UFCEKG-20-2
Data, Schemas & Applications

Lecture 5
XML & PHP
Richter Scales on Web 2.0

The Richter Scales
FTSE 100
Location, Location, Location

Degree Confluence

Twitter Trends
Navigating trees

Tree structures (with their representation as tree diagrams) are a key data structure which appear in many places in addition to XML

- Folder hierarchy
- Document Object Model (DOM)
- Organisational reporting structures
- Paragraph Numbering in a Document
- Natural classification of living things - Linnaeus
- Dewey Classification of knowledge
- Java Class hierarchy

Not all natural structures are pure trees with a single root and only one path from any node to the root
<?xml version="1.0" encoding="UTF-8"?>
<patient nhs-no="7503557856">
  <name>
    <first>Joseph</first>
    <middle>Michael</middle>
    <last>Bloggs</last>
    <preferred>Joe</preferred>
  </name>
  <title>Mr</title>
  <address>
    <street>2 Gloucester Road</street>
    <city>Bristol</city>
    <county>Avon</county>
    <postcode>BS2 4QS</postcode>
  </address>
  <tel>
    <home>0117 9541054</home>
    <mobile>07710 234674</mobile>
  </tel>
  <email>joe.bloggs@email.com</email>
</patient>
Example XML document deconstructed

xml declaration (optional) used by xml processor; this documents conforms to xml version 1 and uses the UTF-8 standard (Unicode optimized for ASCII)

attribute; attributes provide additional information about an element and consist of a name value pair; the value must be enclosed in a single (') or double quote ("")

root element; every well formed xml document must be enclosed by exactly one root element.

a comment; comments must be delimited by the <!-- --> characters as in xhtml

a simple element containing text

a complex element containing other elements and text

empty elements

Feb 2013 N. H. N. D. de Silva
Sample XML

```xml
<a>
  <b>fred</b>
  <c x='5'>Joe</c>
  <d>Eric
    <x>ted</x>
    <x>alice</x>
  </d>
</a>
```

**XML**

**XML as nested lists**
XML processing

Processing XML files requires a language to support operations like:

- **reading** in a XML file and converting to a internal structure
- **navigating** around the internal structure
- **creating** XML structures
Tree Navigation

Syntax for tree navigation. Getting to the node containing the text 'Alice'

- XPath: $xml/d/x[2]
- Javascript DOM: $xml.getElementsByTagName('d')[0].childNodes[1]
  (approx)
- PHP SimpleXML: $xml->d->x[1]
- Java ...
PHP : Constructing an XML object tree

To construct a tree of objects from a string

- $xmltree = new SimpleXMLElement($string);
**PHP : access to child elements**

In PHP the operator -> is used to reference from an object to its properties. The equivalent operator in Java is "." (dot) and in XPath is "/". In an XML object tree, each element is a property of its parent.

If $x$ is an element

- $x$->getName() is the name of the element - its tag
- $x$->y is the sequence of y children - an array even if only one
- $x$->y[0] is the first y child
- foreach ($x$->y as $y) {..} iterates over the y children
- $x$->children() is the sequence of all $x$'s children
- $x$->xpath(...) evaluate an XPath expression
PHP : access to attributes

If $x$ is an element

- $x->attributes()$ is the sequence of $x$ attributes
- $x->attributes()->z$ is the $z$ attribute of $x$
- foreach ($x->attributes() as $att {.. } iterates over attributes of $x$
PHP : access to namespaces

Namespaces in XML comprise

- a prefix used in the tag `geo:lat`,
- a uri which identifies that namespace
  "http://www.w3.org/2003/01/geo/wgs84_pos#"
- a notation for defining the relationship (binding) between the two
  - a pseudo element attribute usually at document level -e.g. `xmlns:geo=http://www.w3.org/2003/01/geo/wgs84_pos#`

Access to elements in a namespace uses the `children()` function with the namespaceuri as the parameter. By default it gives you all children.
XML Processing Example (wines.xml)

We will make use of the file (wines.xml) in many of the following examples:

```xml
<?xml version="1.0" encoding="utf-8"?>
<wines>
  <category name="Red">
    <product>
      <title>Preference Cûtes de Rhûne Rouge</title>
      <desc>This Caves Saint Pierre Préférence Cûtes du Rhûne is an easy drinking and spicy red wine laden with summer fruit flavours.</desc>
      <price>3.99</price>
      <image>crr.jpg</image>
    </product>
  </category>
  <category name="Rose">
    <product>
      <title>Mateus Rose</title>
      <desc>An all time favourite.</desc>
      <price>3.99</price>
      <image>mro.jpg</image>
    </product>
    <product>
      <title>Blossom Hill White Grenache Rose</title>
      <desc>Aromas of red berry, cherry and white pepper. Lush on the palate with lots of berries. Citrus and honeysuckle notes.</desc>
      <price>3.99</price>
      <image>bhro.jpg</image>
    </product>
  </category>
</wines>
```
<?php
    $wines = simplexml_load_file('wines.xml');
    foreach ($wines->category as $category)
    {
        echo '<p>';
        echo '<b>Type: '. $category['name']. '</b><br/>';
        foreach ($category->product as $product)
        {
            echo $product->title. '<br/>';
        }
        echo '</p>';
    }
    ?>

Using SimpleXML extension in PHP 5

run the code
Using XPath to search documents

Sometimes there is a need to search through a document (e.g. when the exact format or order of the nodes is not known in advance) or when only a small set of nodes need to be found in a very large document.

Using the Xpath support built into the SimpleXML extension this is made easy.

The basic format of a XPath query takes the form “a/b/c” where a, b, c are nested xml tags of the form `<a><b><c/></b></a>`.

Some common XPath queries are as follows:

- **a** : matches any tag named **a**
- **a/b** : matches any tag named **b**, directly contained in the tag **a**
- **a/b/..** : matches and returns the tag **a** instead of **b**
- **a//b** : matches **b** when it is any descendent of **a**
- **a[31]** : matches the 31st **a** tag
- **a[last()]** : matches the very last **a** tag
- **a[@att]** : matches any a with an attribute named “att”
- **a[@att="val"]** : matches any tag called a with an attribute named “att” with the value “val”
SimpleXML & XPath example

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

$category = $wines->xpath('//category');

foreach ($category as $c) {
    echo "<p>
        <b>Type: \$c\['name']</b><br/>
    </p>
    
    $title = $c->xpath('./product/title');
    foreach ($title as $t) {
        echo "$t<br/>
    }
}
?>
```

run the code