Introduction to PHP

MUUG May Meeting

May 14, 2013. Winnipeg

Theo Baschak <theo@voinetworks.ca>

http://blog.ciscodude.net/

Who Is Theo?

My name is Theodore Baschak. I currently work for Voi Network Solutions (a commercial Internet Service Provider) as the primary Autonomous System Operator. I am also involved with the organization and operation of the Winnipeg Internet Exchange.

Other Places I've Worked at in the Past

- Manitoba First Nations Education Resource Centre
- Broadband Communications North
- Frantic Films
- Genes Telecom

Obligatory Hello World

Quick Hello World example.

```
<?php
$message = "Hello World!";

echo $message;
echo "\n<br />\n";
?>
```

Output:

```
Hello World!
```

Variables

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive. Our Hello World example used variables.

```
<?php
$message = "Hello World!";
$hey = "Hey,";

$toPrint = $hey . " " . $message;
echo $toPrint . "\n<br />\n";
?>
```

Output:

```
Hey, Hello World!
```

Types

PHP supports four scalar types, boolean, int, float, and string. It also supports two compound types, array and object. There is also two special types, resource, and NULL.

```
<?php
$a_bool = TRUE;   // a boolean
$a_str  = "foo";  // a string
$a_str2 = 'foo';  // a string
$an_int = 12;     // an integer

echo gettype($a_bool); // prints out:  boolean
echo gettype($a_str);  // prints out:  string

// If this is an integer, increment it by four
if (is_int($an_int)) {
    $an_int += 4;
}

// If $a_bool is a string, print it out
// (does not print out anything)
if (is_string($a_bool)) {
    echo "String: $a_bool";
}
?>
```

Output:

```
booleanstring
```

Control Structures

Any PHP script is built out of a series of statements. A statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing (an empty statement). Statements usually end with a semicolon. In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces. A statement-group is a statement by itself as well. There are many syntaxes available, some more readable than others.

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>

<?php
if ($a == 5)
    echo "A is equal to 5";


if ($a == 5) {
    echo "A is equal to 5";
}
?>
<?php
if ($a == 5):
    echo "a equals 5";
    echo "...";
elseif ($a == 6):
    echo "a equals 6";
    echo "!!!";
else:
    echo "a is neither 5 nor 6";
endif;
?>
```

Output:

```
a is neither 5 nor 6
```

Functions

Any valid PHP code may appear inside a function, even other functions and class definitions. Function names follow the same rules as other labels in PHP. A valid function name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

```
<?php
function helloworld()
{
  echo "Hello World.<br>\n";
}

function helloworldDLX($name = "Default")
{
  echo "Hello World $name.<br>\n";
}


helloworld();
helloworldDLX();
helloworldDLX('MUUG Members');
?>
```

Output:

```
Hello World.
Hello World Default.
Hello World MUUG Members.
```

General Usage

Since PHP was originally designed and used for CGI type applications dealing with HTTP GET and POST requests is very easy. The $_GET array contains all variables that were defined on query string, and the $_POST array contains all the posted form variables. These should ALWAYS be sanitized before use, especially in database applications.

```
<?php
include "dbconnect.inc.php";

$first      = 'Theo';
$last       = 'Baschak';
$newline    = "\n<br>";

$name       = $_GET['name'];
$postedname = $_POST['name'];
$cookie     = $_COOKIE['somecookie'];

$cleanname  = mysql_real_escape_string($name);

$fullname   = $first . " " . $last;
$flast      = substr($first, 0, 1) . $last;

echo $fullname . $newline;
echo $flast . $newline;

?>
```

Output:

```
Theo Baschak
TBaschak
```

External Connections

PHP has extensions to connect to many external data sources.

- MySQL
- PostgreSQL
- mSQL
- SQLite
- Many Other DBs
- Image Processing and Generation
- GeoIP
- JSON
- LDAP
- Radius
- Regex

Simple JSON Example

Below is an example of requesting a resource from a HTTP server using HTTP/1.1, and then parsing the JSON response and doing something with the data.

