1)Login Funktion ohne Datenbank

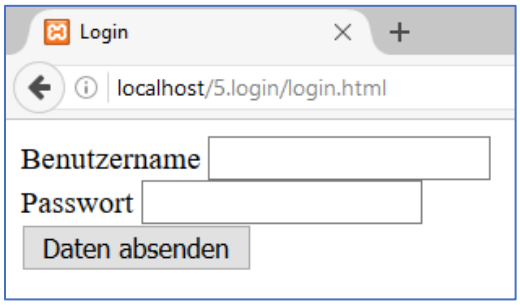
Die Seite „**privat.php**“ soll erst aufrufbar sein, wenn der Login, mit der Datei „**login.html**“ erfolgreich war. Die ist verknüpft mit der „login.php“. Die „**logout.php**“ dient zur ordentlichen Abmeldung.

Öffne die „privat.php“:

Info.:

- Mit „Zur Anmeldung“ kommt man zur Login Möglichkeit.

login.html



1)login.html

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Login</title>
</head>

<body>
<form action="login.php" method="post">
    <label for="benutzername">Benutzername</label>
    <input id="benutzername" name="name" type="text"><br>
    <label for="passwort">Passwort</label>
    <input id="passwort" name="passwort" type="password"><br>
    <input type="submit" name="submit">
</form>
</body>
</html>
```

2)login.php

In den Zeilen 4 bis 7 wird geschaut, ob überhaupt Daten in die <input>-Felder des Formulars eingegeben wurden. Es würde eine Fehlermeldung mit „echo“ erscheinen.

if ($_POST['name'] == 'max.muster' && $_POST['passwort'] == 'geheim')

Prüfen, ob der Benutzername und das Passwort korrekt sind. Hier mit einem festen Begriff, besser wäre ein Zugriff auf eine Datenbank.

Sind beide korrekt, soll der Benutzer in eine „Session“-Variable gespeichert werden. Daher wird die Session in Zeile 2 gestartet:

session_start();

In der Zeile 11 kann man nun, da sie ja gestartet wurde, darauf zugreifen:

$_SESSION['benutzername'] = 'max.muster';

Der Name müsste aber nicht gleich sein wie oben, könnte auch ganz anders heißen, wie z.B. „benutzer_id“ und „123“.

