new DOMDocument();
$doc->load('myStream://myData');
$tree = $doc->documentElement;

$tree->setAttribute("att1", "new val");
$tree->removeAttribute("att2");

foreach ($tree->childNodes AS $child) {
  if ($child->nodeName == 'child') {
    $child->replaceChild(new DOMText("new Content"), $child->firstChild);
    break;
  }
}

print $doc->saveXML();
```

```
<!--My first Document-->
<tree att1="new val"> <child>new Content</child><?php echo "Hello World!"?></tree>
```

DOM: Document Modification

```
/* Remove all children of an element */

while ($entry->hasChildNodes()) {
  $entry->removeChild($entry->firstChild);
}
```

OR

```
$node = $entry->lastChild;
while($node) {
  $prev = $node->previousSibling;
  $entry->removeChild($node);
  $node = $prev;
}
```

/* This Will Not Work! */

```
foreach($entry->childNodes AS $node) {
  $entry->removeChild($node);
}
```

```
/* These will work */
$children = $entry->childNodes;
$length = $children->length - 1;

for ($x=$length; $x >=0; $x--) {
  $entry->removeChild($children->item($x));
}
```

OR

```
$elem = $entry->cloneNode(FALSE);
$entry->parentNode->replaceChild($elem,
                                 $entry);
```

DOM and Namespaces

```
<xsd:complexType
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        name="ArrayOfint">
  <xsd:complexContent>
    <xsd:restriction base="soapenc:Array">
      <xsd:attribute ref="soapenc:arrayType"
                     wsdl:arrayType="xsd:int[ ]"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Dom and Namepsaces

dom/namespace.php

```
define("SCHEMA_NS", "http://www.w3.org/2001/XMLSchema");
define("WSDL_NS", "http://schemas.xmlsoap.org/wsdl/");
$dom = new DOMDocument();

$root = $dom->createElementNS(SCHEMA_NS, "xsd:complexType");
$dom->appendChild($root);

$root->setAttributeNS("http://www.w3.org/2000/xmlns/", "xmlns:wsdl", WSDL_NS);
$root->setAttribute("name", "ArrayOfint");

$content = $root->appendChild(new DOMElement("xsd:complexContent", NULL,
    SCHEMA_NS));

$restriction = $content->appendChild(new DOMElement("xsd:restriction", NULL,
    SCHEMA_NS));
$restriction->setAttribute("base", "soapenc:Array");

$attribute = $restriction->appendChild(new DOMElement("xsd:attribute", NULL,
    SCHEMA_NS));
$attribute->setAttribute("ref", "soapenc:arrayType");

$attribute->setAttributeNS(WSDL_NS, "wsdl:arrayType", "xsd:int[]");
```

DOM and Xpath

dom/xpath/dom-xpath.xml

```
<books xmlns="http://www.example.com/books"
          xmlns:bk="http://www.example.com/book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
    <bk:price>12.99</bk:price>
  </bk:book>

  <book qty="33" xmlns="http://www.example.com/ExteralClassics">
    <name>To Kill a Mockingbird</name>
    <price>10.99</price>
  </book>
</books>
```

DOM and Xpath

dom/xpath/dom-xpath.php

```
<books xmlns="http://www.example.com/books" xmlns:bk="http://www.example.com/book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name></bk:book>
  <book qty="33" xmlns="http://www.example.com/ExteralClassics">
    <name>To Kill a Mockingbird</name></book>
</books>
```

```
$xpath = new DOMXPath($doc);

$nodelist = $xpath->query("//bk:name");
print "Book Title: ".$nodelist->item($nodelist->length - 1)->textContent."\n";

$inventory = $xpath->evaluate("sum(//@qty)");
print "Total Books: ".$inventory."\n";

$xpath->registerNameSpace("ec", "http://www.example.com/ExteralClassics");
$nodelist = $xpath->query("//ec:book");

$book = $nodelist->item(0);
$inventory = $xpath->evaluate("sum(./@qty)", $book);
print "Total Books: ".$inventory."\n";
```

