Multimedia im Netz
Online Multimedia
Winter semester 2016/17

Tutorial 02 – Minor Subject
Today’s Agenda

• Warm Up:
  – Recap
  – Missing parts from last tutorial (Interactive Web Apps)

• Cookies

• Sessions
  – How to use sessions to persist variable values across multiple requests
  – Break out: Parking Lot Monitor

• 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?
PHP and Statefulness: Sessions and Cookies
• 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 web site 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/
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 and identify the client.

- 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>
```
Session Functions - Overview

- **session_start()**
  Start or resume a session

- **session_destroy()**
  Destroy all data from a session, including session ID and cookies (only after page refresh!)

- **session_unset()**
  Free all session variables, but maintains the session ID

- **session_name(...)**
  Get or set the session name

last access on 02/11/2015
Example: Resetting Sessions

```php
<?php
session_start();

<!DOCTYPE html>
<html><head lang="en">
    <meta charset="UTF-8">
    <title>Session Reset</title>
</head>
<body>

<?php
    echo session_name() . '<br />
    echo session_id() . '<br />

$_SESSION['answer'] = 'yes';
session_unset();

echo session_id() . '<br />
    echo $_SESSION['answer']. '<br />
    // ?

$_SESSION['answer'] = 'yes';
    echo $_SESSION['answer']. '<br />
    // ?
?>
</body></html>
```
Break Out: Parking lot counter

• Imagine you are the gatekeeper at a parking lot.
• The parking lot holds exactly 10 spots.
• You need to keep track of the occupied spots, so you use a web app that has a +1 and -1 button.
• The page somehow shows the currently occupied spots.
• If the count reaches the maximum number, the +1 button is disabled (inactive). The same is true, if no cars are at the parking lot.
• Use PHP-Sessions to maintain the current count.

• Take approx. 30 minutes time
• Start with the material that you can download on the web page.
Demo

Watch the demo here: https://youtu.be/vw4nUZYKLrw
Round-up Quiz

1. Why are sessions necessary?
2. Can you initialize a session only at the beginning of a script?
3. Are (session-)cookies stored on the server or on the client?
4. What is the difference between `session_destroy()` and `session_unset()`?
5. Why do you need to refresh the page to see the effects of `session_destroy()`?
Thanks!
What are your questions?
Appendix