Westfälische Wilhelms-Universität Münster

Erstellen dynamischer Webseiten mit PHP

WS 2016/2017

wissen.leben
WWU Münster

Dozent: Patrick Förster, Michael Hasseler

Übersicht der heutigen Inhalte

- Session-Management
  - Session starten, fortsetzen, löschen
  - Sessiondaten speichern, abrufen, löschen
- Cookies
  - setzen
  - löschen
- Exception Handling
  - try-catch-finally
  - throw

Http ist zustandslos

- Angenommen Sie sollen ein Warensystem in PHP umsetzen
- Bis zum endgültigen Kauf sind einige Daten zu erfassen:
  - Waren, Personendaten, Liefer- und Rechnungsadresse, Zahlverfahren
- Klar ist: Alle vorhanden Waren auf einer einzigen Seite darzustellen ist nicht erstrebenswert
- Ein „Warenkorb“ muss seine Daten über mehrere Seiten „behalten“
- Die Abfrage der Personendaten sowie Adressen und Zahlverfahren ist ebenfalls meist auf verschiedene Seite verteilt
- Die Daten müssen über mehrere Anfragen gespeichert werden
- Damit diese Daten auf Serverseite nach einem Request behalten werden, bedarf es einer Art temporärer Speicherung
- Diese wird „Session“ genannt

Session starten

- Eine Session wird innerhalb eines Skriptes mit **`session_start`** gestartet
- Es empfiehlt sich dies bereits zu Anfang eines jeden Skriptes zu machen

```
<!DOCTYPE HTML>
<?php
session_start();
?>

<html>
…
</html>
```

- Jeder Session eine zufällige eindeutige 32 stellige ID zugeordnet
- Diese ID muss auf Client-Seite zur Verfügung stehen und wird mit jedem Request an den Server gesendet

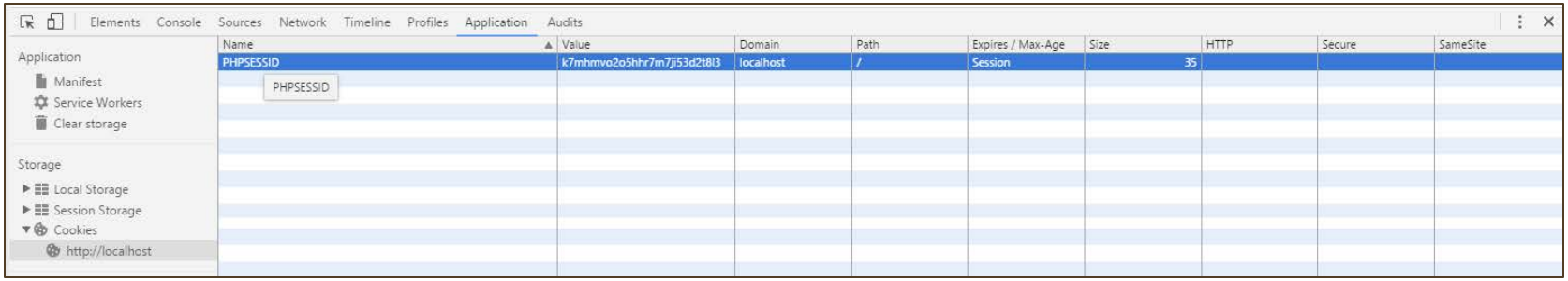
Session Speicherung

- Es gibt mehrere Möglichkeiten die Session-ID zu übermitteln
  - Per Cookie
  - Per GET-Parameter
  - Per Hidden-Input Field (GET oder POST)

- Per Default nutzt PHP Cookies zur Speicherung auf Client-Seite
- Ein Cookie ist ein Name➔Wert Paar, das auf Client-Seite in einer Datei abgelegt wird
- Ein Cookie ist eindeutig einer Domäne zugeordnet
- Nur die zur Domäne gehörenden Cookies werden bei einem Request mitgesendet
- Der Cookie-Name ist per Default „**PHPSESSION**“

Cookies im Browser (Chrome/Windows)

- Im Chrome-Browser mit **Strg+Shift+J** die Entwickler-Tools öffnen
- Tab „Application“, links „Cookies“ aufklappen und „http://localhost“ wählen