Als zweites muss aber noch folgendes passieren, nämlich wenn die Benutzereingabe erfolgreich war: die Weiterleitung auf das Dokument, das ja geschützt sein soll. Diese Weiterleitung passiert, indem man einen „Location-header“ aufruft. Ein „header“ ist ähnlich einem Cookie eine Steuerinformation für den Browser.

header('Location: privat.php');

Nach dieser „Umleitung“ muss aber das Skript beendet werden, nämlich mit „exit“:

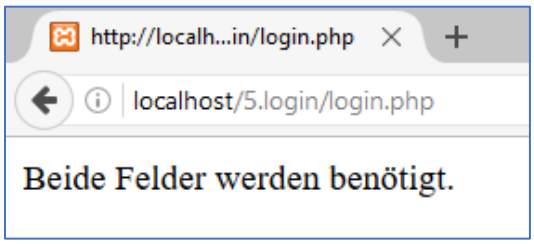
exit;

Nun folgt die Behandlung des Falls, dass die Benutzereingabe falsch ist, nämlich im „else“-Bereich:

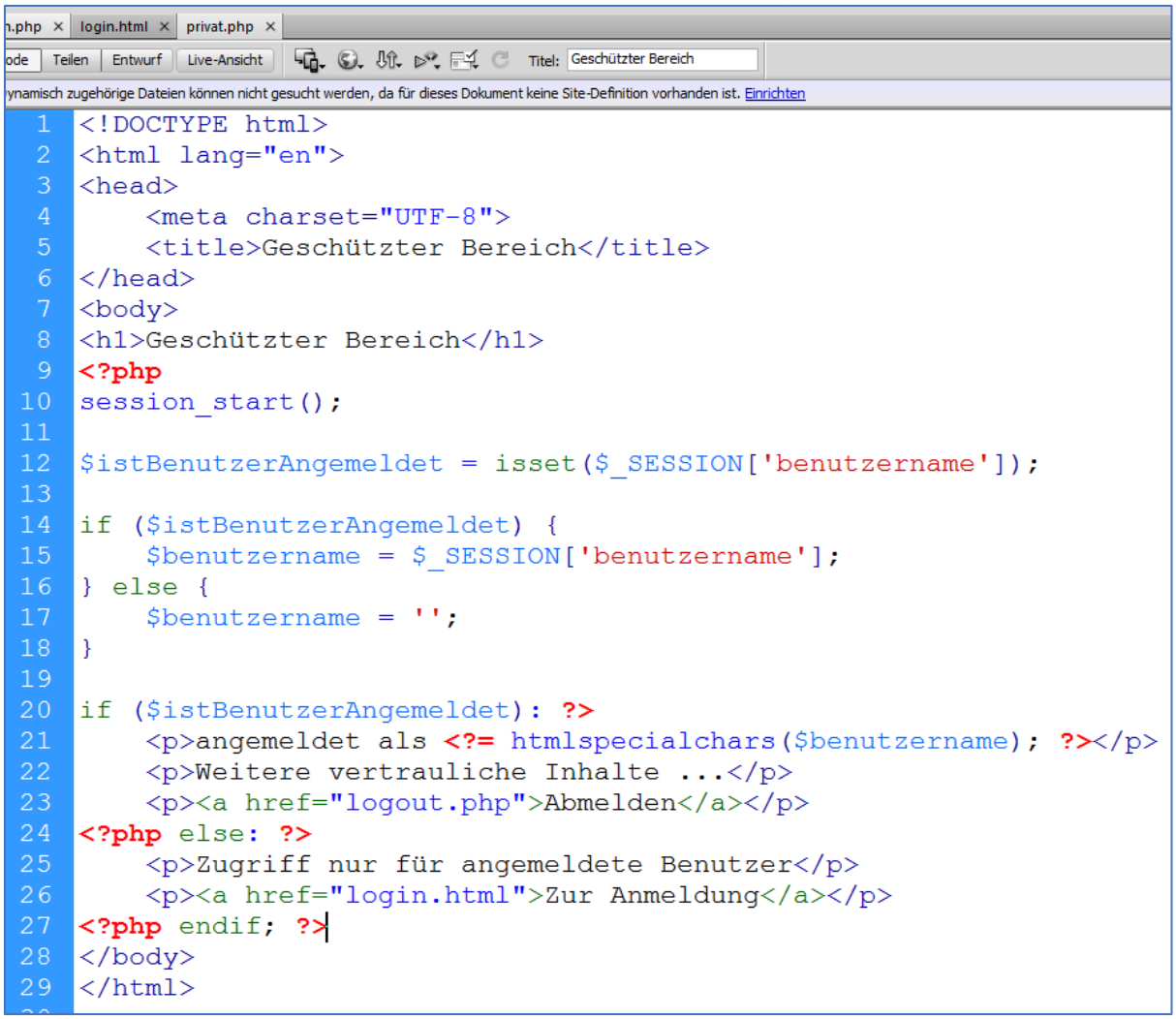
else { echo "falscher Name oder Passwort"; }

Die Benutzung der Variablen erfolgt in der Zeile 26 mittels:

Ergebnis, wenn keine korrekte Eingabe erfolgte:



3)privat.php



Nun muss in der „privat.php“ geprüft werden, ob der Benutzer angemeldet ist. Dazu muss man das Element „benutzername“ in der Session-Variablen betrachten: ist der „benutzername“ vorhanden, kann man davon ausgehen, dass der Login geklappt hat.
Der gesamte Vorgang, inkl. „isset“ wird in einer Variablen „istBenutzerAngemeldet“ gespeichert

$istBenutzerAngemeldet = isset($_SESSION['benutzername']);

Vorher muss die Session gestartet sein, hier in Zeile 10:

session_start();

Die Anwendung der „erfolgreichen Anmeldung“ wird in der Zeile 20 verwendet:

<?php if ($istBenutzerAngemeldet): ?>

Damit, wie in Zeile 21 erkennbar, auch direkt der Name des Benutzers ausgegeben wird, wird zuerst in Zeile 12 geprüft, ob der Benutzer angemeldet ist:

```
if ($istBenutzerAngemeldet) {
   $benutzername = $_SESSION['benutzername'];
} else {
   $benutzername = '';
}
```

Darin erhält die Variabel „$benutzername“ die Information der Session, in der der Benutzername gespeichert ist. Ist kein Benutzer angemeldet, soll die Variable mit einem leeren String belegt werden.

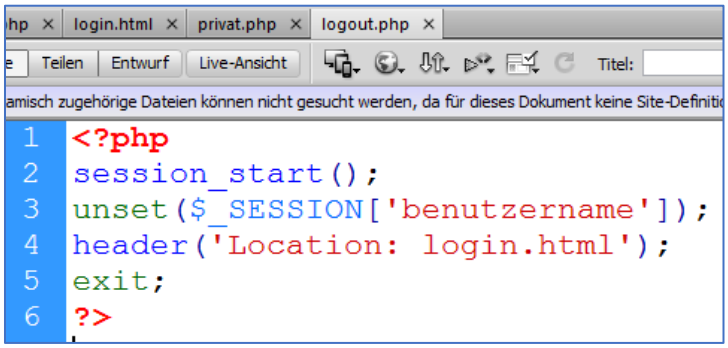
Ausgegeben wird dies in der Zeile 21 in einem kurzen php-Tag:

```
<p>angemeldet als <?= htmlspecialchars($benutzername); ?></p>
```

3)Abmeldung durch „logout“

Die Logout-Datei ist eigentlich leer und hat nur zwei Aufgaben, nämlich

1. den Benutzernamen aus der Session zu entfernen und
2. den Benutzer zum Login umzuleiten.



In Zeile 5 geht man davon aus, dass in der Session-Variablen der Benutzername gesetzt war.

```
unset($_SESSION['benutzername']);
```

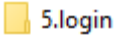
„unset“ löscht das Element „benutzername“. Wenn es nicht gesetzt war, weil ein Besucher „logout“ aufgerufen hat, ohne vorher angemeldet gewesen zu sein, ist das kein Problem.

```
header('Location: login.php');
exit;
```

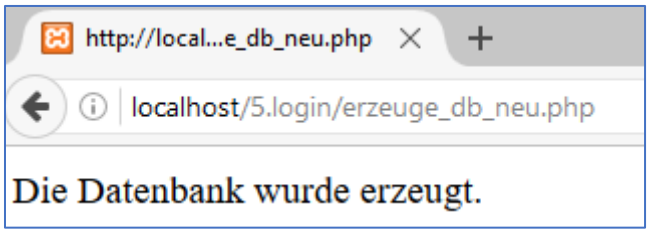
„header“ vollzieht eine Umleitung auf den „login“

Quelle: PHP7 und MySQL; Rheinwerk-Video, 2016, Kapitel 5.3

2)Login mit Datenbankzugriff

a)Erzeuge mit diesem vorgefertigtem PHP-Skript eine neue Datenbank. Man muss nur diese Datei „erzeuge_db_neu.php“ im Xampp-Ordner positionieren, hier in „5.login“ 

und mit dem Browser aufrufen. Die Datenbank wird mit den SQL-Befehlen selbständig erzeugt.



Ohne dem „echo“ am Ende würde sogar keine Info ausgegeben. Die Datenbank wäre aber trotzdem erzeugt worden.

erzeuge_db_neu.php

```php
<!DOCTYPE html><html><head><meta charset="utf-8"></head><body>
<?php
   $con = mysqli_connect("localhost", "root", "");
   $sql = "CREATE DATABASE IF NOT EXISTS login";
   mysqli_query($con, $sql);
   mysqli_select_db($con, "login");

   $sql = "CREATE TABLE IF NOT EXISTS kunde (
      ku_id int(11) DEFAULT NULL,
      name varchar(30) DEFAULT NULL,
      passwort varchar(30) DEFAULT NULL,
      PRIMARY KEY ku_id (ku_id)
      ) ENGINE=MyISAM";
   mysqli_query($con, $sql);