DOM and Xpath Results

Book Title: Grapes of Wrath
Total Books: 58
Total Books: 33

Performing Validation

dom/validation/validate.php

```
$doc = new DOMDocument();
print "DTD Validation:\n";
$doc->load('courses-dtd.xml', LIBXML_DTDVALID);
/* No errors means document is valid */

if ($doc->validate()) {    print "   Document Is Valid\n";    }

print "DTD Validation FAILURE:\n";
$doc->load('course-id.xml');
if ($doc->validate()) {     print "   Document Is Valid\n";    }

$doc->load('course.xml');
print "\nXML Schema Validation:\n";
if ($doc->schemaValidate('course.xsd')) {    print "   Document is valid\n";    }

$doc->load('course.xml');
print "\nRelaxNG Validation:\n";
if ($doc->relaxNGValidate('course.rng')) {    print "   Document is valid\n";
        }
```

Performing Validation Results

DTD Validation:
Document Is Valid

DTD Validation FAILURE:
Warning: DOMDocument::validate(): No declaration for element courses in /home/rrichards/workshop/dom/validation/validate.php on line 11
Warning: DOMDocument::validate(): No declaration for element course in /home/rrichards/workshop/dom/validation/validate.php on line 11
Warning: DOMDocument::validate(): No declaration for element title in /home/rrichards/workshop/dom/validation/validate.php on line 11
. . .

XML Schema Validation:
Document is valid

RelaxNG Validation:
Document is valid

Extending DOM Classes

- Overriding the constructor requires the parent constructor to be called.
- Properties built into the DOM classes cannot be overridden.
- Methods built into the DOM classes may can be overridden.
- The lifespan of an extended object is that of the object itself.

Extending DOM Classes

dom/extending/extending.php

```
class customElement extends DOMElement { }

class customDoc extends DOMDocument {
   public $nodeName = "customDoc";

   function __construct($rootName) {
      parent::__construct();
      if (! empty($rootName))
          $element = $this->appendChild(new DOMElement($rootName));
   }

   function createElement($name, $value, $parent=NULL) {
      $custom = new customElement($name, $value);
      if ($parent && ($parent instanceof DOMElement)) {
          $parent->appendChild($custom); }
      return $custom;
   }
}

$myc = new customDoc("root");
$myelement = $myc->createElement("myname", "value", $myc->documentElement);
if ($myelement instanceof customElement) {    print "This is a customElement\n";    }

print $myc->nodeName."\n";
print $myc->saveXML();
```

DOM Object Scope

dom/extending/object_scope.php

```
class customElement extends DOMElement { }

function changeit($doc) {
    $myelement = new customElement("custom", "element2");
     $doc->replaceChild($myelement, $doc->documentElement);
     print "Within changeit function: ".get_class($doc->documentElement)."\n";
}

$doc = new DOMDocument();

$myelement = $doc->appendChild(new customElement("custom", "element"));
print "After Append: ".get_class($myelement)."\n";

unset($myelement);
print "After unset: ".get_class($doc->documentElement)."\n";

changeit($doc);
print "Outside changeit(): ".get_class($doc->documentElement)."\n";
```

```
After Append: customElement
After unset: DOMElement
Within changeit function: customElement
Outside changeit(): DOMElement
```

DOM: registerNodeClass

dom/extending/register_node_class.php

```
class customElement extends DOMElement { }
function changeit($doc) {
    $myelement = new DOMElement("custom", "element2");
      $doc->replaceChild($myelement, $doc->documentElement);
      print "Within changeit function: ".get_class($doc->documentElement)."\n";
}

$doc = new DOMDocument();
$doc->registerNodeClass('DOMElement', 'customElement');

$myelement = $doc->appendChild($doc->createElement("custom", "element"));
print "After Append: ".get_class($myelement)."\n";

unset($myelement);
print "After unset: ".get_class($doc->documentElement)."\n";

changeit($doc);
print "Outside changeit(): ".get_class($doc->documentElement)."\n";
```

```
After Append: customElement
After unset: customElement
Within changeit function: DOMElement
Outside changeit(): customElement
```

