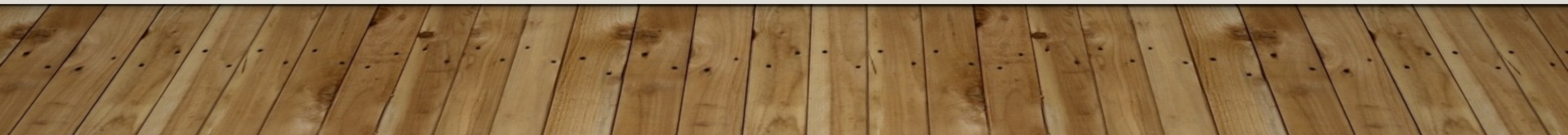
ISIT307 - WEB SERVER PROGRAMMING

LECTURE 7.1 – PHP - XML, PHP - AJAX



LECTURE PLAN

- Working with XML, PHP XML Parsers
- AJAX, Using AJAX with PHP

Look for additional resources:
References in subject outline,
https://www.w3schools.com/php/php_xml_parsers.asp,
https://www.w3schools.com/php/php_ajax_intro.asp

XML

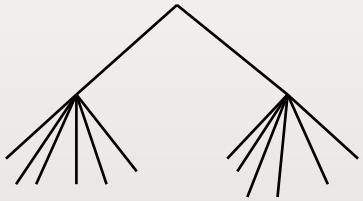
- Extensible Markup Language (XML) is a syntax standard that allows for the exchange of textual information between applications
- Why XML is instrumental to Web Programming?
  - several web technologies (like RSS Feeds and Podcasts) are written in XML
  - XML is easy to create - it looks a lot like HTML
  - a lot of Web Sites share data through XML

XML

- There are two big differences between XML and HTML
  - XML doesn't define a specific set of tags that we must use,
  - XML document must have strict structure
- Tags are words surrounded by "<" and ">" and come in two forms:
  - with values <tagname> value </tagname>
  - without values <tagname />

XML

- Tags can have attributes.
  - <tagname attribute="attributevalue">
- A value of a tag can have other tags within it - these nested tags are called the children tags while the tag itself is the parent tag of the children.
- Example



```
<?xml version='1.0' ?>
<contact idx='37' >
<name>Tom White</name>
<category>Family</category>
<phone type='home'>301-555-1212</phone>
<meta id='x634724' />
</contact>
```

PHP XML PARSER

- To read and update, create and manipulate an XML document, we need an XML parser
- In PHP there are two major types of XML parsers:
  - Tree-Based Parsers (SimpleXML, DOM)
  - Event-Based Parsers (XMLReader, XML Expat Parser)

PHP XML PARSER

- SimpleXML is a PHP extension
- It allows manipulation with XML data
  - transforms an XML document into a data structure (making easy to iterate through like a collection of arrays and objects)
  - if the XML document's structure or layout are known provides an easy way of getting an element's name, attributes and textual content
- Fewer lines of code needed to read text data from an element

PHP XML PARSER

- SimpleXML has:
  - simplexml_load_string() function - operates on an XML document saved into a PHP variable
  - simplexml_load_file() function - operates on a file
- These functions returns an object that replicates the XML document
  - Each XML tag becomes a property of the object, nested accordingly

SIMPLEXML - EXAMPLE (1)

```
<?php
$myXMLData =
"<?xml version='1.0' ?>
<contact idx='37'>
<name>Tom White</name>
<category>Family</category>
<phone type='home'>301-555-1212</phone>
<meta id='x634724' />
</contact>";

if (!($xml=simplexml_load_string($myXMLData)))
      die("Error: Cannot create object");
echo "<pre>\n";
print_r($xml);
echo "</pre>\n";
?>
```

```
SimpleXMLElement Object
(
    [@attributes] => Array
        (
            [idx] => 37
        )

    [name] => Tom White
    [category] => Family
    [phone] => 301-555-1212
    [meta] => SimpleXMLElement Object
        (
            [@attributes] => Array
                (
                    [id] => x634724
                )

        )

)
```

