Multimedia im Netz
Online Multimedia
Winter semester 2016/17

Tutorial 02 – Major Subject
Today’s Agenda

• Recap
• Quick-git
• Quasi stateful web apps with PHP
  – Cookies
  – Sessions
• Object-oriented PHP
• Quiz
Recap

1. Why do we need an Apache server for PHP?
2. What’s a difference between single-quote and double-quote Strings?
3. Can you embed HTML markup inside PHP strings?
4. How do you concatenate two strings?
5. What’s the difference between index-based and associative arrays?
6. What’s the difference between GET and POST?
7. How do you access the transmitted form data in PHP?
Quick git to get tutorial materials

- Open Git Bash (Windows) or a Terminal (Linux/macOS)
- Type `cd && mkdir mmn && cd mmn`
  This will create a new Directory called "mmn" in your home folder and then step into it.
- Type `git clone git@github.com:MIMUC-MMN/tutorials-16-17.git`
  If you are asked if you want to trust the server with the signature, type `y` or just hit Enter
- Type `cd tutorials-16-17`
- Type `pwd`
  This will show you the path of the local repository, so something like this: `/Users/<your_name>/mmn/tutorials-16-17`
- Next time you want an update of the materials, just open a terminal and do this:
  `cd /Users/<your_name>/mmn/tutorials-16-17 && git pull`
PHP and Statefulness: Sessions and Cookies
Break Out

• Visit any website you like.
• Find out, which cookies are transmitted.
• What information is inside the cookies?

• Group discussion: What are the dangers of cookies, when are they harmless?
Problem: HTTP is stateless
Cookies

• Goals:
  – Persist information on the client side
  – Identify client to make communication stateful

• HTTP cookies:
  – Stored in browser
  – Usually small, serialized data (text)
  – Sent with all request headers depending on current host URL

• Example usages:
  – Items in a shopping cart
  – Measure interaction (navigation on a site)
  – Authentication
Cookies are...

• ... not necessarily evil. Typical myths:
  – Malware containers
  – Viruses
  – Spam

• ... not good to store large data on the client
  – only small, serializable chunks
  – use the local storage API instead

• ... not dependent on server-side scripting
  – Also available with JavaScript
Sidenote: The Cookie Dilemma

- There is a “cookie law” in the EU that requires website operators to inform the visitors about the use of cookies.
- Users do not necessarily read / understand / want this
- **Almost all sites require cookies**

https://www.cookielaw.org/the-cookie-law/
Cookies in PHP

• Cookies belong to the HTTP header
  – Must be set before any output is generated
  – Before the <html> tag
  – Before any print / echo / var_dump statements!

• Create a cookie: `setcookie(...)`
  `setcookie("MMNCookie","Hello statefulness!");`

• Read a cookie:
  `var_dump($_COOKIE['MMNCookie']);`
  – reading is done on the server
  – so reading only works after the cookie is sent back to the server!
  – that is, after refreshing the page after cookie was set
Example: PHP cookies

```php
<?php
if(isset($_POST['name'])){
    setcookie('Name',$_POST['name']);
}
?>
<!DOCTYPE html>
<html>
[...]
<body>
<?php
if(isset($_COOKIE['Name'])){
    echo '<h1>Hello ' . $_COOKIE['Name'] .'</h1>';
}
?>
<form method="post">
    <label>Name: <input type="text" name="name"/></label>
    <input type="submit" />
</form>
</body></html>
```
Sessions

- Cookie disadvantage: Only stored on the client (browser)
- Session paradigm maintains “states” on the server side
- Goal:
  store current state of variables across multiple requests

- The client actually doesn’t know about the session. It only receives a so called session (ID) cookie:
  - default cookie name in PHP: PHPSESSID
  - renaming possible with session_name()
  - as all cookies, transferred with all requests to the corresponding site
Sessions with PHP

• Sessions need to be started **before any output occurs** (similar to sending cookies)
• Create session ID cookie: `session_start()`
• Delete the session ID cookie: `session_destroy()`
• Read / write session values:
  – superglobal `$_SESSION` array
  – immediately reset session like this `$_SESSION = array();`
Example: Counting visits

```php
<?php
session_start();

<!DOCTYPE html>
<html>
[...]
<body>

<?php

if(!isset($_SESSION['count'])){
    $_SESSION['count'] = 1;
}
else{
    $_SESSION['count']++;
}

echo '<p>Current count: '.$_SESSION['count'].'</p>';

?>
</body></html>
```
Example: Destroying Sessions

```php
<?php
session_start();

<!DOCTYPE html>
<html>
[...]
<body>

<?php

if(isset($_POST['destroy'])) {
    session_destroy();
    $_SESSION = array();
}

if(!isset($_SESSION['count'])){$_SESSION['count'] = 1;
} else{
    $_SESSION['count']++;
}

echo '<p>Current count: '.$_SESSION['count'].'</p>';

?>
<form method="post">
    <input type="submit" name="destroy" value="Reset"/>
</form>
</body>
</html>
```
Break Out: Hangman

• Create a “hangman” game with PHP.
• You can use the code skeleton from the git repo in /tutorial02/breakout/hangman
• Complete the lines where you find a TODO

• Take 25 minutes time
Object Oriented Programming in PHP
OOP Basics

• OOP paradigms and concepts (examples):
  – classes that turn into objects when instantiated
  – inheritance
  – interfaces
  – „Everything is an object“

• PHP class signature:

```php
<?php
class Lecture{
    //put members and methods here
}
?>
```
Define Member Variables

```php
<?php
class Lecture{
    var $title = "Online Multimedia";
    var $semester = "winter 2016/2017";
    var $professor = "Prof. Dr. Heinrich Hussmann";
    var $date = "Thursdays 10-13h";
}
?>
```
Using a Class

- Importing and instantiating a class:
  ```php
  require_once("Lecture.php");
  $mmn = new Lecture();
  var_dump($mmn);
  ```

- Access to member values: arrow notation
  ```php
  echo 'Title: ' . $mmn->title;
  echo 'Semester: ' . $mmn->semester;
  echo 'Professor: ' . $mmn->professor;
  ```
Adding Methods to Classes

```php
<?php
class Lecture{
    var $title = "Online Multimedia";
    var $semester = "winter 2016/2017";
    var $professor = "Prof. Dr. Heinrich Hussmann";
    var $date = "Thursdays 10-13h";

    function setDate($date){
        $this->date = $date;
    }

    function getDate(){
        return $this->date;
    }
}
?>
```
Calling methods

$mmn->setDate("Thursday morning");
echo $mmn->getDate();
Constructors

• PHP’s constructors are methods with a special name: `__construct();`

```php
function __construct($ttl, $sms, $prf, $dt){
    $this->title = $ttl;
    $this->semester = $sms;
    $this->professor = $prf;
    $this->date = $dt;
}
```

• Use constructor:
```php
$mmi = new Lecture("MMI",
    "Winter semester","Prof. Butz","Wendesdays");
```
Round-up Quiz

1. What does it mean to “serialize data”? 
2. Why are cookies only accessible after a page refresh? 
3. What is the difference between “cookies” and “sessions”? 
4. Are cookies stored on the server or on the client? 
5. What does session_destroy() actually do? 
6. How do you define a member variable for a class? 
7. How do you access a method with a given object?
Thanks!

What are your questions?
Appendix