DOM:Common Issues

- DOM Objects and Sessions
- Removing Nodes while iterating a Nodeset skips nodes
- XML Tree contains garbled characters
- Extended class is not returned from property or method
- Elements not being returned by ID
- Entity errors are issues when loading a document
- New DTD is not recognized by document

XMLReader

- Forward moving stream based parser
- It is a Pull parser
- Based on the C# XmlTextReader API
- Advantages:
  - Low memory footprint
  - Namespace support
  - Simple API
  - Validation support
  - Advanced Feature Set
  - Faster Processing

XMLReader: Simple Example

xmlreader/reader_simple.xml

```
<?xml version='1.0'?>
<chapter xmlns:a="http://www.example.com/namespace-a"
         xmlns="http://www.example.com/default">
  <a:title>XMLReader</a:title>
  <para>
    First Paragraph
  </para>
  <a:section a:id="about">
    <title>About this Document</title>
    <para>
      <!-- this is a comment -->
      <?php echo 'Hi!  This is PHP version ' . phpversion(); ?>
    </para>
   </a:section>
</chapter>
```

XMLReader: Simple Example

xmlreader/reader_simple.php

```
$reader = new XMLReader();
$reader->open('reader_simple.xml');
$reader->read();

print "xmlns Attribute value: ".$reader->getAttributeNo(0)."\n\n";

while ($reader->read() && $reader->name != "a:title") { }
print "Local Name for Element: ".$reader->localName."\n";
print "Namespace URI for Element: ".$reader->namespaceURI."\n";

while($reader->read()) {
   switch ($reader->nodeType) {
       case XMLReader::ELEMENT:
           print "Element: ".$reader->name."\n";
           if ($reader->hasAttributes && $reader->moveToFirstAttribute()) {
              do {
                 print "  ".$reader->name."=".$reader->value."\n";
               } while($reader->moveToNextAttribute());
           }
           break;
       case XMLReader::PI:
          print "PI Target: ".$reader->name."\n  PI Data: ".$reader->value."\n";
   }
}
```

XMLReader: Simple Example

RESULTS

xmlns Attribute value: http://www.example.com/namespace-a

Local Name for Element: title
Namespace URI for Element: http://www.example.com/namespace-a

Element: para
Element: a:section
    a:id=about
Element: title
Element: para
PI Target: php
    PI Data: echo 'Hi! This is PHP version ' . phpversion();

XMLReader: Consuming Yahoo Shopping

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="urn:yahoo:prods"
                  xsi:schemaLocation="urn:yahoo:prods
      http://api.shopping.yahoo.com/shoppingservice/v1/productsearch.xsd"
                  totalResultsAvailable="8850" firstResultPosition="2" totalResultsReturned="2">
  <Result>
    <Catalog ID="1991433722">
      <Url><![CDATA[http://shopping.yahoo.com/p:Linksys. . .2]]></Url>
      <ProductName><![CDATA[Linksys WRT5. . .r Broadband
      Router]]></ProductName>
      <PriceFrom>69.97</PriceFrom>
      <PriceTo>89.99</PriceTo>
      <Thumbnail /><!-- child elements Url (CDATA), Height, Width -->
      <Description><![CDATA[The Wireless-G . . .ces.]]></Description>
      <Summary><![CDATA[IEEE 802.3, ...]]></Summary>
      <UserRating /><!-- Rating sub elements -->
      <SpecificationList /><!-- 0+ Specification child elements --></SpecificationList>
    </Catalog>
  </Result>
</ResultSet>
```

XMLReader: Consuming Yahoo Shopping

xmlreader/rest_yahoo_shopping.php

```
function getTextValue($reader) { ... }
function processCatalog($reader) { ... }
function processResult($reader) { ... }

/* URL to Product Search service */
$url = 'http://api.shopping.yahoo.com/ShoppingService/V1/productSearch';

/* The query is separate here as the terms must be encoded. */
$url .= '?query='.rawurlencode('linksys');

/* Complete the URL with App ID, limit to 1 result and start at second record */
$url .= "&appid=zzz&results=2&start=2";

