6)Webshop in PHP – User registrieren

1)neue User registrieren – mit einem Formular

Dafür gibt es schon eine Tabelle „users“ in der Datenbank.

	#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktio
☐	1	id	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	B
☐	2	username	varchar(255)	utf8mb4_general_ci		Nein	*kein(e)*			B
☐	3	password	varchar(255)	utf8mb4_general_ci		Nein	*kein(e)*			B

Nun soll ein Formular erstellt werden, in dem sich ein neuer Benutzer anmelden kann.

- **registrieren.php erstellen**

Erstelle eine neue Datei (neben „login.php“) mit dem Namen „registrieren.php“, also im Ordner „templates“. Am besten man kopiert den Inhalt der „login.php“ dort hinein und ändert einiges.

Die PHP-teile bezüglich „hasError“ kann man lassen, weil man es auch hier mit den Fehlermeldungen nutzen kann.

Ändere im Formular die beiden „values“ und den Text im Submit-Button:

```
29          <?php endif;?>
30 ▼            <div class="form-group">
31                  <label for="username">Benutzername</label>
32                  <input type="text"  name="username" class="form-control"
                    id="username">
33              </div>
34 ▼            <div class="form-group">
35                  <label for="password">Passwort</label>
36                  <input type="password"  name="password" class="form-control"
                    id="password">
37              </div>
38          </div>
39 ▼        <div class="card-footer">
40              <button class="btn btn-success" type="submit">Registrieren</button>
41          </div>
```

Auch hier ändern:

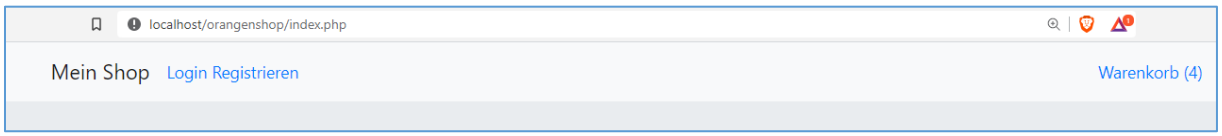
```
17      </header>
18 ▼    <section class="container" id="loginForm" >
19 ▼    <form action="index.php/registrieren" method="post">
20 ▼        <div class="card">
21          <div class="card-header">Registrieren</div>
22 ▼        <div class="card-body">
```

- **Link in der Navbar anlegen**

Öffne die „navbar.php“ und legen dort unterhalb des „Login“ einen Link „Registrieren“ an.

```
 9          <?php if(!isLoggedIn()):?>
10              <a href="index.php/login">Login</a>
11              <a href="index.php/registrieren">Registrieren</a>
12          <?php endif;?>
13      </li>
```

Ergebnis:



Route „registrieren“ anlegen

Öffne „routes.php“ und lege ganz unten eine neue Route an. Diese soll auf die „login“ Seite verbinden, damit man sich nach dem Registrieren gleich einloggen kann.

```
96
97 ▼ if(strpos($route,'/registrieren') !== false) {
98
99       require __DIR__.'/templates/login.php';
100      exit();
101  }
102  }
```

- Nun muss man die Variablen, die der Benutzer eingegeben hat definieren.

```
97 ▼ if(strpos($route,'/registrieren') !== false) {
98
99       $username = "";
100      $password = "";
101
```

- Nun folgt der Schritt für die Verarbeitung dieser. Diese soll jedoch nur dann geschehen, wenn ein POST-Request auch abgesendet wurde. Das wurde bereits in der Route vom „login“ auch schon so gehandhabt. Daher kann man es von oben kopieren.

```
99       $username = "";
100      $password = "";
101      $isPost = strtoupper($_SERVER['REQUEST_METHOD']) ==='POST';
```

- Darunter nun die IF-Abfrage, wenn die POST versandt wurde, lese die Daten aus dem Formular aus. Auch das wurde bereits beim Login oben verwendet. Hier muss man das nun auch für beide Variablen erstellen:

```
101      $isPost = strtoupper($_SERVER['REQUEST_METHOD']) ==='POST';
102 ▼    if($isPost){
103          $username =
             filter_input(INPUT_POST,'username',FILTER_SANITIZE_SPECIAL_CHARS);
104          $password =
             filter_input(INPUT_POST,'password',FILTER_SANITIZE_SPECIAL_CHARS);
105      }
```

```
if($isPost){
    $username = filter_input(INPUT_POST,'username',FILTER_SANITIZE_SPECIAL_CHARS);
    $password = filter_input(INPUT_POST,'password');
  }
```

- Um auch **Fehlermeldungen** ausgeben zu können, muss man auch diese Variable wieder erstellen und diese in ein Array ausgeben lassen:

```
101         $isPost = strtoupper($_SERVER['RE
102         $errors = [];
103         if($isPost){
```

Um zu ermitteln, ob Fehler überhaupt vorliegen, braucht man die Variable „hasErrors“. Diese zählt die Anzahl der Fehlermeldungen und diese müssen mehr als Null sein.

```
105             $password =
                filter_input(INPUT_POST,'password',FILTER_SANIT
106         }
107         $hasErrors = count($errors) > 0;
108
109         require __DIR__.'/templates/registrieren.php';
110         exit();
```

