Multimedia im Netz

Online Multimedia

Winter semester 2015/16

Tutorial 02 – Minor Subject

Today's Agenda

- Server side scripting: PHP Basics
  - PHP Syntax and Core Concepts
  - Forms
- Quiz

PHP Basics

PHP: Hypertext Preprocessor

- Server-side scripting language dating from 1995
- Current stable version 5.6.14
- Official website: http://php.net (logo source)

- To get you started: tutorials (just a few examples)
  - http://tut.php-quake.net/en/
  - http://www.php-einfach.de/php-tutorial/php-tutorial.php (German)
  - http://www.w3schools.com/php/

PHP at the CIP-Pool (I)

- PHP usage is restricted:
  http://www.rz.ifi.lmu.de/Merkblaetter/homepage.html
- To facilitate correction of your assignments, they need to work in the CIP pool:
  - PHP version 5.5.9
  - Put into directory `public_html/php`
  - Usage under your personal webspace
    (replace “login” with your CIP-account name):
    http://php.cip.ifi.lmu.de/~login/php/skript.php
  - You can only put PHP files in this directory. If you use images, you have to put them in `public_html` and other subdirectories

PHP-Server at the CIP-Pool (II)

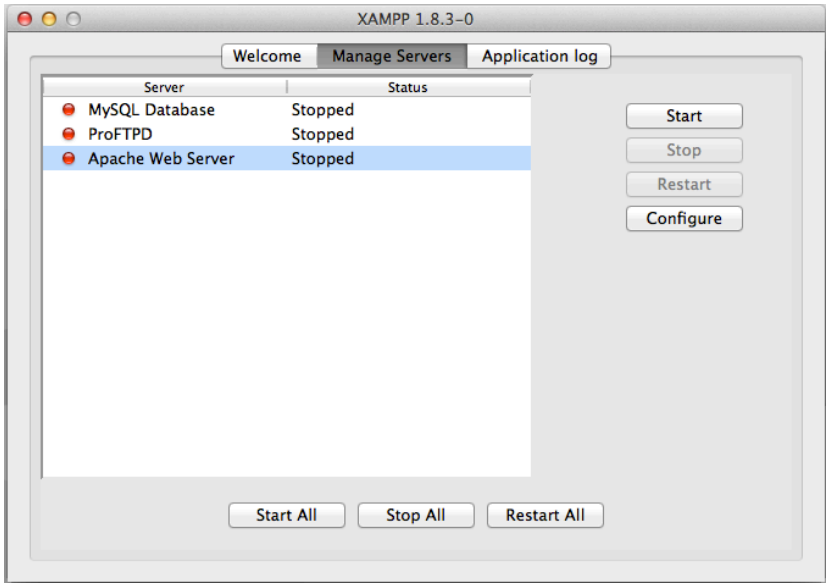
- Server is only accessible in the WAN / CIP Pools
  - Via an ssh-tunnel
    http://www.rz.informatik.uni-muenchen.de/FAQ/Aussenzugriff.faq.html
  - VPN: https://www.lrz.de/services/netz/mobil/vpn/

- Nicely working solution: Remote Desktop Connection
  - Instruction here http://www.rz.ifi.lmu.de/Dienste/rdp.html/
  - Does not work with “Starter” Versions of Windows.

B.Y.O.D.

XAMPP

- You can use your own machine and install a web server there (Apache)
- XAMPP: Convenient bundle
- e.g. LAMP:
  Linux, Apache, MySQL, PHP,



- Get it here for Windows, Mac, Linux:
  https://www.apachefriends.org/download.html
- See if it works: http://localhost

Apache doesn’t start?

- Make sure to:
  - Check the port that is configured in httpd.conf / apache2.conf
  - Apache usually listens on port 80
  - Quit Skype (it sometimes listens on port 80/443)
  - On Unix-based systems this command shows you which ports are already taken:
    ```
    netstat -ntlp | grep LISTEN
    ```
- If you use Jetbrains phpStorm, it has a built-in Webserver. You only need to tell it where the php binaries are.

Hello World!

Create the file **test.php** or use the one provided on GitHub:

```
<?php
    echo "My first PHP script!";
?>
```

On a CIP-pool machine:

1. Put it into `public_html/php`
2. Open a web browser and go to http://php.cip.ifi.lmu.de/~login/php/test.php
3. It should say “My first PHP script!“
4. Collaborate with your neighbor if there are any problems.

Embedding PHP into HTML

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8"/>
    <title>PHP embedded into HTML</title>
</head>
<body>
<h2>
    <?php echo "My Heading"; ?>
</h2>
<?php echo "<p>My paragraph</p>"; ?>
</body>
</html>
```

Syntax

- PHP can be **embedded** into HTML Documents.

```
<?php ... ?>
```

- **Variables** are prefixed with a $-sign:

```
$someVar = 5;
```

- **Printing text:**

```
echo "Sometext";
echo "Even <b>HTML</b> can be printed";
```

- **Concatenation** is done with a dot!