$reader = new XMLReader();

if (! $reader->open($url)) { print "Cannot access Webservice\n"; exit; }

while($reader->name != "Result") { $reader->read(); }
do {
   processResult($reader);
} while($reader->next('Result'));
```

XMLReader: Consuming Yahoo Shopping

xmlreader/rest_yahoo_shopping.php

```
function getTextValue($reader) {
    if ($reader->nodeType != XMLReader::ELEMENT || $reader->isEmptyElement
            || ($reader->read() && $reader->nodeType == XMLReader::END_ELEMENT))
              return;
    $retVal = $reader->value;
    $reader->read();
    return $retVal;
}

function processResult($reader) {
    $depth = $reader->depth;
    if ($reader->isEmptyElement || ($reader->read() &&
                              $reader->nodeType == XMLReader::END_ELEMENT))
              return;

     while($depth < $reader->depth && $reader->name != "Catalog") { $reader->read();
      };
      processCatalog($reader);
      /* Read until </Result> is encountered */
      while($depth < $reader->depth) { $reader->read(); }
}
```

XMLReader: Consuming Yahoo Shopping

xmlreader/rest_yahoo_shopping.php

```
function processCatalog($reader) {
    $depth = $reader->depth;
    print "Catalog ID".$reader->getAttribute('ID')."\n";

    if ($reader->isEmptyElement || ($reader->read() &&
                $reader->nodeType == XMLReader::END_ELEMENT))
          return;
    while($depth < $reader->depth) {
       switch ($reader->name) {
           case "ProductName":
           case "PriceFrom":
           case "PriceTo":
           case "Description":
           case "Url":
                    print $reader->name.": ".getTextValue($reader)."\n";
       }
       $reader->next();
   }
}
```

XMLReader: Consuming Yahoo Shopping

RESULTS (Abbreviated)

Catalog ID1990338714
Url:
http://shopping.yahoo.com/p:Linksys%20Instant%20Broadband%20EtherFast%20Cable%2FDSL%20Router:1990338714
ProductName: Linksys Instant Broadband EtherFast Cable/DSL Router
PriceFrom: 39.99
PriceTo: 72.71
Description: <P>Linksys, a provider of networking hardware for the small/medium business (SMB), small office/home office (SOHO), and enterprise markets and broadband networking hardware for the home, has announced the new EtherFast Cable/DSL Router. The first in the new Instant Broadband series, this Linksys broadband router will enable home or office users to connect their computers to a cable or DSL modem and securely share Internet access and perform networking tasks such as file and printer sharing. The built-in hardware firewall gives users the security of sharing files without fear of intruders hacking into the network. </P>

XMLReader: DTD Validation

xmlreader/validation/reader.xml

```
<!DOCTYPE chapter [
<!ELEMENT chapter (title, para, section)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT para ANY>
<!ATTLIST para name CDATA "default">
<!ELEMENT section (#PCDATA)>
<!ATTLIST section id ID #REQUIRED>
]>
<chapter>
  <title>XMLReader</title>
  <para>
    First Paragraph
  </para>
  <section id="about">
    <title>About this Document</title>
    <para>content</para>
   </section>
</chapter>
```

XMLReader: DTD Validation

xmlreader/validation/reader.php

```
$objReader = XMLReader::open('reader.xml');
$objReader->setParserProperty(XMLReader::VALIDATE, TRUE);

/* As of PHP 5.2 LIBXML Parser Options may be passed */
// $objReader = XMLReader::open('reader.xml', NULL,
                                              LIBXML_DTDVALID);

libxml_use_internal_errors(TRUE);

while ($objReader->read()) {
  if (! $objReader->isValid()) {
    print "NOT VALID\n";
    break;
  }
}

$arErrors = libxml_get_errors();
foreach ($arErrors AS $xmlError) {
  print $xmlError->message;
}
```

XMLReader: DTD Validation

RESULTS

PHP Strict Standards: Non-static method XMLReader::open() should not be called statically in /home/rrichards/workshop/xmlreader/validation/reader.php on line 2

NOT VALID
Element section was declared #PCDATA but contains non text nodes

XMLReader: Relax NG Validation

xmlreader/validation/reader.rng