   $sql = "INSERT INTO kunde(ku_id, name, passwort) VALUES
      (1, 'Darrer', 'hamburg'),
      (2, 'Kraml', 'geheim'),
      (3, 'Huber', 'leo99')";
   mysqli_query($con, $sql);
   mysqli_close($con);

   echo "Die Datenbank wurde erzeugt.";
?>
</body></html>
```

Ergebnis:

+ Optionen

	ku_id	name	passwort
Bearbeiten Kopieren Löschen	1	Darrer	hamburg
Bearbeiten Kopieren Löschen	2	Kraml	geheim
Bearbeiten Kopieren Löschen	3	Huber	leo99

b)HTML Code für das Formular „db_login.html“

```
ogin.php × db_login.html ×
ode | Teilen | Entwurf | Live-Ansicht | Titel: LOgin mit Datenbank
 1  <!doctype html>
 2  <html>
 3  <head>
 4  <meta charset="utf-8">
 5  <title>LOgin mit Datenbank</title>
 6  </head>
 7
 8  <body>
 9  <form action="db_login.php" method="post">
10      <label for="benutzername">Benutzername</label>
11      <input id="benutzername" name="name" type="text"><br>
12      <label for="passwort">Passwort</label>
13      <input id="passwort" name="passwort" type="password"><br>
14      <input type="submit" name="submit">
15  </form>
16  </body>
17  </html>
```

Auch den „Submit“-Button mit einem „name“ versehen. Dieser wird später benötigt. Nämlich in der Datei „db_login.php“, wo in der Zeile 13 mit „isset“ eine Überprüfung stattfindet.

c)PHP-Code:

Verwende die Datei „login.php“ weiter und speichere sie als „db_login.php“

Hier wird die Verbindung zur Datenbank „login“ hergestellt.

```
db_login.php ×
Code | Teilen | Entwurf | Live-Ansicht | Titel:
Dynamisch zugehörige Dateien können nicht gesucht werden, da für dieses Dokument keine Site-Definition vorhanden ist. Einrichten
 2  session_start();
 3
 4  /* Verbindung aufnehmen*/
 5  $con = new MySQLi("localhost", "root", "", "login");
 6  if ($con->connect_error) {
 7      echo "Fehler bei der Verbindung: " . mysqli_connect_error();
 8      exit();
 9  }if (!$con->set_charset("utf8"))  {
10      echo "Fehler beim Laden von UTF8" . $con->error;}
```

```
12 $istEingabeFehlerhaft = ""; //Variable bezüglich fehlerhafte Angaben
13 if(isset($_POST["submit"]))
14 {if(empty($_POST["name"]) || empty($_POST["passwort"]))
15 {echo "Beide Felder werden benötigt.";}
16 else
17 { $name= $_POST['name'];
18   $passwort = $_POST['passwort'];
19
20 $sql = "SELECT ku_id FROM kunde WHERE name = '$name'
21 AND passwort ='$passwort'";
22 $result=mysqli_query($con, $sql);
23
24 if(mysqli_num_rows($result) == 1)
25 {$_SESSION['benutzername'] = "$name";
26         header('Location: db_privat.php');
27     } else {
28         echo "falscher Name oder Passwort";
29     }}}
30  ?>
```

Zeile 12 bis 15 wird geschaut, ob überhaupt Daten in die <input>-Felder des Formulars eingegeben wurden. Es würde eine Fehlermeldung mit „echo“ erscheinen.

Tipp:

Um im Echtbetrieb gefährliche „Injektions“ zu vermeiden sollte zwischen den Zeilen 26 und 27 folgendes eingefügt werden: - hier Zeile 17-22: (http://www.eggslab.net/php-login-script/)

```
13 // Define $username and $password
14 $username=$_POST['username'];
15 $password=$_POST['password'];
16
17 // To protect from MySQL injection
18 $username = stripslashes($username);
19 $password = stripslashes($password);
20 $username = mysqli_real_escape_string($db, $username);
21 $password = mysqli_real_escape_string($db, $password);
22 $password = md5($password);
23
24 //Check username and password from database
25 $sql="SELECT uid FROM users WHERE username='$username' an
```

Ergebnis:

Der Zugriff auf die „db_privat.php“ funktioniert:

Die „privat.php“ wurde als „db_privat.php“ weiterverwendet und die „logout.php“ als „db_logout.php“. Nur die Verweise darinnen wurden in diesen beiden Daten geändert.