PHP
Session-Handling und Weiterleitung

Session-Handling: Start einer Session

- In PHP können Sessions mit dem Funktionsaufruf **`session_start()`** gestartet werden.
- Der Server erzeugt dabei eine ID, die als eindeutiger Schlüssel dient.
  - Beispiel für eine Session-ID: 396o4mfqmjr9okbl7vsd3g8tci
- Der Browser des Clients speichert diese Session-ID in der Regel als Cookie ab.
  - Bei jeder weiteren Anfrage an den gleichen Webserver wird diese Session-ID mitgesendet.
  - Damit kann der Webserver den Client „wiedererkennen“.
    - Stichwort: Session-Hijacking
  - Abhängig von den Browsereinstellungen nutzt der Browser diese Session-ID auch bei Anfragen über einen neuen Tab.
- Der Webserver erzeugt im Verzeichnis /var/lib/php/sessions/ für jede neue Session eine neue Datei.

Session-Handling: Start einer Session

- Der Speicherort kann je nach verwendetem Webserver unterschiedlich sein.
- Die Dateinamen entsprechen der Session-ID, allerdings mit vorangestelltem sess_:
  - Bei unserem Beispiel: sess_396o4mfqmjr9okbl7vsd3g8tci
- Alle Variablen, die innerhalb der Session verwendet werden, werden in dieser Datei gespeichert:
  - prof|s:13:"Frank Dopatka";vorlesung|s:3:"WEB";year|i:2019;
  - Also die Variablen prof, vorlesung und year
- Alle Session-Variablen werden im Klartext in dieser Datei gespeichert.
- Sensible Daten sollten also niemals in der Session gespeichert werden, da bei fehlerhaft konfigurierten Webservern womöglich auf das Dateisystem zugegriffen werden kann.
- In den neueren PHP-Versionen immer noch so?

Session-Handling: Wichtige Informationen

- Analog zu den **`$_POST`** bzw. **`$_GET`** Arrays gibt es ein **`$_SESSION`**-Array.
- Darin können beliebig Daten gespeichert werden.
- Um auf dieses Array zugreifen zu können, muss davor zwingend **`session_start()`** aufgerufen werden.
- Damit ist es möglich, seitenübergreifend auf Informationen eines Clients zugreifen zu können.
- Beispielsweise wird auf einer Seite das Alter des Clients abgefragt und anschließend in dem **`$_SESSION`**-Array gespeichert.
- Nun kann auf jeder anderen Seite problemlos auf das Alter des Client zugegriffen werden, indem die Information aus dem **`$_SESSION`**-Array geholt wird.

Session-Handling: Einführungsbeispiel

- Dieser Code demonstriert die Funktionsweise:

```php
<?php

session_start();

$_SESSION["prof"] = "Frank Dopatka";
$_SESSION["vorlesung"] = "WEB";
$_SESSION["year"] = 2019;

echo "Ihre Session-ID: " . session_id();

?>
```

- Laden Sie die Seite anschließend neu.
- Löschen sie danach Ihre Cookies und laden Sie die Seite erneut.

Weiterleitungen mit PHP: Übersicht

- Weiterleitung in PHP funktionieren grundsätzlich über den HTTP-Header und dem header-String „Location“.
  - Beispiel: `header("Location: login.php");`
- Wenn Sie bereits HTML-Code an den Client gesendet haben, beispielsweise nur `<!DOCTYPE html>`, dann können Sie den HTTP-Header nicht mehr modifizieren, da er bereits zum Client gesendet wurde!
- Alle Prüfungen müssen also erfolgen, bevor die erste Ausgabe an den Client erfolgt.
- HTTP ist zustandslos, will man also geschützte Bereiche auf der Webseite haben, muss bei jeder einzelnen Seite geprüft werden, ob der Besucher auch autorisiert ist, auf diesen Bereich zuzugreifen.

Session-Handling: Erweiterung des Beispiels

- Bisher werten wir nur ein HTML-Formular aus und protokollieren alle Anfragen auf das Auswertungsskript.
- Um ein vollständiges Login-Skript zu realisieren, muss bei korrekten Eingabedaten eine Session mit allen relevanten Informationen gestartet werden.
- Anschließend muss bei jeder zu schützenden Seite geprüft werden, ob der Anfragesteller auch die Berechtigung für den Zugriff hat.
  - Falls nicht, soll der Client zum Login geleitet werden.
  - Ansonsten soll die normale Seite angezeigt werden.
- Weiterhin muss es eine Möglichkeit geben, dass der Client die Session beenden kann.

Session-Handling: Erweiterung des Beispiels

- Die Datei auswertung.php kann nun angepasst werden:

```php
<?php

// Prüfe, ob die Daten stimmen
if(!($username === "Mustermann" && $password === "web-hs-ma")) {
    exit("Benutzername oder Passwort ist nicht korrekt!");
} else {

    session_start();
    $_SESSION["user"] = $username;
    $_SESSION["auth"] = 1;

    header("Location: secure.php");
    exit();

}

?>
```

Session-Handling: Erweiterung des Beispiels

- Die Datei secure.php muss erstellt werden:

```php
<?php

session_start();

// Prüfung, ob Nutzer berechtigt ist
if(!(isset($_SESSION["user"]) && $_SESSION["auth"] == 1)) {
    header("Location: formular.html");
    exit();
}

?>
```

Session-Handling: Erweiterung des Beispiels

- Nach der erneuten Zugangsprüfung kann der Inhalt angezeigt werden:

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Sicherheitsseite</title>
</head>
<body>
    <h1>Diese Seite ist sicher!</h1>
    <h4>Willkommen, <?php echo $_SESSION["user"]; ?>.</h4>
</body>
</html>
```

- Die Zugangsprüfung muss vor jeder einzelnen geschützten Seite erfolgen!

Session-Handling: Erweiterung des Beispiels

- Die Datei logout.php kann nun auch erstellt werden:

```php
<?php

// Zuerst Session starten!
session_start();

// Session beenden
session_unset();
session_destroy();

// Wieder zum Login weiterleiten
header("location: formular.html");
exit();

?>
```

- Damit sind alle grundlegenden Funktionen eines Login-Systems implementiert. Nutzen Sie dieses System NICHT für reale Projekte!

Session-Funktionen

Funktion	Bedeutung
session_start()	startet eine neue Session oder übernimmt eine bereits vorhandene
$erg=session_id()	gibt den eindeutigen Identifier der eigenen, aktuellen Session zurück
$erg=session_encode()	liefert eine Liste mit allen abgespeicherten Variablen der aktuellen Session
session_unregister("name")	entfernt eine Variable mit dem Namen "name" aus der Liste der Variablen einer Session
session_unset()	löscht den Inhalt aller Variablen, die in der aktuellen Session deklariert wurden; die Session selbst bleibt jedoch bestehen
session_destroy()	beendet eine Session und löscht alle Daten, die in der Session verwendet wurden; die Session-ID wird wieder freigegeben