```
<?xml version="1.0" encoding="utf-8" ?>
<element name="chapter"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <element name="title">
    <text/>
  </element>
  <element name="para">
    <text/>
  </element>
  <element name="section">
    <attribute name="id" />
    <text/>
  </element>
</element>
```

XMLReader: Relax NG Validation

xmlreader/validation/reader-rng.php

```
$objReader = XMLReader::open('reader.xml');
$objReader->setRelaxNGSchema('reader.rng');

libxml_use_internal_errors(TRUE);

while ($objReader->read()) {
   if (! $objReader->isValid()) {
         print "NOT VALID\n";
         break;
   }
}

$arErrors = libxml_get_errors();
foreach ($arErrors AS $xmlError) {
      print $xmlError->message;
}
```

XMLReader: Relax NG Validation

RESULTS

PHP Strict Standards: Non-static method XMLReader::open() should not be called statically in /home/rrichards/workshop/xmlreader/validation/reader.php on line 2

NOT VALID
Did not expect element title there

XMLReader: XML Schema Validation

xmlreader/validation/reader.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="chapter">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="para" type="xsd:string"/>
        <xsd:element name="section">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="id" type="xsd:ID"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

XMLReader: XML Schema Validation

xmlreader/validation/reader-xsd.php

```
$objReader = XMLReader::open('reader.xml');
$objReader->setSchema('reader.xsd');

libxml_use_internal_errors(TRUE);

while ($objReader->read()) {
   if (! $objReader->isValid()) {
         print "NOT VALID\n";
         break;
   }
}

$arErrors = libxml_get_errors();
foreach ($arErrors AS $xmlError) {
      print $xmlError->message;
}
```

LIBXML2-2.20+ REQUIRED

XMLReader: XML Schema Validation

RESULTS

PHP Strict Standards: Non-static method XMLReader::open() should not be called statically in /home/rrichards/workshop/xmlreader/validation/reader.php on line 2

NOT VALID
Element 'para', attribute 'name': The attribute 'name' is not allowed.
Element 'section': Element content is not allowed, because the content type is a simple type definition.

Tree Parsers

- Pros:
  - Full navigation and modification of the XML document
  - Navigating and searching are extremely fast once the tree is loaded into memory
- Cons:
  - Must wait until entire tree is loaded to begin working with the XML.
  - Memory intensive

Streaming Parsers

- Pros:
  - Uses minimal memory
  - Processing takes place immediately while the document is parsed
- Cons:
  - Minimal to no navigation support (forward only)
  - No document editing capabilities

Raw Test Data

```
<books>
  <book id="1"><title>1</title><pages>1</pages></book>
  <book id="2"><title>2</title><pages>2</pages></book>
  <!-- Remaining book elements for a total of 200,000 -->
</books>
```

Memory Usage:

DOM	SimpleXML	ext/xml	XMLReader
85.6MB	85.6MB	26KB	177KB

Using every optimization possible the following results show the time in seconds to locate the book element having id="5000".

Average Time in Seconds for Optimized Search for an Element:

DOM	SimpleXML	ext/xml	XMLReader
6.623	6.583	0.930	0.238

XMLWriter

- Lightweight and forward-only API for generating well formed XML
- Automatically escapes data
- Works with PHP 4.3+ available at http://pecl.php.net/package/xmlwriter
- Object Oriented API available for PHP 5+
- Part of core PHP distribution since PHP 5.1.2

XMLWriter: Simple Example

xmlwriter/simple.php

```
<?php
$xw = new XMLWriter();
$xw->openMemory();

/* Turn on indenting to make output look pretty and set string
   used for indenting as teh default space is too short*/
$xw->setIndent(TRUE);
$xw->setIndentString('   ');

/* Write out the optional XML declaration only specifying version */
$xw->startDocument('1.0');

