Systems Programming & Scripting

Lecture 21: XML and File Access in PHP

Accessing a File

```
$fd = fopen(file, mode)
```

- Opens a file. Two arguments:
  - *file*: specifies the name of the file (with path) or an URL to open
  - *mode*: specifies the mode in which to open the file. E.g. “r” – read, “w” – write (can open in read/write mode as well).
- Return value:
  - A file descriptor/handle, which can be used for interaction with the file.

Accessing a File

`fwrite(handle, string);`

- Writes string to the file with descriptor handle.

`fprintf(handle, format, arg1...);`

- Writes the format string, with meta-chars replaced by arg1 .. to handle

`fread(handle, len);`

- Reads len bytes from handle and returns them.

`fgetcsv(handle, len);`

- Reads a comma-separated line from handle, and returns an array of elements.

`feof(handle);`

- Tests if file pointer is pointing to the end of file.

`fclose(handle);`

- Closes the file with file descriptor handle.

Writing to a File

```
<?php
$fileHandle = fopen('test.txt', 'w+') or
  die ("File can't be opened\n");

$result = fwrite($fileHandle, "This is to
  test how to write to a file using PHP");

fclose($fileHandle);
?>
```

PHP XML Parser

- PHP provides 3 different XML parsers
  - An event-driven parser (Expat) based on C libraries
  - A DOM-based parser
  - A SimpleXML parser
- Expat is an event-based parser.
  - The XML document is considered as a series of events.
  - The occurrence of an event triggers a call to an event-handler function.
- Expat is a non-validating parser.
  - Does not check if the XML document is well-formed (recall XML parsers – Lecture 7)

Example

<personName>Peter</personName>

- An event parser will consider the above XML as three events
  - Start element: *personName*
  - CData section, value: *Peter*
  - Close element: *personName*

Parsing

```
$parser = xml_parser_create();
```

- Start an XML parser
- Create functions to handle different events (see below)
- Associate functions with events (see below)

```
$success = xml_parse($parser, $data);
```

- Run the parser on the XML input *data.*
- Close the parser.

Event Handlers

```
xml_set_element_handler(parser,
  start_element, end_element);
```

- Defines functions start_element and end_element that are executed at the start/end of an XML element.
- The name of the element is passed as an argument to the handler functions.
- *parser* is a reference to the XML parser.

Event Handlers

```
my_start_element_handler(parser,
  element, attributes);
```

- Defines a concrete handler function
- *parser* is a reference to the parser
- *element* is the name of the element
- *attributes* is an array of XML attributes associated with this element

Event Handlers

```
my_end_element_handler(parser, element);
```

- Handler for the end tag of element

- These functions can be used to extract information from the XML file and transform or display it.

Event Handler

`xml_set_character_data_handler(parser, handler)`

- Defines a *handler* for character data, which is called after each block of character data.

Event Handler

`xml_set_processing_instruction(parser, handler)`

- Defines a handler, which is called when finding a processing instruction in the XML text, of the form

<?target instructions ?>

Event Handler

```
my_processing_instruction_handler(parser
  , target, instructions);
```

- Defines an instruction handler, with contents target and instructions
- PHP's `eval` function can be used to evaluate PHP code that is embedded in an XML file, eg.

```
if ($target == 'php') { eval($instructions); }
```

Entity Handlers

- The Expat parser supports handlers for two kinds of entities:
  - External entities
  - Unparsed entites
- External entities specify the name of a file or URL, that should be included.
- The corresponding handler has to do the inclusion, possibly using its own parser
- Unparsed entities represent blocks of raw data, eg. jpg images etc
- Unparsed entities have to be accompanied by notation declarations
- A notation handler specifies how to handle the raw data

Default Handler and Options

`my_set_default(parser, handler)`

- Defines a handler for any other event.

`my_default_handler(parser, text)`

- The handler takes the parser and the text as arguments.

