Faculty of Science,
Technology
and Medicine

Web Programming

Volker Müller
University of Luxembourg

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

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 ( );
```

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 !!

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');
```

OO Programming with PHP

PHP supports "standard features" of OOP:

- Classes and inheritance
- Static class members
- Variable Visibility: public, protected, private
- Abstract class, exceptions, ….
- Namespaces (stored in directories, access uses "\")

Example PHP Class

```
class A {
 public $a;
 public static $b;
 const C = 1;
 public function f( )
        {    return ($this->a + self::$b + self::C);    }
}
print A::$b; print A::C;
$a = new A(); print $a->a;  print $a->f();
```

Common OOP Keywords

- Constructor: __construct (…) { … }
- Destructor: __destruct( ) { … }
- Visibility: public, protected, private
- Inheritance: extends, final
- "Overloading" (dynamically adding properties): __set($name, $value); __unset($name)
- Exceptions: Exception(...), try – catch
- Abstract classes, interfaces + implements

Autoloading Classes

Often, developers write one class per file (the Java approach)

Using these classes requires ALL files to be manually included into using file (with include command) → prone to errors

Better solution: Function spl_autoload_register can be defined to automatically load classes on first usage

Autoloading Example

Add to one file which is always included the code:

```
spl_autoload_register(function ($class) {
    include 'classes/' . $class . '.class.php';
});
```

Loads file <CLASSNAME>.class.php from sub-directory "classes"

Example uses a closure function

Traits: Code Reuse in Single Inheritance Languages

Single inheritance = class can only have one direct super-class

Traits enable developer to reuse sets of methods freely in several independent classes living in different class hierarchies

Traits can not be instantiated (similar to abstract classes)

Traits Example

```
trait T      {  function A () { … some code … }  ….   }


class test extends B {

  use T; ….

}
```

→ method A() available in test like normal member function

Priority: local methods override Trait methods override inherited methods

Namespaces

Namespaces were added to PHP to allow grouping files in different directories (like packages in Java)

Namespace names are case-insensitive !!

Namespace must be declared at top of code

Ex: namespace MyProject\Sub\Level;

→ File stored in directory MyProject/Sub/Level

Reference Code in Namespace

Use fully qualified name: My\Sub\Level\foo()

Use partial name in dir. "My": Sub\Level\foo()

Equivalent of "import":

```
use My\Full\Cname as c;

use My\Full\Cname;    // equivalent to as Cname

use function My\Full\functionname as f;
```

PECL

PECL provides extensions to php

PECL extensions have to be compiled and stored as dynamic libraries, then loaded by php executable → compiler infrastructure must be available on machine

https://pecl.php.net/ contains information about available extensions

See Docker definition for Nginx-PHP-MongoDB as usage example

PHP-related Tools – PHP Archives

PHP Archives (file extension .phar): put entire PHP applications into a single archive file for easy distribution and installation

Created either by IDE or with command phar included in PHP installation

PHP class "Phar" also exists for dynamic generation

PHP-related Tools – Phing

Phing: PHP project build system based on Apache ant, manages testing, package generation, documentation generation

Rich set of provided standard tasks, applied in XML configuration files

Available at https://www.phing.info/

PHP-related Tools - Composer

Composer: Dependency Manager for PHP, allows you to declare the libraries (with PHP code) your project depends on and it will manage (install/update) them for you:
https://getcomposer.org/

Define dependencies in composer.json:

```
{   "require": {   "monolog/monolog": "1.0.*"  }   }
```

→ Run php composer.phar install

Default repo: packagist.org

PHP-related Tools - PHPUnit

PHPUnit = programmer oriented testing framework for PHP (http://phpunit.de)

Can be nicely integrated with composer

Uses annotations for meta-information

Idea similar to JUnit used for Java unit tests

Example TestCase

```
use PHPUnit\Framework\TestCase;

class DataTest extends TestCase
{
    /**   @dataProvider additionProvider  */
    public function testAdd($a, $b, $expected)
    {  $this->assertSame($expected, $a + $b);   }

    public function additionProvider()
    {   return [  [0, 0, 0],   [1, 1, 3] ];    }
```

PHP Annotations

Annotations = meta-data embedded in source code

PHP-Annotations are no official part of PHP, but used quite often

Similar form as in Java: @var

But: Annotations must be written inside comments

Add to project: composer require annotations

More details: php-annotations.readthedocs.io

Next Week

Web Services with PHP

PHP and NoSQL Databases