SIMPLEXML - EXAMPLE (2)

```
<?php
$myXMLData =
"<?xml version='1.0' ?>
<contact idx='37'>
<name>Tom White</mname>
<category>Family</category1>
<phone type='home'>301-555-1212</phone>
<meta id='x634724' />
</contact>";

libxml_use_internal_errors(true);

$xml=@simplexml_load_string($myXMLData);
if ($xml===false){
       echo "Failed loading XML: ";
       foreach(libxml_get_errors() as $error)
         echo "<br>". $error->message;
}else{
       echo "<pre>\n";
       print_r($xml);
       echo "</pre>\n";
} ?>
```

SIMPLEXML - EXAMPLE (3)

```
<?php
$xml = simplexml_load_file('contacts.xml');
echo "<ol>\n";
foreach ($xml->contact as $c) {
        // print contact's name, id, email
        // The attribute will be accesible as if it were an assoc array entry. Using the
        //entry itself, will echo it's value.
      echo '<li>' . $c->name . " - " . $c['idx'] . ", email:" . $c->email;

      // start an unordered list for contact's phone numbers
      echo '<ul>';

      // loop over every phone number for this person
      foreach ($c->phone as $p) {
            // echo out a line including the type of number.
            // The attribute will be accesible as if it were an assoc array entry.
            // Using the entry itself, will echo it's value.
            echo '<li>', ucfirst($p['type']), ': ', $p, '</li>';
      }
      echo "</ul></li>\n"; // close the phone list
}?>
```


White III, E. and Eisenhamer, J.D., 2007. PHP 5 in Practice. Pearson Education.

THE XML DOM PARSER

- The DOM sees the XML as a tree structure:
  - Level 1: XML Document
  - Level 2: Root element (e.g.<to>)
  - Level 3: Text element (e.g. "Jack")
- Example

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("list.xml");
echo $xmlDoc->saveXML();
echo "<br /><br />";
$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item) {
    echo $item->nodeName . " = "
    . $item->nodeValue . "<br />";
} ?>
```

```
<?xml version='1.0' ?>
<note>
    <to>Jack</to>
    <from>Anna</from>
    <heading>List</heading>
    <body>All important items</body>
</note>
```

Jack Anna List All important items

#text =
to = Jack
#text =
from = Anna
#text =
heading = List
#text =
body = All important items
#text =
?>

THE XML DOM PARSER

- Example

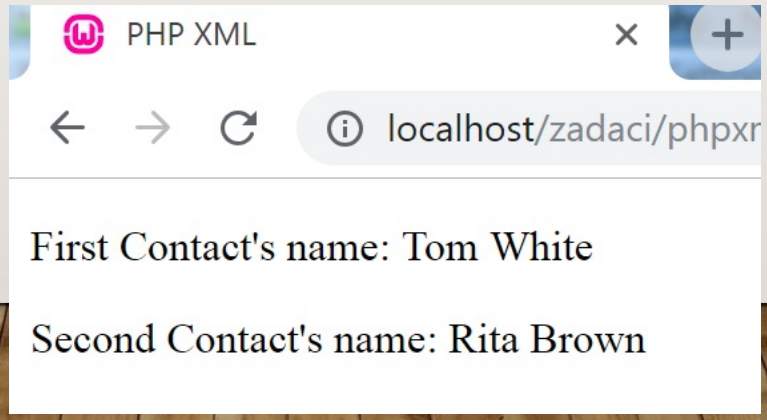
```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("contacts.xml");
$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item) {
  print $item->nodeName . " = " . $item->nodeValue .
"<br>";
}
?>
```

