Sessions und Login

1)Sessions in PHP (Kurzzeitgedächtnis in einer PHP-Sitzung)

HTTP ist ein zustandsloses Protokoll, das bedeutet, Informationen werden zwischen den verschiedenen Aufrufen eines Besuchers nicht zwischengespeichert. Dies ist unpraktisch, wenn wir gewisse Informationen zu einem Besucher speichern müssen, beispielsweise mit welchem Benutzernamen sich dieser eingeloggt hat. Um dies zu lösen, verwendet man in PHP-Sessions. Über die Sessions haben wir eine einfache Möglichkeit, uns Informationen, Daten und Zustände während einer kompletten Nutzungsdauer eines Besuchers zu merken.

Mit der Funktion wird eine Sitzung gestartet **session_start()**

PHP-Sessions, man kann sie auch „**Sitzungsvariablen**“ nennen, werden mit der globalen PHP-Variablen $_SESSION gesetzt.

Sessions können Benutzerinformationen speichern, die auf mehreren Seiten verwendet werden sollen (z. B. Benutzername, Rechnungsnummer usw.). Sitzungsvariablen (Sessions) enthalten Informationen über einen einzelnen Benutzer und sind für alle Seiten in einer Anwendung verfügbar. Standardmäßig bleiben Sitzungsvariablen bestehen, bis der Benutzer den Browser schließt.

Auf dem Server ist dann für jede Session-ID lokal ein Speicher eingerichtet, der beliebige Variablen für den Besucher beinhalten kann. So können wir in der Session für einen Besucher beispielsweise abspeichern, mit welchem Benutzernamen sich dieser eingeloggt hat, welche Waren in seinem Warenkorb liegen usw.

Der Benutzer hat **keine** Möglichkeiten, die Variablen in seiner Session zu sehen oder zu manipulieren. Er besitzt nur die Information, welche Session ID ihm vom Webserver zugewiesen wurde.

Lebenslauf einer Session-Variablen
Die Session-Variable wird automatisch zerstört, wenn der Browser geschlossen wird oder eine gewisse Zeit vergangen ist (meistens 180 Minuten). Es gibt auch die Möglichkeit, dass Sessions in Cookies gespeichert werden und dadurch über Tage vorgehalten werden können (mit Sicherheitsproblemen – ein anderer nutzt die Webanwendung weiter, weil sich der eigentliche Nutzer nicht ausgeloggt hat).

Man kann sich ausgeben lassen, wie lange eine Session-Zeit eingestellt ist:
echo session_cache_limiter() bzw. eine eigene Zeit setzen, nach dem die Session aus Mangel an Bewegung verfällt (ohne Aktivität war). Über den Befehl session_cache_limiter(20) wird die Lebensdauer der Session auf 20 Minuten gesetzt.

Session registrieren

Erstelle eine neue Datei „session.php“

Möchte man in einem Script auf Sessions zurückgreifen, muss man, bevor man irgendeine Ausgabe macht, den Befehl *session_start();* aufrufen:

```php
<?php
session_start();
?>
```

Es empfiehlt sich, diesen Code immer ganz oben der Scripts stehen zu haben.

Einen Wert / eine Variable über mehrere Seitenaufrufe hinweg in der Session speichern.
Zum Beispiel eine Zahl und einen String. Man kann dabei die Session-Variable wie jede andere Variable in PHP verwenden. Man kann darin Zahlen, Zeichenketten oder sogar Arrays abspeichern.

Beachte: in einer Session müssen die Anführungszeichen immer dieselben sein, also entweder

```php
$_SESSION['kunde'] = 'Huber';
```

oder

```php
$_SESSION["kunde"] = "Huber";
```

aber auf keinen Fall:

```php
$_SESSION['kunde'] = "Huber";
```

Übung:
Erstelle eine Datei „session.php“:
Damit sollen 2 Sessions erstellt werden.

Zur besseren Fehlerverfolgung sollen diese umfangreich angezeigt werden. Nutze dafür in jeder PHP-Datei am Beginn:

<?php
error_reporting(-1);
ini_set('display_errors','On');

session.php | session2.php | formular2.php

session.php

```php
<?php
error_reporting(-1);
ini_set('display_errors','On');

session_start();

$_SESSION['u_id'] = 4;
$_SESSION['email'] = 'franz@nawi.at';

header('Location: session2.php');

?>
```

Erstelle danach die zweite Datei mit dem Namen „session2.php“.
Dort sollen diesen Wert ausgegeben werden.

Damit die SESSION auch gesetzt und somit im Hintergrund laufen kann, muss man diese Datei mit der Session auch in jeder Datei aufrufen.

Wichtig, immer wenn man irgendwo mit Sessions arbeitet, muss zuvor *session_start()* ausgeführt worden sein.

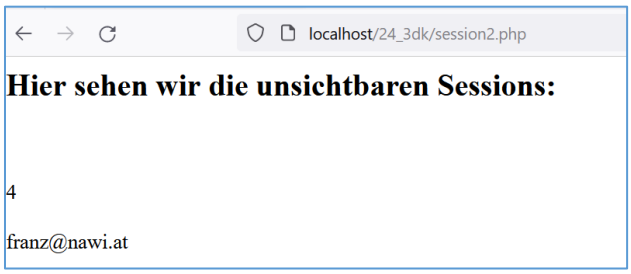
session.php | session2.php | formular2.php

session2.php

```php
<?php
error_reporting(-1);
ini_set('display_errors','On');

session_start();

echo "<h2>Hier sehen wir die unsichtbaren Sessions: </h2><br><br>";

echo $_SESSION['u_id'];
echo "<br><br>";
echo $_SESSION['email'];

session_destroy();

?>
```

Starte den Browser und XAMPP und gib die URL der „session.php“ ein, die dann auf die „session2.php“ weiterleitet.

