XML

Lecture 14, CMSC 126

John Roy Daradal

Instructor

Today's Topics

- XML Introduction
- XML in PHP
- XPath

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Lily</to>
    <from>Marshall</from>
    <subject>Proposal</subject>
    <message language="english">
        Will you marry me?
    </message>
</note>
```

XML

- "skeleton" for creating markup languages
- Used to present complex data in human-readable form
- "Self-describing data"

XML

- Data interchange format (**de facto** universal format)
- Used to store and transfer data
- Sometimes used in lieu of a database

XML is not a replacement for HTML.

HTML vs XML

- HTML is specific (web pages), XML is general.
- HTML has predefined tags, XML has none

XML Syntax

XML Syntax

```
<element attribute="value">content</element>
```

XML Syntax

XML element names

- begin with letter or underscore
- include digits, hyphens, and periods
- case sensitive (unlike HTML)

XML Syntax

- XML document must have a single root element
- Content can contain other elements (nesting)
- Define a new nested tag to provide more info about content of tag

XML Syntax

- You choose the tags and attributes that best represent the data
- No predefined tags (unlike HTML)
- Use any tags you want, but be consistent

XML Syntax

- Nested tags > attributes
- Attributes can't describe structure

Rule of thumb

- Data = Tag
- Metadata = Attribute

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Lily</to>
    <from>Marshall</from>
    <subject>Proposal</subject>
    <message language="english">
        Will you marry me?
    </message>
</note>
```

- Spot the data and metadata.

Well-Formed XML

- Single root element
- Other elements are correctly nested
- Not well-formed XML = program reports an error

XML Comments

```
<!-- This is a comment. -->
```

Exercise

Books

- The Art of Rails, Edward Benson, 2008
- A Book on C, Al Kelley, Ira Pohl, 1998
- Foundations of Security, Neil Daswani, 2007
- The Definitive Guide to Django, Adrian Holovaty and Jacob Kaplan-Moss, 2008

Solution #1

```
<books>
  <book title="The Art of Rails" author="Edward Benson"
        year ="2008" />
  <book title="A Book on C"
    author="Al Kelley and Ira Pohl" year="1998" />
  <book title="Foundations of Security"
    author="Neil Daswani" year="2007" />
  ...
</books>
```

Solution #2

```
<books>
  <book>
    <title>The Art of Rails</title>
    <author>Edward Benson</author>
    <year>2008</year>
  </book>
  <book>
    <title>A Book on C</title>
    <author>Al Kelley and Ira Pohl</author>
    <year>1998</year>
  </book>
  ...
</books>
```

Solution #3

```
<books>
  <book>
    <title>The Art of Rails</title>
    <authors>
        <author>Edward Benson</author>
    </authors>
    <year>2008</year>
  </book>
  <book>
    <title>A Book on C</title>
    <authors>
        <author>Al Kelley</author>
        <author>Ira Pohl</author>
    </authors>
    <year>1998</year>
```

Solution #4

```
  </book>
  ...
</books>

<books>
  <book>
    <title>The Art of Rails</title>
    <author>Edward Benson</author>
    <year>2008</year>
  </book>
  <book>
    <title>A Book on C</title>
    <authors>
        <author>Al Kelley</author>
        <author>Ira Pohl</author>
    </authors>
    <year>1998</year>
  </book>
  ...
```

</books>

Issues

- Multiple authors
- Order of title, author, and year
- Author name (First name, last name)

XML in PHP

XML in PHP

SimpleXML library

- provides easy way to work with XML documents
- PHP5

Example: books.xml

```
<books>
  <book isbn="21356294582">
    <title>The Art of Rails</title>
    <authors>
        <author>Edward Benson</author>
    </authors>
    <year>2008</year>
  </book>
  <book isbn="9529195961005">
    <title>A Book on C</title>
    <authors>
        <author>Al Kelley</author>
        <author>Ira Pohl</author>
    </authors>
    <year>1998</year>
```

SimpleXML

```
    </book>
    ...
</books>

//Load an XML string

$xmlstr = file_get_contents('books.xml');
$books = simplexml_load_string($xmlstr);
```

SimpleXML

```
//Load an XML file

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

SimpleXML

Object-Oriented Approach

```
$xmlstr = file_get_contents('books.xml');
$books = new SimpleXMLElement($xmlstr);

$books = new SimpleXMLElement('books.xml',NULL,true);
// Second param: additional libxml parameters
// Third param: True = first argument represents
        path to file
```

Accessing Children and Attributes

```
foreach($books->book as $book){
  echo $book->title."\n";
  echo $book['isbn']."\n";
  echo $book->year."\n";
  foreach($book->authors->children() as $author){
    echo $author."\n";
  }
}
```

More Code

```
$books->children()
$book->getName()

$book->attributes()
$attr->getName()
```

Generating XML

- Programmatically build XML string
- Save into XML file

Solution # 1

