Web Programming

YJ – Aug 2015

Course Outline

There're 16 lectures (2.5 hrs each) in this course. We'll mainly focus on web programming using LAMP model.

During the 24 hours tutorials in each week, our tutors will help you go through all the contents in the lecture notes and help you with the assignments

All scheduled lectures are learner-paced which means they could be altered depends on the situation. We help you gain rather than cram!

Course Outline

- L1 ~ Web programming intro
- L2 ~ HTML5
- L3 ~ CSS3
- L4 ~ JavaScripts
- L5 ~ PHP 1 Basic
- L6 ~ PHP 2 OO && Functions
- L7 ~ PHP 3 OO && Sessions
- L8 ~ PHP 4 Advanced features
- L9 ~ Mysql
- L10 ~ Mysql 2
- L11 ~ XML/JSON
- L12 ~ Ajax
- L13 ~ Framework: Bootstrap && Laravel
- L14 ~ CMS: Wordpress, Joomla, etc.
- L15 ~ Final Projects
- L16 ~ Final Projects

XML

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to describe data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

PHP SimpleXML Parser

- SimpleXML is a tree-based parser.
- SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.
- SimpleXML turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

simplexml_load_file()	Converts an XML file into a SimpleXMLElement object
simplexml_load_string()	Converts an XML string into a SimpleXMLElement object
getName()	Returns the name of the XML tag referenced by the SimpleXML element
getNamespaces()	Returns the namespaces USED in document
addAttribute()	Adds an attribute to the SimpleXML element
addChild()	Adds a child element the SimpleXML element
count()	Counts the children of a specified node
asXML()	Returns a well-formed XML string from a SimpleXML object

PHP SimpleXML Parser

- Simplexml_load_string

```
$myXMLData =
    "<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";

$xml=simplexml_load_string($myXMLData) or die("Error: Cannot create object");
print_r($xml);
```

- Simplexml_load_file

```
$xml=simplexml_load_file("note.xml") or die("Error: Cannot create object");
print_r($xml);
```

PHP SimpleXML Parser

- getName()

```
$xml=<<<XML
<?xml version="1.0" standalone="yes"?>
<cars>
  <car id="1">Volvo</car>
  <car id="2">BMW</car>
  <car id="3">Saab</car>
</cars>
XML;

$sxe=new SimpleXMLElement($xml);
echo $sxe->getName() . "<br>";
foreach ($sxe->children() as $child)
{
    echo $child->getName() . "<br>";
}

$ns=$sxe->getNamespaces(true);
var_dump($ns);
```

PHP SimpleXML Parser

- addChild() vs addAttribute()

```
$note=<<<XML
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
XML;

$xml = new SimpleXMLElement($note);
$xml->addAttribute("type","private");
$xml->body->addAttribute("date","2014-01-01");

echo $xml->asXML();

// Add a child element to the body element
$xml->body->addChild("date","2014-01-01");
// Add a child element after the last element inside note
$footer = $xml->addChild("footer","Some  footer text");

echo $xml->asXML();
```

PHP SimpleXML Parser

- count()

The count() function counts the children of a specified node

```
$xml=<<<XML
  <cars>
    <car name="Volvo">
    <child/>
    <child/>
    <child/>
    <child/>
  </car>
  <car name="BMW">
    <child/>
    <child/>
  </car>
</cars>
XML;

$elem=new SimpleXMLElement($xml);
foreach ($elem as $car)
{
    printf("%s has %d children.<br>", $car['name'], $car->count());
};
```

JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format.
- It is easy for humans to read and write. It is easy for machines to parse and generate.
- JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, properties make JSON an ideal data-interchange language.

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

JSON Functions

json_encode	Returns the JSON representation of a value
json_decode	Decodes a JSON string
json_last_error	Returns the last error occurred

```php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
echo json_encode($arr);
//output
{"a":1,"b":2,"c":3,"d":4,"e":5}

$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));
//output
object(stdClass)#1 (5) {
["a"] => int(1)
["b"] => int(2)
["c"] => int(3)
["d"] => int(4)
["e"] => int(5)
}
```

```
array(5) {
  ["a"] => int(1)
  ["b"] => int(2)
  ["c"] => int(3)
  ["d"] => int(4)
  ["e"] => int(5)
}
```