`xml_parser_set_options(parser,option,value)`

- Sets a parser option that controls the behavious of the parser, eg. Character encoding

Methods as Handlers

- Typically, the code for a concrete parser needs several properties to track progress and store results.
- Additionally, several handlers must be defined.
- To encapsulate this information, an object-oriented design is used.

```
xml_set_object(object);
```

- Registers an object with a parser.
- The XML parser looks into the object find handlers and properties.

Example: A library

```
<?xml version="1.0" ?>
<library>
  <book>
    <title>Programming PHP</title>
    <authors>
      <author>Rasmus Lerdorf</author>
      <author>Kevin Tatroe</author>
    </authors>
    <isbn>1-56592-610-2</isbn>
    <comment>A great book!</comment>
  </book>
  <book>
    <title>PHP Pocket Reference</title>
    <authors>
      <author>Rasmus Lerdorf</author>
    </authors>
    <isbn>1-56592-769-9</isbn>
    <comment>It really does fit in your pocket</comment>
  </book>
</library>
```

Example: Bookparse

```
class BookList {

  var $parser;
  var $records;
  var $record;
  var $current_field = '';
  var $field_type;
  var $ends_record;
```

Example from "Programming PHP", R. Lerdorf et al, O'Reilly, 2006

Example: Bookparse

```
 function BookList ($filename) {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser, &$this);
    xml_set_element_handler($this->parser, 'start_element',
'end_element');
    xml_set_character_data_handler($this->parser, 'cdata');

    // 1 = single field, 2 = array field, 3 = record
container
    $this->field_type = array('title' => 1,
                              'author' => 2,
                              'isbn' => 1,
                              'comment' => 1);
    $this->ends_record = array('book' => true);

    $x = join("", file($filename));
    xml_parse($this->parser, $x);
    xml_parser_free($this->parser);
 }
```

Example: Bookparse

```
  function cdata ($p, $text) {
     if ($this->field_type[$this->current_field] === 2) {
       $this->record[$this->current_field][] = $text;
     } elseif ($this->field_type[$this->current_field] === 1)
{
       $this->record[$this->current_field] .= $text;
     }
   }
```

Example: Bookparse

```
function start_element ($p, $element, &$attributes) {
   $element = strtolower($element);
   if ($this->field_type[$element] != 0) {
     $this->current_field = $element;
   } else {
     $this->current_field = '';
   }
 }

 function end_element ($p, $element) {
   $element = strtolower($element);
   if ($this->ends_record[$element]) {
     $this->records[] = $this->record;
     $this->record = array();
   }
   $this->current_field = '';
 }
```

Example: Bookparse

```
function show_menu() {
   echo "<table border=1>\n";
   foreach ($this->records as $book) {
     echo "<tr>";
     $authors = join(', ', $book['author']);
     printf("<th align=left><a href='%s'>
                %s</a></th><td>%s</td></tr>\n",
         $_SERVER['PHP_SELF'] . '?isbn='$book['isbn'],
         $book['title'],
         $authors);
     echo "</tr>\n";
   }
   echo "</table>\n";
 }
```

Example: Bookparse

```
function show_book ($isbn) {
  foreach ($this->records as $book) {
    if ($book['isbn'] !== $isbn) {
      continue;
    }

    $authors = join(', ', $book['author']);
    printf("<b>%s</b> by %s.<br>",
              $book['title'], $authors);
    printf("ISBN: %s<br>", $book['isbn']);
    printf("Comment: %s<p>\n", $book['comment']);
  }
```

Example: Bookparse

```
// main program code

$my_library = new BookList ("books.xml");
if ($_GET['isbn']) {
  // return info on one book
  $my_library->show_book($_GET['isbn']);
} else {
  // show menu of books
  $my_library->show_menu();
}
```

