Faculty of Science,
Technology
and Medicine

Web Programming

Volker Müller
University of Luxembourg

PHP: Hypertext Preprocessor

HTML is static, no programming features

PHP = widely-used Open Source general-purpose scripting language

PHP code embedded in HTML

Syntax very similar to Perl (C, Java)

Documentation at www.php.net

Example: PHP embedded in HTML

```
<?php   if ($_POST["user"] != "vmueller")  {      ?>
        <b>You are not allowed to login. </b>
<?php exit ( );   }
          else {                                          ?>
                <b>Welcome. </b>
<?php   session_start ( );
          $_SESSION["login"] = true;   }      ?>
```

Example
PHP 5.x Function

```
function slotnumber ( )
 {
   srand( time() );
   for ($i = 0; $i < 3; $i++)
      {  $random = (rand() % 3);  $slot[$i] = $random;  }
   print("<td width=\"33%\">$slot[0]</td>");
   print("<td width=\"33%\">$slot[1]</td>");
   print("<td width=\"33%\">$slot[2]</td>");
....
}
```

Example: Typed functions in PHP 7.x

```
function sum (int $a, int $b) : int
```

```
function sum (… $numbers) : int
```

```
function sum (int … $numbers) : int
```

You can loop over arbitrary number of parameters with foreach loop

Usually wrong-typed input parameters casted to desired type

Strict mode forces error: `declare (strict_types=1);`

Sessions

Way to preserve certain data across subsequent accesses (note: HTTP is stateless !)

Session identified by session id (often stored in a cookie)

Example of "Superglobal"

Session data stored in $_SESSION["key"], physically stored in file on the server / user-defined (e.g. in a DB) with a SessionHandler

Accessing Files

PHP can read / write files:

```
$fp = fopen ("test.txt", "r");

while (! feof ($fp))

   $buffer = fgets ($fp, 4096);

fclose ($fp);
```

fopen even works for URLs (usually read-only)

Newer & easier way: file_get_contents(...)

Regular Expressions

Way to describe a set of strings based on common characteristics shared by each string in the set

Combination of characters sets and quantifiers

Examples: [abc] = a OR b OR c

[a-z0-9] = small character OR digit

a* = a block with ≥ 0 a's

What can we do with REs?

Check whether a input string has a substring of a special form (defined by a RE)

Extract such substrings

Replace such substrings by something else

Some Tutorials for Regular Expressions

www.regular-expressions.info/tutorial.html

regexone.com

www.tutorialspoint.com/php/
php_regular_expression.htm

Character Sets (1)

[abc] a, b, or c (set of characters)

[^abc] any character except a, b, c (negation)

[a-zA-Z] a through z, OR A through Z (range)

[a-z&&[def]] d, e, or f (intersection, AND)

[a-z&&[^bc]] a to z, except for b, c (subtraction)

Character Sets (2): Simplifications

.	Any character
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Escaped Characters

Characters which are used for defining REs can itself be represented by adding a backslash before:

RE for a dot	\.
RE for [	\[
RE for \	\\

Quantifiers

X | Y	X OR Y
X?	X, once or not at all
X*	X, zero or more times
X+	X, one or more times
X{n}	X, exactly n times
X{n,}	X, at least n times
X{n,m}	X, at least n, not more than m times

[X, Y are RE sets]

Note: \d+ = "non-empty block of digits" does not mean that all digits must be equal

Boundaries

^ The beginning of a line or input (default)

$ The end of a line or input (default)

Application: if you want ensure that a complete string satisfies form, then you enclose RE inside ^...$, similar for prefix / suffix

Greedy versus Reluctant

Input: RE .*b with input string aabaaaab

Greedy: find the longest substring that matches RE

Reluctant: find the shortest substring that matches RE

RE Quantifiers are per default greedy

To make then reluctant, add ? after quantifier

Using RE: preg_match (1)

Return whether input string contains substring of some form:

```
$subject = "abcdef";

$pattern = '/^def/';

echo preg_match ($pattern, $subject);
```

Note: Regular expressions given inside / /

Using RE: preg_match (2)

Can also be used to extract strings:

```
preg_match ( '/^http:\/\/([^\/]+)/i',
                    "http://www.php.net/index.html",
                    $matches);
```

matches[0] = complete string that matches
matches[i] = substr. in i-th bracket pair (i > 0)

First Homework Exercise

Use regular expressions to extract information (#COVID-19 cases) from websites

Last year exercise on Moodle: extract DAX value from website

PHP and MySQL

MySQL is popular open-source Relational DBMS (http://www.mysql.com)

PHP provides (many) functions for directly using MySQL, i.e. connect to MySQL server, choose database, send query, access result of query, etc.

