CSCB20 – Week 9

Introduction to Database and Web Application Programming

*Anna Bretscher**

Winter 2017

*thanks to Alan Rosselet for providing the slides these are adapted from.

Week 9

Web services – defining return types
Web services – defining return errors

PHP and MySQL

Connecting to a database

Making queries

Displaying the results

Web Services

- In Assignment 2 – part 2 you use PHP embedded within an HTML document to implement dynamic HTML content
- However, HTML is only one of several kinds of data a server could produce for use by a client

- A Web service refers to use of the Web's HTTP protocol to invoke programs and retrieve their results
- The general idea is that we should be able to call programs using URL references, just as we do to refer to Web pages
- Like traditional functions, Web-service programs can take parameters and produce results, which may be written as HTML, but also as XML, JSON, plain text, or other formats

Web Services and PHP

- The *type* of output produced by a Web service must be explicitly specified, since it can take different ones
- The client needs to know how to interpret the byte values returned by the server
- HTTP, the Internet protocol used for Web URL requests and responses, provides a "Content-type" header for this purpose
- In PHP, the "type" of the result value(s) defaults to HTML ("text/html"), but can be explicitly specified using:

```
header("Content-type: type/subtype");
```

- The `header()` function must be called *before* a PHP script generates any output (since the client who called the script needs the header information to interpret that output)
-
-

MIME Content-Types

- MIME types are used to communicate the type of data sent by a server to a client (e.g. a jpeg image, or an html file), and vice versa (e.g. a file upload from a client)

- MIME types are specified in two parts: “type/subtype”, e.g.:

MIME type	related file extension
text/plain	.txt
text/html	.html, .htm, ...
text/css	.css
application/json	
image/png	.png
image/jpg	.jpeg, .jpg, .jpe
text/javascript	.js

•

•

A PHP Web service

- Let's examine a simple example of a PHP Web service that take "base" and "exp" parameters, and returns the base raised to the exp (exponent) power.
- A URL to invoke this service might look like this:

https://mathlab.../cscb20w17/utorid/power.php?base=5&exp=3

- How would we implement this service in PHP?

```
<?php
    header("Content-type: text/plain");
    $base = (int) $_REQUEST["base"];
    $exp  = (int) $_REQUEST["exp"];
    $result = pow($base, $exp);
    print $result;
?>
```

•

•

Web Service Errors

- When implementing a Web service, we must make provision for *errors*, such as omission of a required parameter or an invalid parameter value. E.g.

```
https://mathlab…/utorid/power.php?base=5&exp=w
https://mathlab…/utorid/power.php?base=5
```

- How should such an error be reported?

- We could return an HTML error message, but what if the client (caller) takes the result and displays it in a result <div> on their Web page, now they display an error message where the user expects a number

- We need a mechanism that will enable the caller to detect that the result is an error, as opposed to a result value.

HTTP Status Codes

The Web's HTTP protocol provides a mechanism for signaling the outcome of a request, that can be used for both ordinary Web pages (e.g. 404 Not Found), and for Web services (e.g. 400 illegal request)

HTTP code	Meaning
200	OK
301-303	page has moved (temporarily or permanently)
400	illegal request
401	authentication required
403	you are forbidden to access this page
404	page not found
410	gone; missing data or resource
500	internal server error

A Web Service with Error Handling

We could rewrite the `power()` web service to detect missing or invalid parameters as follows:

```php
<?php
   $base = $_REQUEST["base"];
   $exp = $_REQUEST["exp"];
   if (is_numeric($base) and is_numeric($exp)) {
      header("Content-type: text/plain");
      ... as before for valid input ...
   } else {
      header("HTTP/1.1 400 Invalid Request");
      die("invalid request; required parameters");
   }
?>
```

Web Service Output Types

So far, our Web service examples have output values expressed as MIME type text/plain.

More commonly, a Web service invoked from a Web page will return an HTML fragment, XML data, or JSON data.

Why an HTML fragment? Because normally the result returned by a Web service will be inserted into an existing HTML document, e.g. as the content of a <div>

Web Service Output Types

Suppose we want to generate an HTML list of factorial values, up to a user-supplied value of “n”:

```
<?php
   header("Content-type: text/html");
   $limit = (int) $_GET["n"];
   $fact = 1;
   for ($i = 1; $i < $limit; $i++) { ?>
        <li>Factorial of <?= $i ?> is <?= $fact ?> </li>
        <?php $fact = $fact * $i;
   }
?>
```

Later we’ll look at how an HTML fragment, like the one generated by this script could be inserted into a Web page

PHP and MySQL

We can use PHP to connect to a MySQL database using MySQLi:

```
<?php
$servername = "localhost";    // mathlab.utsc.utoronto.ca
$username = "username";      // utorid
$password = "password";      // utorid password
$db = "database_name"    // could be utorid, imdb
// Create connection
$conn = mysqli_connect($servername, $username, $password, $db);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";

// After we are done, close the connection
mysqli_close($conn);
?>
```

•

•

Using the data

Once we have a connection, we can begin using the database.

```
<?php
$sql = "SELECT first_name, last_name FROM actors";
$result = mysqli_query($conn, $sql);

// if the query returned any tuples output each tuple
if (mysqli_num_rows($result) > 0) {
  // as long as there is a next tuple, output
  while($row = mysqli_fetch_assoc($result)) {?>
    Name <?= $row["firstname”].“ “.$row["lastname”]?><br>
<?php } ?>
} else {
    echo "0 results";
}
?>
```

•

•

Staying Secure

Recall we can use PHP to connect to a MySQL database using MySQLi:

```
<?php
$servername = "localhost"; // mathlab.utsc.utoronto.ca
$username = "username”;      // utorid
$password = "password";      // utorid password
$db = "database_name"   // could be utorid, imdb
// Create connection
$conn = mysqli_connect($servername, $username,
                        $password, $db);
```

But in this way we have all our private passwords in our file – accessible. Better solution…

Including .php file

We include a `config.php` file with this data specified:

```
<?php
$servername = "localhost"; //mathlab.utsc.utoronto.ca
$username = "username”;     // utorid
$password = "password";     // utorid password
$dbname = "database_name"   // could be utorid, imdb
```

In our original file we include the config file:

```
include 'config.php';
// Create connection
$conn = mysqli_connect($servername, $username,
                       $password, $dbname);
```

•

•