Landesberufsschule 4 Salzburg

Übungen im

IT - Laboratorium

PHP

für die Übung Nr. 2

Katalog - Nr.: ***3Ib***

Name : ***Martina Janjic***

Jahrgang : ***Mai – Juli, 2021***

Datum der Übung : ***08.06.2021***

Inhalt

1 Anweisung der Übung:

Ein PHP-Skript soll Daten eines HTML-Formulars in einer .csv Datei speichern, in eine DB hochladen und eventuell noch unterscheiden, ob jemand schon ein Konto besitzt und sich dadurch anmelden kann.

1 Einleitung

PHP (Hypertext Präprozessor) ist eine serverseitige Skriptsprache zur dynamischen Erstellung von Webseiten. Das bedeutet, dass sie nicht beim Client so ankommt.

Bei jedem Aufruf wird die Seite neu generiert, was mehr Energie benötigt.

Variablen beginnen mit $, bezeichnungen müssen mit unterstrich oder buchstaben beginnen

Php ist typenlos integer – grundtypen existieren trotzdem string float bool

Punktoperator

Konstruktotr nicht selben namen wie klasse

Attribute mit var sind public

Enxtends vererbung

Kommentar php /**/

2 Inventarliste

2.1 Hardware

- IT-87, PC Dell 7950 SN. (i7-8665U, 16GB RAM, 500 GB HDD, …)

2.2 Software

- Text Editor *.txt
- Notepad++
- Microsoft Edge

3 Übungsdurchführung

3.1 Grundaufbau

Damit XAMPP meine Daten leicht findet, habe ich sie in einem selbsterstellten Ordner im XAMPP-Dateipfad erstellt. *C:\xampp\htdocs\myOwnFiles*

Ich habe meine Anmeldeseite von der ersten Übung umgeändert, sodass es eine Anmelde- und Registrierungsseite ist. *phpLogin.html*

Von der HTML-Seite führen zwei Buttons zu PHP-Skripts, die den Nutzer entweder Registrieren oder Anmelden versuchen. *Register.php* und *Login.php*
Die Anmeldeaktion erfolgt durch ein HTML-internen Skriptaufruf:

```
<button id="btnLogin" onclick="login()" name="btnLogin"
type="submit">Anmelden</button>
-----
function login(){
form1.action = "/myOwnFiles/Login.php";}
```

Zudem verwende ich in den PHP-Seiten $_POST, also musste ich die methode beim HTML-Form umändern, weil sie vorher GET war:

```
<form id="form1" onsubmit="return fname()" method="POST">
```

3.2 PHP

3.2.1 Daten von HTML an PHP

Um die Formulardaten von HTML nach PHP übertragen zu können, war es nötig Variablen mit der HTML-Option **name** zu bilden.

```
<label for="txtUsername" >Username</label>
<input id="txtUsername" name="txtUsername" type="text" autofocus
placeholder="Username" required>
```

Leider dachte ich anfänglich, dass id dafür zuständig war und übergab deshalb NULL an die PHP-Seite.

Alle meine HTML-Felder haben **required**, also dürfte theoretisch kein leeres Feld übertragen werden. Trotzdem teste ich sie mit einem **isset()**

```
if (isset($_POST['txtUsername']) || isset($_POST['txtPasswort']))
{
     $txtUsername = $_POST['txtUsername'];
     $txtPasswort = $_POST['txtPasswort'];
```

$** erstellt eine Variabel in PHP und **$_POST sendet den Wert des definierten **name** weiter.

3.2.2 Password_hash: Salt & Pepper

Ein Aufgabenkriterium war, dass Passwörter nicht als Cleartext gespeichert werden dürfen.

Hash verändert einen String irreversibel und das Salt ist dafür da, dass bei jedem Hash von sogar demselben Passwort, verschiedene Hashes erstellt werden würden.
Pepper wird im Vergleich zum Salt nicht in der Datenbank gespeichert, sondern an einem geheimen Ort und gilt für alle Passwörter, um diese zu erschweren.

```
$password1 = password_hash('$txtPasswort', PASSWORD_DEFAULT);
```

.crypt() wäre zum Encrypten und würde es ermöglichen, das Passwort wieder in Cleartext umwandeln zu können.

3.2.3 Write to csv

PHP hat eine simple Schreibfunktion **fputcsv(),** um Daten in .csv Dateien speichern zu können.

```
$f = fopen('myCsv.csv', 'a');
fputcsv($f, ["$txtUsername", "$password1"]);
fclose($f);
```

3.2.4 Write from csv to sql

Leider konnte ich keinen Weg finden, **fopen(x, ‚a‘)** wiederzuverwenden, weil es ‚a‘ als Aktion braucht und beim read eine Art von ‚r‘ benötigt wird. Deshalb musste ich leider ein neues **fopen()** erstellt.

```
$g = fopen('myCsv.csv', 'r+');
$con = mysqli_connect('localhost', 'root', '', 'php_db_test');
while (($data = fgetcsv($g)) !== false)
{
     $sql = "INSERT INTO `table1` (`id`, `flUsername`, `flPasswort`)
VALUES ('0', '$data[0]', '$data[1]' );";
}
$rs = mysqli_query($con, $sql);
fclose($g);
mysqli_close($con);
```

$con verbindet sich mit meiner zuvor erstellten Datenbank und **mysqli_query()** schreibt den insert-Befehl in table1 von dieser Datenbank.

Im **while()** wird die CSV_Datei gelesen, bis es keine Einträge mehr gibt und pro Eintrag werden der Usernahme und das Passwort in die Spalten flUsername und flPasswort von table1 in $sql gespeichert, sodass sie mit **mysqli_query()** letztendlich zur Datenbank gesendet werden.

3.2.5 verify hashed password

Weil unsere Serverpasswörter gehasht sind, gibt es keine Möglichkeit, sie wieder nach Cleartext umzuwandeln. Um trotzdem Anmeldeversuche erlauben zu können, wird beim Login das jeweilige Passwort vom User selektiert und **password_verify(string $password, string $hash)** vergleicht die Cleartext Eingabe mit der gehashten für uns.
Es übergibt den Algorithmus, Aufwand und Salt als Teil des Hashes zurück, was genug Information ist um den Hash zu vergleichen, ohne dass sie irgendwo gespeichert wird. Der return-Type ist ein boolean.

```
$con = mysqli_connect('localhost', 'root', '', 'php_db_test');
$sql = "SELECT flPasswort FROM `table1` WHERE flUsername='$password' LIMIT 1;";
$rs = mysqli_query($con, $sql);
mysqli_close($con);
return password_verify($password, $rs['$username']);
```

4 Erkenntnisse

PHP ist eine simple Weise HTML mit anderen Funktionen zu verbinden. Es hat ebenfalls viele spezifische Funktionen, die einem die Arbeit erleichtern.
Ich weiß nicht, wie oft ich PHP in der Zukunft verwenden werde, aber die Syntax verstehe ich jetzt definitiv leichter.