- Der Wert ist die Session-ID
- Der Cookie-Name lässt sich mit `session_name` setzten

```
<?php
session_name("ZIV");
sessin_start();
…
?>
```

Session auf dem Server

- Eine Session ist ein assoziativer Array
- Der Zugriff erfolgt über die super-globale Variable **$_SESSION**

```
<?php
session_name("ZIV");
sessin_start();

if (!isset($_SESSION["seite"])) {        // Session-Wert lesen
    $_SESSION["seite"] = "start";        // Session-Wert setzen
}

?>
```

- Am Ende des Skriptes wird der Session-Array gespeichert
- Default: Schreiben in eine Datei mit dem Namen **„sess_“ Session-ID**
- Speicherort (Windows, XAMPP, Default): **XAMPP\temp**

Session per GET/POST

- Nicht jeder Nutzer lässt Cookies zu
- In diesem Fall erhält man über die super-globale Variable SID Name und Wert der ID

```
<?php
session_name("ZIV");
sessin_start();
echo SID;
?>
➔ ZIV=2c39bgh247vbfu3o209sgr4r55
```

- Dieser Wert wird dann an alle Links angehangen, die zu einem Skript führen, das die Session nutzen soll

```
<?php
echo '<a href="b01.php?' . SID. '">Weiter</a>';
?>
```

Session per GET/POST (II)

- Gleiches funktioniert auch für das `action`-Attribut eines `<form>`-Tags

```
<form action="b01.php?<?=SID?>" method="POST">…</form>
```

- *Bemerkung: `<?=VARIABLE?>` ist eine abkürzende Schreibweise, zur Auswertung eine PHP-Variablen, außerhalb von `<?php…?>`*

- Alternativ innerhalb einer Form eine „versteckte Eingabe“ nutzen:
- Statt mit SID das Name-Wert-Paar auszuwerten nutzt man **`session_id()`** und **`session_name()`**

```
<form action="b01.php" method="POST">
  …
  <input type="hidden"
    name="<?=session_name()?>" value="<?=session_id()?>"/>
</form>
```

Beenden einer Session

- Automatisch durch Schließen des Browsers
- Automatisch nach einer bestimmten Zeitspanne
- Programmatisch mit **`session_destroy`**

```php
<?php
if (isset($_POST["logout"])) {
    session_unset();
    session_destroy();

    echo "logout";
}
?>
```

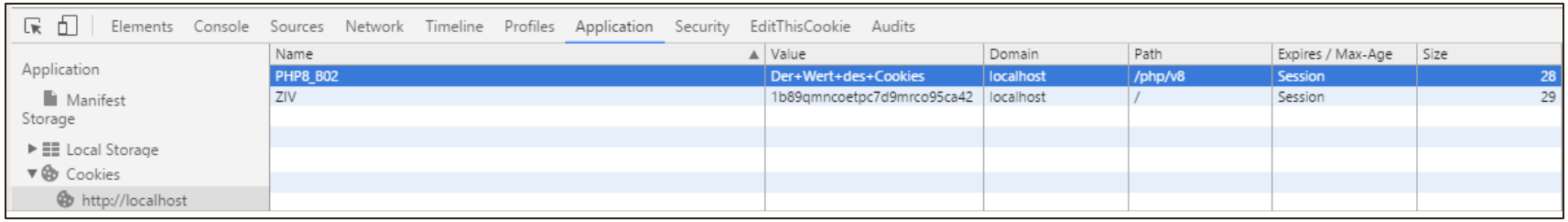
  - Mit **`session_unset`** wird der Session-Array geleert
  - Dies sollte unbedingt durchgeführt, falls nach dem Schließen noch auf `$_SESSION` zugegriffen werden könnte

Mehr zu Cookies

- Mit PHP kann man eigenständig Cookies verwalten
- Können die PHP-Session oder sogar den Server „überleben“
- Nachteile
  - Nutzer kann Cookie manipulieren
  - Nutzer kann Cookies deaktivieren

- Mit **`setcookie(`*`NAME, WERT`*`)`** kann ein Cookie gesetzt werden:

```
setcookie("PHP8_B02", "Der Wert des Cookies");
```