Ergebnis:



Funktioniert.
Lösche diese Zeilen in „login.php“ wieder. Es hat nur zur Vorführung gedient.

2)Beispiel mit Login - unser Login verbessern

Ziel: Unser Login von vorher soll nun verbessert werden, indem bei einem erfolgreichen Login eine Session erzeugt wird, die dann später noch verwendet werden kann. Das kann sinnvoll sein, um Zugänge zu ermöglichen bzw. in der Navbar ein „eingeloggt als:“ zu ermöglichen.

2a)Session setzen

Arbeite in der „login.php“ weiter.

Füge in der IF ein:

```
session_start();
$_SESSION['email'] = $email;
```

```php
if($email == "andi@hak.it" && $pwd == "1234")
{
    // erfolgreich eingeloggt
    session_start();
    $_SESSION['email'] = $email;

    header("Location: ../index.php");
    exit;
} else
{
    echo "Zugang verweigert.";
}
```

Nun hat man eine neue Session erstellt und kann diese z.B. in der Zieldatei verwenden.

Zusätzlich könnte man auch noch ein zweite session erstellen:

```php
    session_start();
    $_SESSION['email'] = $email;
    $_SESSION['eingeloggt'] = true;

    header("Location: ../index.php");
    exit;
```

$_SESSION['eingeloggt'] = true;

2b)Session in anderer Datei nutzen

In der index.php erstelle eine Ausgabe der nun vorhandenen E-Mail:

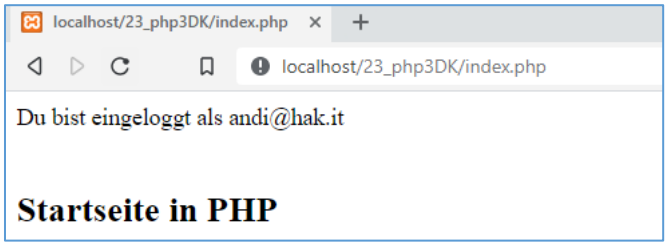
- Die session starten
- Die nun vorhandene session bezüglich „email“ mit dem Verkettungspunkt zum vorhandenen Text (String) anhängen.
- 2 x
 für einen schönen Abstand sorgen

login.php | login2.php | index.php

index.php

```php
ini_set('display_errors','On');

session_start();
echo "Du bist eingeloggt als " . $_SESSION['email'];
echo "<br><br>";

echo "<h2>Startseite in PHP</h2>";
```

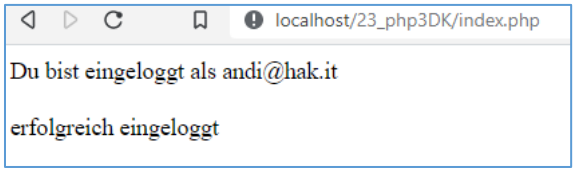
Ergebnis:



Die 2. Session kann ebenfalls genutzt werden:

```php

if($_SESSION['eingeloggt'] == true) {
    echo "erfolgreich eingeloggt";
}
echo "<br><br>";

echo "<h2>Startseite in PHP</h2>";
```

Ergebnis:



3)Session löschen

Eine Session soll nicht ewig vorhanden sein.

Session-Variablen zerstören

Es kann gezielt die Session-Variable gelöscht werden. Dies geschieht für einzelne Variablen über

- unset ($_SESSION['variablenname']); oder für eine komplette Session über
- unset ($_SESSION); oder
- session_destroy();

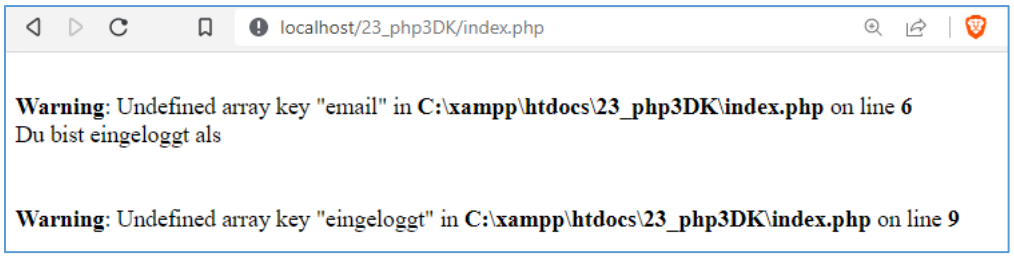
Erstelle eine neue Datei im Ordner „templates“ mit dem Namen „logout.php“.

Diese erhält nur wenig Code:


```
<?php
session_start();
session_destroy();
header("Location: ../index.php");
exit();
```

Testen:

Öffne die logout.php und betrachte danach die „index.php“. Da die Sessions nun nicht mehr vorhanden sind, sollte auch der Fehler kommen, dass diese Inhalte nicht mehr verfügbar sind:



Quelle: https://www.youtube.com/watch?v=O0Ky0tKvsJ8

https://www.php-kurs.com/session-anwenden.htm

https://www.php-einfach.de/php-tutorial/php-sessions/

https://www.w3schools.com/php/php_sessions.asp#