/* Create the opening document element, which is namespaced */
$xw->startElementNs(NULL, "chapter", "http://www.example.com/default");

/* Write out an xml namespace declaration that is used later in the document */
$res = $xw->writeAttribute('xmlns:a', 'http://www.example.com/namespace-a');

/* Write complete elements with text content */
$xw->writeElement('a:title', 'XMLReader');
$xw->writeElement('para', 'spec chars < > & " inside para element');
```

XMLWriter: Simple Example

xmlwriter/simple.php

```
/* start an element and add an attribute to it */
$xw->startElement('a:section');
$xw->writeAttribute('a:id', 'about');

/* Write out an element with special characters */
$xw->writeElement('title', 'Pro PHP XML & Webservices');

$xw->startElement('para'); /* This opens the para element */

$xw->writeComment("this is a comment");
$xw->text("          ");
$xw->writePi("php", "echo 'Hi!  This is PHP version ' . phpversion(); ");
$xw->text("\n      ");

$xw->endElement(); /* This will close the open para element */

$xw->endDocument();

/* Flush and clear the buffer */
echo $xw->flush(true);
?>
```

XMLWriter: Simple Example

xmlwriter/simple.php

```
/* start an element and add an attribute to it */
$xw->startElement('a:section');
$xw->writeAttribute('a:id', 'about');

/* Write out an element with special characters */
$xw->writeElement('title', 'Pro PHP XML & Webservices');

$xw->startElement('para'); /* This opens the para element */

$xw->writeComment("this is a comment");
$xw->text("          ");
$xw->writePi("php", "echo 'Hi!  This is PHP version ' . phpversion(); ");
$xw->text("\n      ");

$xw->endElement(); /* This will close the open para element */

$xw->endDocument();

/* Flush and clear the buffer */
echo $xw->flush(true);
?>
```

XMLWriter: Simple Example
Results

```
<?xml version="1.0"?>
<chapter xmlns="http://www.example.com/default"
         xmlns:a="http://www.example.com/namespace-a">
  <a:title>XMLReader</a:title>
  <para>spec chars &amp;lt; &amp;gt; &amp;amp; &amp;quot; inside para
element</para>
  <a:section a:id="about">
    <title>Pro PHP XML &amp;amp; Webservices</title>
    <para>
      <!--this is a comment-->
      <?php echo 'Hi!  This is PHP version ' . phpversion(); ?>
    </para>
  </a:section>
</chapter>
```

XSL

- Used to transform XML data
- XSLT based on XPath
- Works with DOM and SimpleXML, although the DOM extension is required.
- Provides the capability of calling PHP functions during a transformation
- DOM nodes may be returned from PHP functions
- The LIBXML_NOCDATA and LIBXML_NOENT constants are your friends.
- libxslt 1.1.5+ is recommended to avoid problems when using xsl:key

XSL: XML Input Data

xsl/sites.xml

```
<?xml version="1.0"?>
<sites>
  <site xml:id="php-gen">
    <name>PHP General</name>
    <url>http://news.php.net/group.php?group=php.general&amp;format=rss</url>
  </site>
  <site xml:id="php-pear">
    <name>PHP Pear Dev</name>
    <url>http://news.php.net/group.php?group=php.pear.dev&amp;format=rss</url>
  </site>
  <site xml:id="php-planet">
    <name>Planet PHP</name>
    <url>http://www.planet-php.org/rss/</url>
  </site>
</sites>
```

XSL: Simple Transformation

xsl/simple_stylesheet.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <body>
        <xsl:apply-templates select="/sites/site"/>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="/sites/site">
    <p><xsl:value-of select="./name"/> : <xsl:value-of select="./url"
                                                       disable-output-escaping="yes"/></p>
  </xsl:template>
</xsl:stylesheet>
```

XSL: Simple Transformation

xsl/simple_stylesheet.php

```
/* Load Stylesheet */
$stylesheet = new DOMDocument();
$stylesheet->load('simple_stylesheet.xsl');

/* Create XSL Processor */
$proc = new xsltprocessor();
$proc->importStylesheet($stylesheet);

/* Load XML Data */
$dom = new DOMDocument();
$dom->load('sites.xml');

print $proc->transformToXML($dom);
```