Mehr zu Cookies: Zeitstempel

- Beim Setzen eines Cookies kann eine Ablaufzeit angegeben werden
  - Nach Ablauf wird der Cookie ungültig und nicht mehr mitgeschickt

```
setcookie(NAME, WERT, ZEITSTEMPEL);
```

- Der Zeitstempel wird in Sekunden angegeben
- Die Funktion **`time()`** liefert die aktuelle Sekundenanzahl ab dem 1.1.1970 („Unixzeit“)
- Cookie für eine Stunde:

```
setcookie("PHP8_B02", "Der Wert des Cookies", time() + 3600);
```

- Wird keine Zeit angegeben, so wird der Cookie mit dem Schließen des Browsers ungültig

Mehr zu Cookies: Löschen, Auslesen

- Cookies können nicht explizit gelöscht werden
  - Man kann den Wert auf den leeren String setzten

```
setcookie("PHP8_B02", "");
```

  - Zudem kann man den Zeitstempel in die Vergangenheit legen
  - Der Cookie wird damit ungültig und vom Browser gelöscht

```
setcookie("PHP8_B02", "", time() - 3600);
```

- Über die super-globale Variable **`$_COOKIE`** erhält man Zugriff auf die Cookie-Werte

```
if (!isset($_COOKIE["PHP8_B02"])) {
    setcookie("PHP8_B02", "Der Wert des Cookies", time() + 3600);
}
```

Mehr zu Cookies: Mehrfache Werte

- Ein Cookie enthält immer nur einen Wert (String)
- Will man mehrere Werte auf Client-Seite ablegen gibt es zwei Wege:
  - Mehrere Cookies anlegen und verwalten
  - Mehrere Werte innerhalb des Cookie-Strings ablegen
- Für letzteres gibt es diverse Strategien:
  - alle Werte hintereinander per Trennzeichen separiert auflisten (CVS)
  - Schlüssel und Werte separiert auflisten
  - Notationssprachen wie JSON verwenden
- Vorgehen:
  - Zu Anfang des Skriptes den Cookie auslesen und den String zerlegen
  - Bei Änderung der Werte, den Cookie neuzusammen bauen und setzen

Fehler an Ort und Stelle vermeiden

- Grundsätzlich sollten Fehler an dem Ort Ihres möglichen Auftretens vermieden oder behandelt werden
- Die Division durch 0 ist mathematisch nicht definiert
- Und programmatisch führt sie häufig zum Absturz
- In PHP zur Warnung: `Warning: Division by zero`
- Die Division sollte daher vermieden werden:

```
function dividiere($zaehler, $nenner) {
  if ($nenner == 0) return ;

  return $zaehler / $nenner;
}
```

Fehlerbehandlung durch Codierung von Rückgabewerten

- Wenn eine Funktion einen Fehler „bemerkt“ kann sie durch spezielle Rückgabewerte außerhalb des normalen Wertebereichs eine Fehlerbehandlung an dem Ort ermöglichen an dem sie aufgerufen wurde.

```
$test = strpos($string, "PHP");
```

- Die Methode **`strpos`** liefert **`FALSE`** wenn „**`PHP`**“ nirgends in **`$string`** vorkommt
- Der Fehler „nicht vorhanden“ wird abgefangen und als **FALSE** dem Aufrufer signalisiert
- Diese kann nun entsprechend reagieren:

```
$test = strpos($irgendeinString, "PHP");
if (!$test) {
    echo "Fehler: PHP ist nicht im String $string vorhanden";
}
```

Vermeidung nicht immer möglich

```
function dividiere($zaehler, $nenner) {
  if ($nenner == 0) return ;
  return $zaehler / $nenner;
}
```

- Was ist das Resultat dieser Funktion falls **`$nenner === 0`**?
- Das „leere“ **`return`** suggeriert Abbruch der Funktion
- Wahrscheinlich wird die Funktion in einem Wertekontext aufgerufen wird:

```
$test = dividiere($zaehler, $nenner);
```