```
#text =
contact = Tom White Family 301-555-1212
#text =
contact = Rita Brown Friends 240-555-1212 410-555-7676 ritab@example.com
#text =
```

THE XML DOM PARSER - EXAMPLE

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("contacts.xml");

$xml = simplexml_import_dom($xmlDoc);
echo "<p>First Contact's name: {$xml->contact[0]->name}</p>\n";
//echo "<p>First Contact's id: {$xml->contact[0]['idx']}</p>\n";
echo "<p>Second Contact's name: {$xml->contact[1]->name}</p>\n";
?>
```



PERFORMING SEARCHES WITH XPATH

- Search through the document for the exact item (or items) is available with `XPath` (a standard for performing searches through XML)
- Some common XPath queries are
  - x - Matches any tag named x
  - x/y/ - Matches any tag named y, directly contained in a tag named x
  - x/y/.. - Similar to the preceding item but actually matches and returns tag x, instead of tag y
  - x//y - Matches any tag named y that is a descendant of a tag named x (any number of levels deep)
  - x[5] - Matches the fifth tag named x
  - x[last()] - Matches the last tag named x
  - x[@att] - Matches any tag named x, with an attribute named att
  - x[@att="val"] - Matches any tag named x, with an attribute named att that has the value "val"

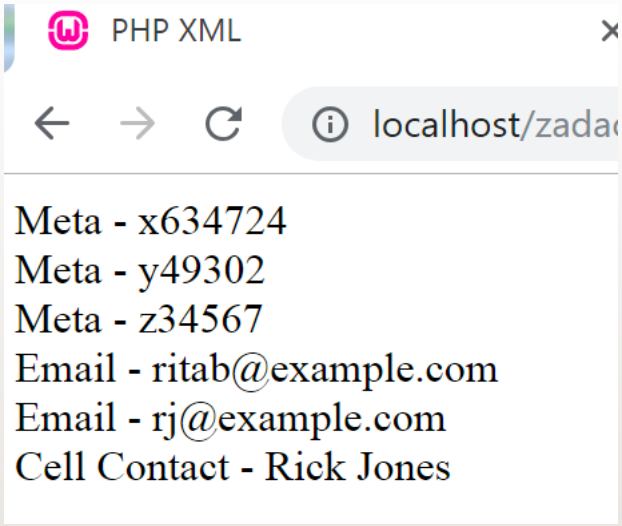
PERFORMING SEARCHES WITH XPATH - EXAMPLE

```
<?php
$xml = simplexml_load_file('contacts.xml'); // using SimpleXML, read
                                            //the file into memory

$meta = $xml->xpath('//meta'); // Xpath search to find all 'meta'
                               //tags no matter what the depth
foreach ($meta as $m) {
      echo "Meta - {$m['id']}<br />\n";
}
$email = $xml->xpath('/contacts/contact/email'); // find all email
      //tags within a contact tag from the root of the XML document
foreach ($email as $e) {
      echo "Email - {$e}<br />\n";
}
$cell = $xml->xpath('contact/phone[@type="cell"]/..'); // find any
                                  //contact who has a cellphone number
foreach ($cell as $c) {
      echo "Cell Contact - {$c->name}<br />\n";
}
?>
```


White III, E. and Eisenhamer, J.D., 2007. PHP 5 in Practice. Pearson Education.

PERFORMING SEARCHES WITH XPATH – EXAMPLE OUTPUT



EXAMPLE

- Write piece of code to convert the XML string/file into an array

```
<?php
$myXMLstr =
"<aaaa Version='1.0'>
   <bbb>
      <cccc>
         <dddd id='id-pass' />
          <eeee name='hearman' age='24'  />
      </cccc>
    </bbb>
</aaaa>";

if (($xml=simplexml_load_string($myXMLstr))){
       $json = json_encode($xml);
       $array = json_decode($json, true);
       echo "<pre>\n";
       print_r($array);
       echo "</pre>\n";
}
?>
```