Old mysql methods have been replaced in PHP7 with improved framework mysqli

Improvements in mysqli

Object-oriented interface

Support for Persistent Connections

Support for Prepared Statements

Support for Multiple Statements

Support for Transactions

Enhanced debugging capabilities

Example: Connect to DB

```
$I = new mysqli ('localhost', 'USER', 'PWD', 'DB');

if ($I->connect_errno)  { die ('Could not connect'); }

    // HERE ARE THE QUERY COMMANDS

$I->close ( );
```

Persistent Connections

Persistent connection = connection between client process and database can be reused by client process, rather than being created and destroyed multiple times

Reduces overhead of creating fresh connections every time one is required, as unused connections are cached

Open persistent connection: prepend “p:” to hostname when connecting

Accessing Form Data

Users input information in HTML in a form:

```
<form action="t.php" method="post">
    <input type="password" name="pwd" />
    <button type="submit">Submit</button>
</form>
```

Key used in superglobal

Entered information accessible in t.php in superglobals $_GET[ ], $_POST[ ], or $_REQUEST[ ] (depending on HTTP method)

Example: Check Login Info (Basic Idea)

```
// assume: form data in var. $account, $pwd given

$query = "SELECT * FROM loginTBL WHERE
account = '" . $account . "' AND pass = '" . $pwd . "' ";

If (! ($res = $l->query ($query)))
    die ("ERROR in query");

if ($res->num_rows == 0)
    print ("Login failed"); …..
```

NOTE: contains security flaws !!

Some Security Aspects (I)

Passwords should never be stored in cleartext in a database

Better: encrypted or salted hashed

Appropriate hash function password_hash available in PHP, several alternatives can be set in MySQL

Security (II): Example SQL Injection

```
$query = "SELECT * FROM loginTBL WHERE
account = '" . $account . "' AND pass = '" . $pwd . "' ";
```

Assume input: $account = vmueller

$pwd = test' or '1' = '1

Query becomes

```
SELECT * FROM loginTBL WHERE
    account = 'vmueller' AND pass = 'test' or '1' = '1'
```

Result: if “vmueller” is valid account, then at least one row returned, even if valid password not known

Measures against SQL Injections

Countermeasure: Always escape SQL special characters in a string before use

PHP function: mysqli::real_escape_string

Rule: Always apply this function to the parts of query based on user input (from forms, cookies, URL parameters, …)

Alternative (preferred): prepared statements, use stored procedures

Example: Prepared Statement

```
$stmt = $l->prepare ("SELECT District FROM City
   WHERE Name=? ") ;

$stmt->bind_param ("s", $city);

$stmt->execute ( );

$stmt->bind_result ($district);

$stmt->fetch ( );                    // can be done also in a loop

printf("%s is in district %s\n", $city, $district);
```

Prepared statements escape arguments automatically during binding !!

Example: Looping over Result $res (without binding)

```
while ($row = $res->fetch_assoc())
   { echo $row['first_name'] . ' ' . $row['last_name'];  }
```

Transactions:

```
$I -> autocommit (FALSE); ….
$I -> commit ( );     //   $I -> rollback();
```

For all methods, see

http://php.net/manual/en/class.mysqli.php

PDO – PHP Data Objects

Lightweight object-oriented interface to access arbitrary DBs (similar idea as JDBC for Java)

```
$db = new PDO
('mysql:host=localhost;dbname=D;charset=utf8',
'user', 'pwd'); ….

$res = $db->query("SELECT * FROM T");

foreach($res as $row)

    echo $row['field1'] . '  ' . $row['field2'];
```

Strings in PHP

```
$a = "Test: x = $x";          –> Var. $x expanded

$a = 'Test: x = $x';          –> Var. $x not expanded

$a =  <<< EOT

   Text for many lines ....

  EOT;
```

Variables in "Heredoc" are expanded, no expansion in 'Nowdoc' ('EOT')

Also possible var. notation: ${x}

Variable variables and more

Variable variables:

```
$a = "var";
```

```
$$a = "B"  corresponds to $var = "B"
```

References:

```
$a =& $b;   // a and b point to same content
```

User-defined Functions

```
function foo($arg1, &$arg2, $arg3 = "T")
{   …  return $retval;    }


function foo( )                  // PHP 5.x, simplified in PHP7
{ $numargs = func_num_args();
 if ($numargs >= 2) echo func_get_arg(1);
}
```

Variable Functions - Closures

```
$func = 'foo';

$func();     // calls function foo( )
```

```
echo preg_replace_callback('/-([a-z])/', function
($match) {
    return strtoupper($match[1]);
}, 'hello-world');
```

Next Week

Object-oriented Programming in PHP