- Als wichtige Sicherheitsmaßnahme sollte man nun nachprüfen, ob die Inputfelder leer sind oder nicht. Folgende Überprüfungen sind zu erstellen: also wenn das Feld leer ist wird eine Fehlermeldung angezeigt.

```
                filter_input(INPUT_POST,'password',FILTER_SA
106
107         if(!username){
108             $errors[]="Bitte Username eintragen";
109         }
110         if(!password){
111             $errors[]="Bitte Passwort eintragen";
112         }
113         $hasErrors = count($errors) > 0;
```

```
if(!username){
    $errors[]="Bitte Username eintragen";
  }
  if(!password){
    $errors[]="Bitte Passwort eintragen";
  }
```

- **Sicherheit: Leerzeichen verhindern**
  Zur weiteren Sicherheit sollte man Leerzeichen nicht zulassen. Daher verwende die Funktion „trim“ die die Variable aufnimmt, vorne und hinten die Leerzeichen entfernt und wieder ausgibt. Werden nur Leerzeichen eingegeben, werde alle entfernt.

```
103         if($isPost){
104             $username =
                filter_input(INPUT_POST,'username',FILTER_SANITIZE_SPE
105             $username = trim($username);
106             $password =
                filter_input(INPUT_POST,'password',FILTER_SANITIZE_SPE
107             $password = trim($password);
108
```

```
$username = trim($username);
```

$password = trim($password);

- Nun folgt die **<u>Verbindung zu einer neuen Funktion</u>**, die später in „user.php“ angelegt wird.

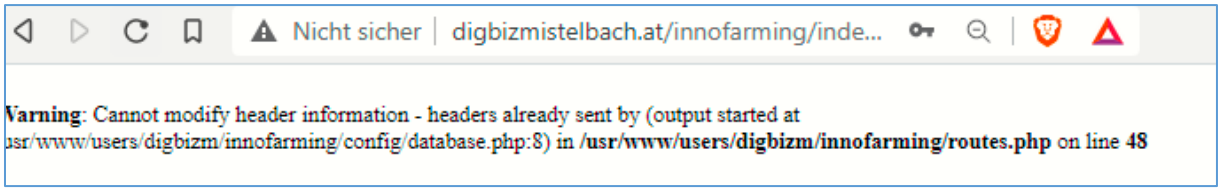
```
114             $errors[]="Bitte Passwort eintragen";
115         }
116         if(count($errors) === 0){
117             registrierenUser($username,$password);
118             header("Location: /orangenshop/index.php/login");
119             exit();
120         }
121 
122         }
123         $hasErrors = count($errors) > 0;
```

```
if(count($errors) === 0){
    registrierenUser($username,$password);
    header("Location:  /shop4ck2/index.php/login");
    exit();
  }
```

Wenn es keine Fehler gibt, soll die Funktion ausgeführt werden.

Danach soll auf die Login verzweigt werden, damit man den User gleich nutzen kann.

<u>BEACHTE:</u>
In „localhost“ ist folgendes KEIN Problem. Aber z.B. auf einem Domian-Server wie www.hetzner.de kommt es zu einer Fehlermeldung bezüglich dieser Weiterleitung in Zeile 118. Es wird durch die „session_start“ der „header“ dann nochmals beansprucht, was aber nicht möglich ist. Folgende Fehlermeldung KANN kommen:



Daher sollte man die Zeile mit der „header("Location: ".$zielEingeloggt);“ ersetzen durch folgenden Code, der ebenfalls weiterleitet

echo("<script>location.href = '/innofarming/indexshop.php/login';</script>");

```
  if(count($errors) === 0){
      registrierenUser($username,$password);
      //header("Location: /innofarming/indexshop.php/login");
      echo("<script>location.href = '/innofarming/indexshop.php/login';</script>");
      exit();
  }
```

Lösung gefunden bei:

https://stackoverflow.com/questions/8028957/how-to-fix-headers-already-sent-error-in-php

2)neue Funktion für das Schreiben in die Datenbank:

Erstelle dann im „user.php“ eine neue Funktion, die die Daten in die DB schreibt.

- Die Funktion hat den Namen „registrierenUser“ und es werden die beiden Parameter übergeben, die aus dem Formular kommen.
- Da die Variable „$password“ bereits als Parameter vorhanden, wird das Passwort sofort in der nächsten Zeile „gehashed“ – dazu verwendet man die Funktion „password_hash“ und „DEFAULT“.
- Darunter erfolgt der SQL-Befehl. Hier wird aus Sicherheitsgründen wiederum die Variante mit prepaired statement verwendet.

```
40 ▼ function registrierenUser(string $username,string $password){
41
42       $password = password_hash($password,PASSWORD_DEFAULT);
43
44       $sql ="INSERT INTO users SET
45               username= :username,
46               password= :password";
47
48       $statement = getDB()->prepare($sql);
49       //überprüfen ob hier ein Fehler ist
50 ▼     if(!$statement){
51           return 0;
52       }
53       //Ausführen des Statements und Übergabe der Platzhalter
54 ▼     $statement->execute([
55           ':username'=>$username,
56           ':password'=>$password
57       ]);
58       }
```

function registrierenUser(string $username,string $password){

$password = password_hash($password,PASSWORD_DEFAULT);