```
<?php
$context = stream_context_create(
  array('http' => array(
    'method'=>"GET", 'header'=>"Content-Type: text/html; charset=utf-8",
    'header' => 'Host: 192.0.2.1')));
$url = 'http://192.0.2.1/live.php';
$contents = file_get_contents($url, 0, $context);
$json = json_decode($contents, TRUE);
if(count($json[1]) > 0) // result returned
{
  foreach($json[1] as $items)
  {
    $state = $items[1];
    switch($state)
    {
      case 0:
        $status = "<font color='green'>UP</font>";
        break;
      case 1:
        $status = "<font color='red'>DOWN</font>";
        break;
      default:
        $status = "<font color='orange'>UNKNOWN</font>";
    }
    printf("<tr><td>s</td><td>s</td></tr>\n", $items[0], $status);
  }
}
?>
```

Bcrypt

When writing your own web applications which store and process passwords it is important to ensure the safety of the passwords. PHP 5.5 includes a new function called password_hash which uses the standard Bcrypt method and should simplify this area a great deal.

Things to avoid when handling passwords

- Plaintext storage
- Unsalted MD5/SHA1/anything
- Designing your own algorithm

```
<?php
include "Bcrypt.php";

// In a registration or password-change form:
$hash_for_user = Bcrypt::hash($_POST['password']);

// In a login form:
$is_correct = Bcrypt::check($_POST['password'], $stored_hash_for_user);

?>
```

Simple Session Based Login

Below is a very simple example of a session based login system.

```
<form method='POST' action='login.php'>
  UserName: <input name='user' type='text' size='20'><br>
  Password: <input name='pass' type='password' size='20'><br>
  <input type='submit' name='submit' value='Login'>
  <input type='reset' name='reset' value='Reset'>
</form>


<?php
    include_once("inc/Bcrypt.php");
    include_once("inc/session.inc.php");
    include_once("inc/mysql.inc.php");
    $escapedName = mysql_real_escape_string($_POST['user']);
    $stored_hash_for_user = mysql_one_data(
        "SELECT bcrypt FROM users WHERE username = '$escapedName';");
    $is_correct = Bcrypt::check($_POST['pass'], $stored_hash_for_user);
    if ($is_correct) {
        $row = mysql_one_array(
            "SELECT * FROM users WHERE username = '$escapedName';");
        $_SESSION['valid'] = 1;
        $_SESSION['user'] = $row['username'];
        $_SESSION['userid'] = $row['id'];
        $_SESSION['email'] = $row['email'];
        $_SESSION['desc'] = $row['description'];
        printf("%s is now logged in.<br><a href='index.php'>Continue On...</a>",
            $row['username']);
    } else {
        //User was not logged in successfully
    }
?>
```

Logging Out

Below is a very simple example of logging out of a session based login system.

```
<?php
    include_once("inc/session.inc.php");
    $_SESSION = array(); //destroy all of the session variables
    session_destroy();
    header("Location: index.php"); // redirect back to index after
?>
```

Changing Password

Changing the password requires a few things to be verified, that the old password matches the old one, that the new password meets the length and complexity requirements, and that the old and new passwords are not the same.

```
<?php
    include_once("inc/session.inc.php");
    if (isset($_SESSION['user'])) {
        // logged in
        // Form Variables
        $old   = $_POST["oldpass"];
        $new1  = $_POST["new1"];
        $new2  = $_POST["new2"];
    // do a few checks on the password, abbreviated for brevity
        if(isset($new1) && $new1 == $new2 && $new1 != "" && $old != $new1)
        {
            include_once("inc/Bcrypt.php");
            include_once("inc/mysql.inc.php");
            $escapedName = mysql_real_escape_string($user);
            $stored_hash_for_user = mysql_one_data(
              "SELECT bcrypt FROM users WHERE username = '$escapedName';");
            $is_correct = Bcrypt::check($old, $stored_hash_for_user);
            if($is_correct)
            {
                // Old password matches Old password...
                $hash_for_user = Bcrypt::hash($new1);
                $query = sprintf(
                  "UPDATE users SET bcrypt = 's' WHERE id = 's'",
                  $hash_for_user, $_SESSION['userid']);
                mysql_query($query);
                // Password successfully changed, should check for errors
            } else {
                print "Old password does not match.";
            }
        } else {
            // if new is equal to old or blank, or new passwords don't match
            header("Location: changepw.php");
        }
    } else {
        // not logged in, redirect to login screen
        header("Location: index.php");
    }
    ?>
```

Summary

Resources to visit afterwards.

- http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/

Index