```
$xml = "";
$xml .= '<?xml version="1.0" encoding="utf-8"?>';
$xml .= '<books>';
foreach($books as $book){
    $xml .= "<title>".$book->title."</title>";
}
$xml .= '</books>';
file_put_contents('books.xml',$xml);
```

Solution # 2

```
$xml = array();
$xml[] = '<?xml version="1.0" encoding="utf-8"?>';
$xml[] = '<books>';
foreach($books as $book){
    $xml[] = "<title>".$book->title."</title>";
}
$xml[] = '</books>';
$xml = implode("\n",$xml);
file_put_contents('books.xml',$xml);
```

XPath

XPath

- Language used to navigate through elements and attributes in XML document
- "Query language"-ish

Path Expressions

- Select nodes or node-sets in XML document
- Looks like the traditional computer file system expression

Example

```
<books>
  <book category="Ruby">
    <title>The Art of Rails</title>
    <author>Edward Benson</author>
    <year>2008</year>
    <price>30.00</price>
  </book>
  <book category="C">
    <title>A Book on C</title>
    <author>Al Kelley</author>
    <year>1998</year>
    <price>5.00</price>
  </book>
  ...
</books>
```

Example

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

$results = $books->xpath('/books/book/title');
foreach($results as $title){
    echo $title."\n";
}
```

Example

```
// Get the title of the first book
$results = $books->book[0]->xpath('title');
foreach($results as $title){
    echo $title."\n";
}
# Note: XPath returns an array,
# even if only one element is returned
```

XPath Examples

```
# All book titles
/books/book/title

# Title of first book
/books/book[1]/title

# Title of last book
/books/book[last()]/title
```

XPath Examples

```
# Second to the last book
/books/book[last()-1]

# First two books
/books/book[position()<3]
```

XPath Examples

```
# All book elements with a category attribute
//book[@category]

# All books under "Ruby" category
//book[@category='Ruby']
```

XPath Examples

```
# All prices
/books/book/price/text()

# Price nodes with price > 20
/books/book[price>20]/price

# Title nodes of price > 20
/books/book[price>20]/title
```

XPath Examples

```
# All nodes with the name book
book

# All book elements that are children of books
books/book

# All book elements that are descendants of books
books//book
```

XPath Examples

```
# Wildcards
/books/*

//book[@*]
```

XPath Operators

- | OR
- AND
- = !=
- < <=
- > >=
- + - * div mod

Other Related Topics

- Document Type Definitions
- Namespaces
- Schemas
- XSLT
- Web Services
- RSS

Assignment

Write an XML file for the ff. data:

Laboratory Schedule (Section 2)

See "Exercises" section for lab materials.

Date		Activity
Jan 9	Wed	- Check MP#1 - Assignment: Proposal for Final Project
Jan 16	Wed	- HTTP Activity - HTML & CSS Quiz - Release: PHP Exercise - Deadline: Proposal for Final Project
Jan 23	Wed	- PHP Exercise - PHP Quiz - Release: MP#2 (PHP + XML)
Jan 30	Wed	- Work with PHP Exercise and MP#2

Lab Concerns

- PHP Exercise

MP#2: PHP and XML

- Sessions and Cookies
- Reading XML
- Creating XML

MP#2: Steps

- Identify all the static/hard-coded labels/headers in your Pagbutlak pages
- Create an XML file for the default language (English)

Example

```
<labels language="english">
    <home>
        <title>Pagbutlak</title>
        <subtitle>Official CAS Publication</subtitle>
        <announcements>Announcements</announcements>
        <headlines>Headlines</headlines>
    </home>
    <navigation>
        <home>Home</home>
        <archives>Archives</archives>
        ...
    </navigation>
    <archives>
        ....
    </archives>
```

</labels>

MP#2: Steps

- Remove all hardcoded labels from your Pagbutlak pages
- Replace them with dynamic data loaded from the XML file you just created

MP#2: Steps

- Provide a form where users can create a new set of labels for a new language
- The field names can be hardcoded
- Validate the data passed (no blanks)
- Upon successful submission, create a new XML file for that new language

MP#2: Steps

- Provide a page where users can choose a language

Form fields needed

- a select element containing all the languages available
- a "Remember choice" checkbox
- submit button

MP#2: Steps

- If "Remember choice" is checked, store the current language as a cookie, so that even after the browser is closed, the selected language will be remembered
- If unchecked, store the current language as a session variable.
- Upon setting the current language, the web site labels will be changed (using the current language)

References

- *Programming the World Wide Web*, 6E, R.Sebesta, 2010
- Sun Yat-Sen University, Web 2.0 Programming
- *XML Primer*, Oxford University, 2002
- *Zend PHP 5 Certification Study Guide*, Davey Shafik, 2006
- *Programming PHP*, Rasmus Lerdorf, 2002