$sql ="INSERT INTO users SET
username= :username,
password= :password";

$statement = getDB()->prepare($sql);
//überprüfen ob hier ein Fehler ist
if(!$statement){
return 0;
}
//Ausführen des Statements und Übergabe der Platzhalter
$statement->execute([
':username'=>$username,
':password'=>$password
]);
}

Testen:

Nach dem Login kann man mit Logout wieder verlassen

Nur Info:

Natürlich müssen die Argumente in der Funktion aus der „routes.php“ Zeile 98 genauso viele sein, wie auch in der Funktion in der „users.php“ Zeile

```
97      if(count($errors) === 0){
98          registrierenUser($username,$password,$vorname);
99          //header("Location: /innofarming/indexshop.php/login"
```

und

```
34
35  function registrierenUser(string $username,string $password,string $vorname){
36
```

Ansonsten bekommt man auch die passende Fehlermeldung:

atal error: Uncaught ArgumentCountError: Too few arguments to function registrierenUser(), 2 passed in C:\xampp\htdocs\innofarming\routes.php on line 98 and exactly 3 expected in C:\xampp\htdocs\innofarming\function\users.php:35 Stack trace: #0 C:\xampp\htdocs\innofarming\routes.php(98): registrierenUser('ww', 'ww') #1 C:\xampp\htdocs\innofarming\includes.p C:\xampp\htdocs\innofarming\indexshop.php(7): require('C:\\xampp\\htdocs...') #3 {main} thrown in **C:\xampp\htdocs\innofarming\function\users.php** on line **35**

Es müssen auch die Variablen in der if($isPost) geschrieben werden, auch die Postleitzahl

```
85      if($isPost){
86          $username = filter_input(INPUT_POST,'username',FILTER_SANITIZE_SPECIAL_CHARS);
87          $username = trim($username);
88          $password = filter_input(INPUT_POST,'password',FILTER_SANITIZE_SPECIAL_CHARS);
89          $password = trim($password);
90          $vorname = filter_input(INPUT_POST,'vorname',FILTER_SANITIZE_SPECIAL_CHARS);
91          $vorname = trim($vorname);
92          $nachname = filter_input(INPUT_POST,'nachname',FILTER_SANITIZE_SPECIAL_CHARS);
93          $nachname = trim($nachname);
94          $plz = filter_input(INPUT_POST,'plz',FILTER_SANITIZE_SPECIAL_CHARS);
95          $plz = trim($plz);
96
```

PLZ:

Im Formular ist es ein type=‘number‘ und in der Datenbank ein interger

```
</div>
<div class="form-group">
    <label for="plz">Postleitzahl</label>
    <input type="number" name="plz" class="form-control" value="<?= $plz ?>">
</div>
```

5	Nachname	text
6	PLZ	int(11)
7	Wohnort	text

In der Funktion ist die $plz kein string, aber auch kein ‚int'. Ohne reicht

```
function registrierenUser(string $username,string $password,string $vorname,string $nachname, $plz,
```

Beachte: KEIN scharfes ß in der Datenbank – kein straße!!!

LOGIN ändern:

Damit man sich nun mit einem „gehashten" Passwort auch wieder einloggen kann, muss man in der Route von „login" dies berücksichtige. Zuvor hatten wir ja kein gehastes Passwort in der Datenbank.

Also

Die Funktion „password_verify" verwenden:

```
57          if((bool)$password &&
58            isset($userData['password']) &&
59            false === password_verify($password,$userData['password'])){
60              $errors[]="Passwort stimmt nicht";
61          }
```

Erklärung in https://www.php.net/manual/de/function.password-verify.php

password_verify

(PHP 5 >= 5.5.0, PHP 7)
password_verify — Überprüft, ob ein Passwort und ein Hash zusammenpassen

Beschreibung

```
password_verify ( string $password , string $hash ) : boolean
```

Überprüft, ob ein Passwort und ein Hash zusammenpassen.

Beachte, dass password_hash() den Algorithmus, den Aufwand und den Salt als Teil des Hashes zurückgibt. Somit sind alle benötigten Informationen im Hash enthalten, was der Funktion erlaubt den Hash zu prüfen, ohne dass Informationen über den Salt oder den Algorithmus an anderer Stelle gespeichert werden müssen.

INFO:

Bestellen geht nur, wenn man eingeloggt ist. Hat man den Warenkorb vor dem Login befüllt, und loggt sich dann erst ein, ist der Warenkorb wieder leer. Um das zu vermeiden, kann man hier in diesem Youtube-File das ändern: https://www.youtube.com/watch?v=2NDq7tojttc

Bestellformular - https://www.youtube.com/watch?v=lzxoi-aW5Sk