- Die Variable **`$test`** muss nach dem Aufruf einen Wert haben
- Möglich wäre **`FALSE`** als Rückgabe
- Allerdings verarbeitet PHP **`FALSE`** numerisch automatisch zu **`0`**
- Solche semantische Doppeldeutigkeit führt schnell zu Fehlern

Exceptions „werfen“

- PHP bietet mit *Exceptions* eine Möglichkeit Fehler von Rückgabe zu trennen, falls eine sinnvolle Fehlerbehandlung nicht an Ort und Stelle möglich ist
- Dies bewirkt eine Trennung von Fehlerauftreten und –behandlung

```
function dividiereMitException($zaehler, $nenner) {
    if ($nenner == 0) {
        throw new Exception("Division durch null");
    }

  return $zaehler / $nenner;
}
```

- Mit dem Schlüsselwort **`new`** wird eine **`Exception`** erzeugt
- Mit dem Schlüsselwort **`throw`** wird eine **`Exception`** „geworfen“

Exceptions „werfen“

- Der Aufrufer hat die Möglichkeit eine Exception zu „fangen“:

```
try {
    $test = dividiereMitException($zaehler, $nenner);
    ...
}
catch (Exception $e) {
    echo "Es ist ein Fehler aufgetreten: " . $e;
}
```

- Mit **`try-catch`** werden Anweisungen eingeschlossen, die potentiell eine Exception werfen könnten
- Wird eine Exception geworfen, so wird die Ausführung an der Stelle des Wurfs abgebrochen und mit dem **`catch`**-Teil weitergemacht
- Hier muss der Programmierer dafür sorgen, dass sein Programm in einen Zustand versetzt wird, der den Fehler „akzeptiert“

Exceptions „werfen“

- Wird die **`try-catch`**-Anweisung weggelassen, so wird die aktuelle Methode abgebrochen
- Die Fehlerverarbeitung wird damit an den vorherigen Aufrufer delegiert
- Findet bis zum „Hauptskript“ keine Verarbeitung statt, bricht das gesamte Programm mit einem **`fatal error`** ab

```
function tueWas() {
  echo "Ich teile gerne durch 0";
  dividiereMitException(100, 0);
}
```

```
function dividiereMitException () {
    …

}
```

```
try {
  tueWas();
}
catch (Exception $e) {echo $e;}
```

Exceptions „werfen“

- Falls eine Behandlung nicht möglich ist, aber dennoch bestimmte Anweisungen ausgeführt werden sollen, hilft **`finally`**

```
try {
 …
}
finally {
    …
}
```

```
try {
 …
}
catch {
  …
}
finally {
    …
}
```

- Der **`finally`**-Teil wird in jedem Fall ausgeführt
  - Egal ob ein Fehler auftrat, dieser behandelt und weitergereicht wurde!

Zusammenfassung

- Wann wird ein Session gestartet, fortgesetzt und beendet?
- Was ist eine Session-ID und wozu dient diese?
- Wie werden die Sessiondaten verwaltet und wie können diese angelegt, geändert oder gelöscht werden?
- Was muss man alles beachten, wenn man eine Session für eine Webanwendung mit mehreren Seiten verwenden will?
- Wozu werden Cookies verwendet
- Wie kann man einen Cookie in PHP definieren, auslesen und „löschen“
- Sie kennen die Funktionsweise eines `try-catch-finally`-Blocks und der Schlüsselwortes `throw`?

Aufgaben

1. Erweitern Sie das Beispiel 1 dieser Vorlesung, um eine weitere Eingabeseite für den Fall, dass der Nutzer als Zahlungsart "Kreditkarte" oder "Einzugsermächtigung" gewählt hat; unterscheiden Sie zwischen beiden Arten in Ihrem Eingabeformular.
2. Erweitern Sie das Beispiel 1 dieser Vorlesung, um die Möglichkeit auf eine beliebige Seite des Bestellvorganges (Name, Adresse, Zahlungsart) zu springen. Auf der ausgewählten Seite sollen die bisher getätigten Eingaben vorausgefüllt sein.

Anhang: ZIVinteraktiv-Abstimmung

- QR-Code zur anonymen Abstimmung:

- https://www.uni-muenster.de/ZIV/anw/zivinteraktiv/abstimmung.php?code=5804a66a