```
echo "Variable content: " . $someVar;
```

- **Comments**

```
// This is a comment
/* This is a comment
that spans multiple lines! */
# I can't get enough of those comments!
```

Variables Inside Double-Quoted Strings

```php
<?php

$currentTime = date("d.m.Y, H:i:s", time());

echo "It's $currentTime";

?>
```

Types and Operators

- PHP is weakly / dynamically typed
- Data types: Boolean, Integer, Float, String, Array
- Arithmetic operators:

  `+  -  *  /  %`
- Bit-operators:

  `&  |  ^  ~  <<  >>`
- Comparison:

  `==  ===  !=  <>  <  >`
- Increment and decrement operators:

  `++$a  $a++  --$a  $a--`
- Logic operators:

  `&&  ||  !  XOR`

What output does this generate? (1)

```php
<?php

// 1:
echo 1 + "10 little pigs";


// 2:
$test = 2 . "10 little pigs";
echo $test;

// 3:
echo 3 , "10 little pigs";

?>
```

Conditional Statements

- If-else:

```
if($a > $b){
    echo "a is greater than b";
} else {
    echo "a is not greater than b";
}
```

- Ternary operator (syntactic sugar)

```
echo $a > $b ? "a greater than b" : "a less than b";
```

What output does this generate? (2)

```php
<?php
$intZero = 0;
$stringZero = "0";

if($intZero == $stringZero)
        echo "== Equal";
else    echo "== unequal";



if($intZero === $stringZero)
        echo "=== identical!";
else    echo "=== unidentical!";
?>
```

Arrays – Part 1 – Warm-up

- Data type for variables containing multiple values at once
- No fixed size (compare to other programming languages!)
- Useful to group values/data logically
- Creating arrays:
  - Empty index-based array:
    ```
    $fruits = array();
    ```
  - Array with initial values:
    ```
    $veggies = array("lettuce","turnip","beets");
    ```
  - Associative array:
    ```
    $fruitColors = array(
        "banana"=>"yellow",
        "strawberry"=>"red",
        "apple"=>"green"
        )
    ```

Arrays – Part 2 – The nitty gritty

- Index based arrays:
  - adding values at the end: `$fruits[] = "apple";`
  - changing values at given index: `$fruits[4] = "banana";`
  - accessing values, e.g.: `echo $fruits[0];`

- Associative arrays
  - adding values: `$fruitColors["cherry"] = "dark red";`
  - removing values: `unset($fruitColors["apple"]);`
  - accessing values, e.g.: `echo $fruitColors["banana"];`

Useful Array Functions

- **count**: returns the number of elements in the array
- **array_search**: searches an array and returns found index
- **in_array**: determines if a value exists in the array
- **shuffle**: shuffles an array

- Example:

```
count($fruits);
array_search("pear",$fruits);
```

While Loops

- Example

```
$isHomerHungry = true;
while($isHomerHungry){
    echo 'Homer is still hungry. ';
    $isHomerHungry = (rand(0,10) != 10);
}
echo "<p>Homer is not hungry
anymore</p>";
```

- Make sure to find a reliable break condition.

For / Foreach Loops

- ```
  for($donut=1;$donut<=10;$donut++){
      echo "Homer is eating donut $donut";
  }
  ```

- Foreach:
  ```
  $donuts = array(
      "sprinkled",
      "maple",
      "glazed");

  foreach($donuts as $donut){
      echo "Homer likes $donut donuts. ";
  }
  ```
- `break`: terminates the execution of the loop.
- `continue:` current loops is interrupted and the loop continues with the next iteration.
  ```

Break Out

- Write a small script that…
  - takes an array of arbitrary length
  - doubles the value stored in the array
  - prints the doubled value

- If you have time:
  - consider what to do if the array contains strings.
  - the values are stored in an associative array. How do you access them?

- Take 20 minutes time.

Functions

- Void function:

```php
function someFunction($parameter1, $parameter2){
    // do something
}
```

- With a return value:

```php
function square($number){
    return $number * $number;
}
echo square(4);
```

Interactive Webpages with PHP

PHP + Forms

- PHP can handle user input, but only *after* it was sent to the server, where the script is executed.
- Typical user input comes from HTML <form> elements
- There are many different input elements (see next slide)

<input type="..." />

radio

- Red
- Green
- Blue

checkbox

- [x] Cream
- [x] Milk
- [ ] Sugar

text

Your text: my text

button

Hardest button to button.

file

Choose File No file chosen

password

Password:

Example Form: Favorite Color

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8”>
    <title>Favorite Color!</title>
</head>
<body>
<p>Please pick your favorite color:</p>
<form>
    <label> <input type="radio" name="color"/> Red
    </label>
    <label> <input type="radio" name="color"/> Green
    </label>
    <label> <input type="radio" name="color"/> Blue
    </label>
</form>
</body></html>
```

Passing Data between Browser and Server

- The example from the previous slide, allows the user to make a selection, that is, to **enter data**
- How do we pass it from the user's browser to the server, where we can evaluate the data?
  - action="..." attribute tells where the data should go
  - method="... " attribute tells how it should be "wrapped"
- An `<input type="submit" />` sends the form

Extending the Form: Action, Method, Submit

```
<form action="formExample.php" method="post">
    <label>
        <input type="radio" name="color"/>
        Red
    </label>
    <label>
        <input type="radio" name="color"/>
        Green
    </label>
    <label>
        <input type="radio" name="color"/>
        Blue
    </label>
    <input type="submit"
           name="submit"
           value="Save" >
</form>
```

Extending the Form: Values

```
<label>
    <input  type="radio" name="color"
            value="red" />
    Red
</label>
<label>
    <input  type="radio" name="color"
            value="green" />
    Green
</label>
<label>
    <input  type="radio" name="color"
            value="blue" />
    Blue
</label>
<input type="submit" name="submit"
       value="Save”>
```

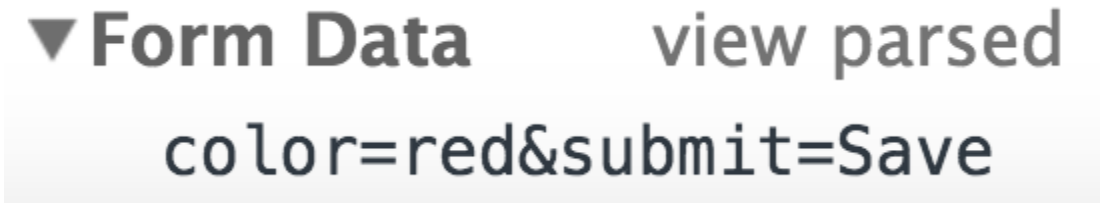
Extending the Form: Control output

```php
<?php
if(isset($_POST['color'])) {
    echo "<p>Your favorite color is " . $_POST['color'] . "</p>";
}
else{
    ?>
    <p>Please pick your favorite color:</p>
    <form action="formExample_finish.php" method="post">
        <label>
            <input  type="radio" name="color"
                    value="red" />
            Red
        </label>
        <label>
            <input  type="radio" name="color"
                    value="green" />
            Green
        </label>
        <label>
            <input  type="radio" name="color"
                    value="blue" />
            Blue
        </label>
        <input type="submit" name="submit"
               value="Save">
    </form>
<?php } ?>
```

GET & POST

- GET
  - Query string is sent in the URL of the request: http://localhost/test.php?lecture=onlineMultimedia
  - Parameters are visible to the user!
  - Superglobal variable in PHP: `$_GET` (Associative Array!)

- POST
  - Query string is sent in the HTTP message body of the request
  - Superglobal variable in PHP: `$_POST` (Associative Array!)



Comparison

GET Requests	POST Requests
can be **cached**	are **never cached**
stay in the **browser history**	do **not** show up in the browser history
can be **bookmarked**	**cannot** be bookmarked
have a **fixed length**	do not have length restrictions
should be used to **retrieve** data	should be used to **modify** data
should **not** be used with **sensitive data**	are a little safer for sensitive data

http://www.w3schools.com/tags/ref_httpmethods.asp

Useful String Functions

- **strlen**: returns the length of a string
- **strstr**: finds the first occurrence of a substring
- **substr**: returns a substring
- **htmlspecialchars**: converts special characters to HTML codes
- **strip_tags**: removes all PHP and HTML tags from a string
- **explode**: splits a string and returns an array with the chunks
- **implode**: takes an array and concatenates the fields to a string
- **str_replace**: replaces all matches with a replacement string

Link Collection

- https://secure.php.net/docs.php
- http://www.w3schools.com/php/php_intro.asp
- https://www.codecademy.com/courses/web-beginner-en-StaFQ

- IDEs:
  - https://www.jetbrains.com/phpstorm/
  - http://www.aptana.com/products/studio3/download.html
  - https://netbeans.org/features/php/
- Useful text editors:
  - https://www.sublimetext.com/

Round-up Quiz

1. Is a PHP script evaluated in the browser or somewhere else?
2. Is PHP typed statically or dynamically?
3. How do you concatenate strings in PHP?
4. What's the difference between the == and === operator?
5. What's going on here:
```
$grades = array('johnson'=>1.0);
$grades['smith'] = 3.0;
```
6. Is GET or POST more suitable for transmitting passwords? Why?
7. Is the correct syntax `count($array)` or `$array.count()`?

Thanks!

What are your questions?

Let’s begin with the Assignment!

- Download the assignment sheet
- Start with task 1
- You can collaborate with your neighbor

- Turn in the assignment by November 4th, 12:00 noon via UniWorX