Server Side Languages

Web Design & Development

Day 5

Lecture Overview

- 5.1 Student Topic 1 Presentations
- 5.2 Web APIs
- 5.3 XML
- 5.4 JSON
- 5.5 AJAX – creating server responses
- Lab MVC App & API

Student Presentations

Topic 1

APIs

- API is a set of routines, protocols, and tools for building software applications.
- A good API makes it easier to develop a program by providing all the building blocks.
- A programmer then puts the blocks together and you can use all of the blocks or some of the blocks.

XML

- Extensible Markup Language. Standardized by the W3C. http://www.w3.org/XML/
- Header tag indicates the document is in XML format
- Opening and closing tags, similar to HTML
- Nesting tags creates hierarchy
- Data schema is included by virtue of tags - Database=Catalog, Table=CD, Columns=Title,Artist,Country,Company,Price,Year
- Lots of redundancy because the tags are used as metadata.

Sample XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>CBS Records</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1988</YEAR>
    </CD>
</CATALOG>
```

Create XML Data in PHP

```
header("Content-Type: text/xml"); //make sure the correct MIME type is set
$doc=new DOMDocument('1.0');
$doc->formatOutput=true; // we want a nice output
$root=$doc->createElement('employees');
$root=$doc->appendChild($root);
  $person=$doc->createElement('person');
  $person=$root->appendChild($person);
    $firstname=$doc->createElement('firstName');
    $firstname=$person->appendChild($firstname);
      $text=$doc->createTextNode('John');
      $text=$firstname->appendChild($text);
    $lastname=$doc->createElement('lastName');
    $lastname=$person->appendChild($lastname);
      $text=$doc->createTextNode('Doe');
      $text=$lastname->appendChild($text);
echo $doc->saveXML()."\n"; //output the XML to the browser

<?xml version="1.0"?>
<employees>
  <person>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
  </person>
</employees>
```

Read XML in PHP

```
<?xml version="1.0"?>
<employees>
  <person>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
  </person>
  <person>
    <firstName>Anna</firstName>
    <lastName>Smith</lastName>
  </person>
  <person>
    <firstName>Peter</firstName>
    <lastName>Jones</lastName>
  </person>
</employees>

<?php //NOTE that XML tags are Case Sensitive
$xmlStr = file_get_contents("http://localhost:8888/Day4/xmlencode.php");
$xml = new SimpleXMLElement($xmlStr);
foreach ($xml->person as $person) {
    echo $person->firstName . ' ' . $person->lastName . '<br />';
 }
```

Exercise 5.1: Create and Read XML in PHP

1. Using your "fruits" database from Day 3, instead of echoing the output to the screen, generate a PHP script that outputs XML. Your output would be like:

```
<?xml version="1.0"?>
<fruits>
  <id>1</id>
  <fruitname>Apple</fruitname>
  <fruitcolor>Red</fruitcolor>
</fruits>
```

2. Using the script you created in Part 1, create a second PHP script that accepts the XML input and echoes out the fruitnames and fruitcolors.

   If Step 1 was fruitxml.php, then your input file would be "http://localhost:8888/fruitxml.php" (relative to your MAMP path)

JSON

JavaScript Object Notation. http://www.json.org/

Contains a collection (array) of name/value pairs in an ordered list.

Delimited with colons, commas, curly braces, and square brackets.

Much more simplified than XML. Smaller and faster than XML, easier to parse.

http://json.org/example

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

Create JSON Object

```
<?php
$json = array(
    'employees'=>array(
        array("firstName"=>"John", "lastName"=>"Doe"),
        array("firstName"=>"Anna", "lastName"=>"Smith"),
        array("firstName"=>"Peter", "lastName"=>"Jones")
        )
);
header("Content-Type: application/json");
echo json_encode($json);

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

Decode JSON Objects

Decoding JSON objects in PHP is very straightforward. The jsondecode() function will take a string in JSON format and convert to a variable containing a multidimensional, associative array.

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

```
<?php
$JSONstr = file_get_contents("http://localhost:8888/Day4/jsonencode.php");
$JSONarr = json_decode($JSONstr, true);
foreach($JSONarr['employees'] as $employee) {
  echo $employee['firstName'] . ' ' . $employee['lastName'] . '<br />';
}
```

Query Database

Let’s test our query

select * from cities where city_ascii='orlando';
(10 rows, 2.01s)

select * from cities where city_ascii='orlando' AND country='us';
(10 rows, 218ms)

select * from cities where city_ascii LIKE "winter%" AND country="us" order by city_ascii limit 20;
(20 rows, 206ms)

Exercise 5.2: Server-side AJAX Response

What if we want to build interactive input, say autocomplete like Google

- HTML Page - generates the JavaScript triggered by events
- JavaScript - handle events, send AJAX request to server and wait for response
- PHP - handle AJAX request sent by JavaScript and return some response back.
- JavaScript - handle response back event, do something with response.

Copy the following two scripts, city.html and cityresponse.php

Open up the tools on your browser so you can see the requests send/return and run.

city.html

```
<html>
<head>
    <script type="text/javascript">
        function checkcity(cityentered) {
            var xmlhttp;

            if (window.XMLHttpRequest) {
                xmlhttp = new XMLHttpRequest();
            }

            xmlhttp.onreadystatechange=function()
            {
                if (xmlhttp.readyState==4 && xmlhttp.status==200)
                {
                document.getElementById("cityresponse").innerHTML=xmlhttp.responseText;
                }
            }
            xmlhttp.open("GET","cityresponse.php?cityget="+cityentered,true);
            xmlhttp.send();
        }
    </script>
</head>
<body>
    <input type="text" name="city" id="city" placeholder="Enter City Name" onkeyup="
    checkcity(this.value)"/>
    <div id="cityresponse"></div>
</body>
</html>
```

cityresponse.php

```php
<?php
//cityresponse.php

$cityentered=trim($_GET['cityget']).'%';
$user='root';
$pass='root';
$dbh = new PDO('mysql:host=localhost;dbname=worldcities;port=3306;',
    $user, $pass);

$stmt = $dbh->prepare('SELECT city, region
    FROM cities
    WHERE city_ascii LIKE :cityentered
    AND country = "us"
    ORDER BY city_ascii LIMIT 20;
    ');
$stmt->bindParam(':cityentered', $cityentered);
$stmt->execute();
$result = $stmt->fetchAll();
print_r($result);
//echo json_encode($result);
```

Questions?

Lab 5: APIs

In this lab you will utilize APIs and interact with browser jQuery requests. You will use MVC methods and combine all your work into one app.

by now your app should have:
Design
Login area
Protected page with CRUD functionality

Lab Assignment
At the end of this lab your app should have MVC structure, API, and ajax functionality.

5**.1** Using the jQuery Autocomplete plugin, create a form that allows a user to input a city, and suggest city/state based on queries from your Worldcities database.
http://jqueryui.com/autocomplete/