AJAX

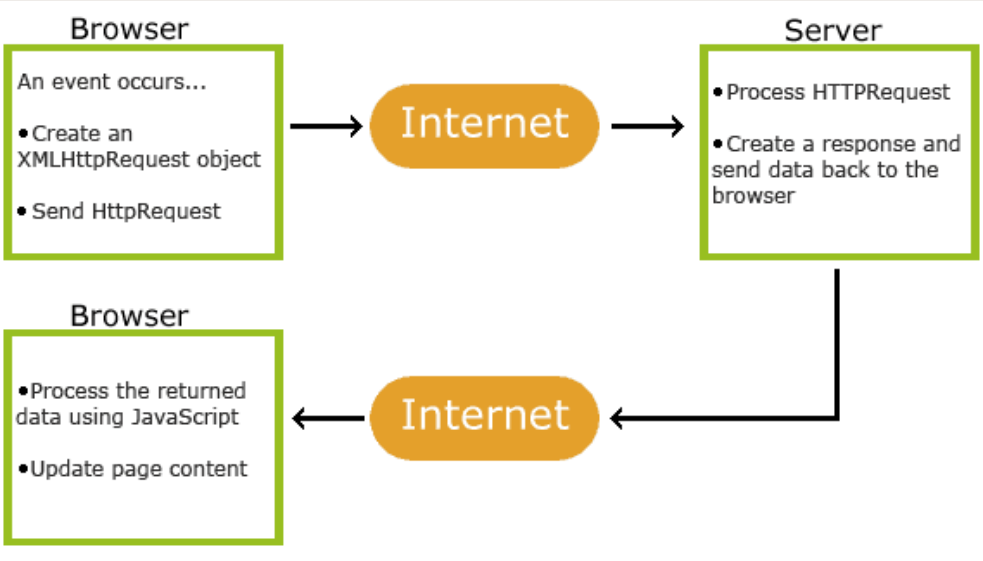
- Ajax - Asynchronous JavaScript and XML, allows a web page, via JavaScript, to make a request back to the server, receive data, and process it without the page ever having to reload
- It allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes, updating parts of a web page, without reloading the whole page

AJAX

- Ajax uses a combination of:
  - XMLHttpRequest object (to exchange data asynchronously with a server)
  - JavaScript/DOM (to display/interact with the information)
  - CSS (to style the data)
  - XML (often used as the format for transferring data)

AJAX

- Ajax in its purest form uses XML as the basis of communication
  - The requested file on the server will respond with an XML document or anything that JavaScript can parse
  - It can also return text or raw HTML as their Ajax responses if it is more convenient



https://www.w3schools.com/php/php_ajax_intro.asp

AJAX AND PHP

- Ajax can call a PHP script, pass it data, and display the results back to the user
- For example, when a user types a character in the input field, a function ("showHint()") is executed - the function is triggered by the onkeyup event

AJAX AND PHP

txtFieldHint.html

```
<html>
<head>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtHint").innerHTML = this.responseText;
            }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

AJAX AND PHP

- The function ("showHint()”):
  - Create an XMLHttpRequest object
  - Create the function to be executed when the server response is ready
    - readyState (Holds the status of the XMLHttpRequest):
      - 0: request not initialized, 1: server connection established, 2: request received, 3: processing request, 4: request finished and response is ready
    - Status:
      - 200: "OK”, 403: "Forbidden”, 404: "Page not found”, …
  - Send the request off to a PHP file (gethint.php) on the server with q parameter (gethint.php?q="+str)
  - The str variable holds the content of the input field

AJAX AND PHP

gethint.php

```php
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";

// get the q parameter from URL
$q = $_GET["q"];
$hint = "";
// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}
// output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

AJAX AND PHP

- AJAX can be used for interactive communication with
  - database
  - text file
  - XML file
- AJAX can be used to create more user-friendly and interactive searches
- Examples: https://www.w3schools.com/php/php_ajax_intro.asp