XSL: Simple Transformation

RESULTS

```
<html>
  <body>
    <p>PHP General :
http://news.php.net/group.php?group=php.general&format=rss</p>
    <p>PHP Pear Dev :
http://news.php.net/group.php?group=php.pear.dev&format=rss</p>
    <p>Planet PHP : http://www.planet-php.org/rss/</p>
  </body>
</html>
```

XSL: Advanced Transformation

xsl/advanced_stylesheet.php

```
function initReader($url) {
  $GLOBALS['reader'] = new XMLReader();
  if ($GLOBALS['reader']->open($url)) {
    while ($GLOBALS['reader']->read() && $GLOBALS['reader']->name != 'item') { }

    if ($GLOBALS['reader']->name == 'item')
      return 1;
  }
  $GLOBALS['reader'] = NULL;
  return 0;
}

function readNextItem() {
  if ($GLOBALS['reader'] == NULL)
    return NULL;
  if ($GLOBALS['beingProc'])
    $GLOBALS['reader']->next('item');
  else
    $GLOBALS['beingProc'] = TRUE;
  if ($GLOBALS['reader']->name == 'item')
      return $GLOBALS['reader']->expand();
  return NULL;
}
```

XSL: Advanced Transformation

xsl/advanced_stylesheet.php

```
$beingProc = FALSE;
$reader = NULL;

/* Load Stylesheet */
$stylesheet = new DOMDocument();
$stylesheet->load('advanced_stylesheet.xsl');

/* Create XSL Processor */
$proc = new xsltprocessor();
$proc->importStylesheet($stylesheet);

/* Load XML Data */
$dom = new DOMDocument();
$dom->load('sites.xml');

$proc->setParameter(NULL, 'siteid', 'php-gen');
$proc->registerPHPFunctions('initReader');
$proc->registerPHPFunctions('readNextItem');
print $proc->transformToXML($dom);
/* END */
```

XSL: Advanced Transformation

xsl/advanced_stylesheet.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:php="http://php.net/xsl" version="1.0">
  <xsl:output method="html"/>
  <xsl:param name="siteid" select="0" />

  <xsl:template match="/">
    <html><body>
      <xsl:apply-templates select="id($siteid)"/>
    </body></html>
  </xsl:template>

  <xsl:template match="/sites/site">
    <xsl:variable name="itemnum" select="php:functionString('initReader', ./url)" />
    <xsl:if test="number($itemnum) > 0">
      <xsl:call-template name="itemproc" />
    </xsl:if>
  </xsl:template>
```

XSL: Advanced Transformation

xsl/advanced_stylesheet.xsl

```
<xsl:template match="item">
    <p>
      Title: <b><xsl:value-of select="./title" /></b><br/><br/>
      URL: <xsl:value-of select="./link" /><br/>
      Published: <xsl:value-of select="./pubDate" /><br/>
    </p>
 </xsl:template>

 <xsl:template name="itemproc">
  <xsl:variable name="nodeset" select="php:functionString('readNextItem')" />
  <xsl:if test="boolean($nodeset)">
      <xsl:apply-templates select="$nodeset"/>
      <xsl:call-template name="itemproc" />
  </xsl:if>
 </xsl:template>

</xsl:stylesheet>
```

XSL: Advanced Transformation

Results viewed through a browser
xsl/advanced_stylesheet.html

Title: Re: Spreadsheet Writer

URL: http://news.php.net/php.general/241446
Published: Thu, 07 Sep 2006 13:52:09 –0400

Title: Re: Spreadsheet Writer

URL: http://news.php.net/php.general/241447
Published: Thu, 07 Sep 2006 13:52:09 -0400

Title: Re: Spreadsheet Writer

URL: http://news.php.net/php.general/241448
Published: Thu, 07 Sep 2006 13:52:09 -0400

Questions?

rrichards@ctindustries.net

http://www.cdatazone.org/talks/quebec_2007/workshop.zip