SimpleXML

- For simple, small XML data, the SimpleXML parser can be used.
- It reads the entire XML file and generates a hierarchy of classes, reflecting the structure of the XML document.
- The elements of the XML file can be accessed as properties in these classes

SimpleXML Methods

- The following methods can be used to traverse the class hierarchy
  - `$node->children()` … iterate over the children of $node
  - `$node->attributes()` … iterate over attributes of a node
  - `$node->asXML()` … retrieves the original XML text for the node

SimpleXML Example

```
$document =simplexml_load_file('books.xml');
foreach ($document->book as $book) {
  echo $book->title;
}
```

Transforming XML with XSLT

- Extensible Stylesheet Language Transformations (XSLT) is a language to transfrom XML files into XML or HTML etc
- PHP provides an interface to a C library of XSLT functions.
- The documents involved in the transformation are:
  - The original XML document
  - The XSLT document with the transformation rules
  - The resulting doument

Basic XSLT Interface

```
$proc = new XSLTProcessor;
```

- Generates a transformation object

```
$xsl = new DOMDocument;
$xsl->load($file);
```

- Parses an XML or XSL document into a DOM object

```
$proc->importStyleSheet($xsl);
```

- Attach the XML rules to the $proc object

```
$proc->transformToXML($xml);
```

- Performs the transformation

Example: RSS Viewer

```
$proc = new XSLTProcessor;

$xsl = new DOMDocument;
$xsl->load('rss2html.xsl');
$proc->importStyleSheet($xsl);

$xml = new DOMDocument;
$xml->load('news.rss');
$res = $proc->transformToXML($xml);

echo "<pre>Result: $result</pre>";
```

Web Services

- It is possible to call functions on a server by embedding information on remote procedure calls in XML text.
- This is a generic mechanism for providing remote services as web services
- Two mechanisms exist:
  - XML-RPC
  - SOAP

XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<course>
<courseName>XML Course</courseName>
<lecturer>John Smith </lecturer>

<university>Heriot Watt University </university>

<description>
    The main concepts and technologies of XML and PHP
    are taught in this course.
</description>
</course>
```

XML Parsing Program*

```
<?php
//xml parser initialisation
$parser = xml_parser_create();
//function for use when dealing with an element start.
function start($parser, $element_name, $element_attrs){
    if($element_name == “course”){echo “Course <br /> <br />”;}
    else if($element_name == “courseName”){echo “Course
    Name:”;}
    else if($element_name == “lecturer”){echo “Lecturer:”;}
    else if($element_name == “university”){echo “University:”;}
    else if($element_name == “description”){echo “Course
    Description:”;}
}
```

* Based on http://www.w3schools.com/php/php_xml_parser_expat.asp

Cont. XML Parsing Program

```
// function for use when dealing with an element end.
function stop($parser, $element_name){
    echo “<br/>”;
}
// function for use when dealing with a character.
function char($parser,$data){
    echo $data
}
// specifying functions that handle the start and end of elements
xml_set_element_handler($parser, “start”,”stop”);

//specify function that handles characters.
Xml_set_character_data_handler($parser, “char”);
```

Cont. XML Parsing Program

```
$xmlfile = fopen(“coursedetails.xml”, “r”);

while ($content =fread($xmlfile, filesize($xmlfile))){
    xml_parse($parser, $content, feof($xmlfile)) or
    die (sprintf("XML Error: %s at line %d",
    xml_error_string(xml_get_error_code($parser)),
    xml_get_current_line_number($parser)));
}
// free the parser
Xml_parser_free($parser);
//close the file
fclose($xmlFile);
```

Output

Course

Course Name: XML Course
Lecturer: John Smith
University: Heriot Watt University
Course Description: The main concepts and technologies of XML and PHP are taught in this course.

Summary

- PHP provides a rich set of libraries for interacting with files.
- The plain interface of fopen/fwrite/fclose is similar to C#
- Several XML parsers can be used to read/write XML data
- XSLT transformations can be used to transform XML data