Passwortgenerator in PHP

Description

Passwörter brauchen wir jeden Tag. Sei es die PIN, TAN, im Internet oder Games. Und manchmal ist es hilfreich, zufällige Passwörter zu erzeugen. Dies kann ebenfalls für ein Spiel nützlich sein, aber auch für selbstgeschriebene Webseiten. Moderne Browser und CMS‘ generieren sie selbst, es gibt dennoch Situationen, in denen man einen eigenen Generator haben will. Dieses Tutorial zeigt, wie man es in PHP realisiert.

Was haben wir vor?

Einen Passwortgenerator in PHP zu schreiben ist nicht schwierig. Bereits nach wenigen Zeilen Code hat man ein brauchbares Resultat. Im nachfolgenden Tutorial möchte ich es dennoch erweitern. Wir werden Funktionen für Arrays und die Generierung erhalten. Dies ist bei einem größeren Webprojekt nützlich.

Außerdem werden wir verschiedene Arten von Passwörtern erstellen. Das Resultat kannst Du Dir hier ansehen. Auf Design und unnützen Schnickschnack lege ich dieses Mal keinen Wert. Es geht um Funktionalität.

Wozu Arrays?

Irgendwo müssen die Zeichen für ein Passwort herkommen. Und da bietet es sich an, bestimmte Zeichen als Arrays zu speichern. Aus diesen Arrays können anschließend zufällige Zeichen zurückgegeben werden. Das gleich gezeigte Prinzip ist also sehr einfach.

Um den Überblick zu wahren, stecken wir alles in eine Datei. Das ist kein guter Stil. Besser ist es, wenn man die Arrays und die anderen Funktionen jeweils in eine separate Datei steckt. Für dieses Tutorial ist es aber ausreichend.

mb_internal_encoding

Wir erstellen eine neue Datei. Bei mir heißt sie *index.php*, weil sie in einem eigenen Ordner liegt. Wie ihr sie nennt, ist egal, so lange die Endung *.php* lautet. Unsere ersten beiden Zeilen lauten:

Die erste Zeile sollte klar sein. Wir starten mit PHP. Und die zweite? Encodin
UTF-8 sind.

Die vier Arrays

Die im oben verlinkten Beispiel erzeugten Passwörter beruhen auf vier Arrays,

get_number()

```
function get_number()
{
 $numbers = array('0', '1', '2', '3', '4', '5', '6', '7', '8', '9');
 return $numbers[mt_rand(0, sizeof($numbers) - 1)];
}
```

Ja, das war es schon. Wir erstellen eine Funktion mit dem entsprechenden Namen
$numbers
 mit dem Array. Das Array enthält die Zahlen 0 bis 9. Anschließend geben wir m
return eines dieser Zeichen zufällig zurück. Dazu nehmen wir das Array
$numbers[] (man beachte die eckigen Klammern) und rufe die PHP-Funktion
mt_rand() auf. Hier brauchen wir zwei Parameter, **min** und **max**
, also Anfang und Ende einer möglichen Zahl. Arrays beginnen immer mit 0. Nun
sizeof($numbers) ab.

Aber warum *-1*? Nun, *sizeof*
 ist ein Zähler. In dem Fall schaut er sich das Array *$numbers*
 an und gibt zurück, wie viele Werte enthalten sind. Wir brauchen aber nicht e
sizeof nämlich 0, bedeutet es, dass es nichts finden konnte.

Nach dem selben Prinzip arbeiten auch die folgenden Funktionen.

get_char()

```
function get_special()
```

```
{
 $specials = array('!', '??', '$', '%', '&', '-', '_', '[', ']', '(', ')', '='
 return $specials[mt_rand(0, sizeof($specials) - 1)];
}
```

get_shorts()

```
function give_passwords_speak()
{
 $password   = „";

 for ($i=0; $i<=4; $i++)
 {
  $password.= get_shorts();
 }

 $password.= rand(1, 100);

 $password = ucfirst($password);

 return $password;
}
```

Man versteht den Code besser, wenn man die Absicht dahinter kennt. Wir wollen

Die for-Schleife läuft viermal durch. Dabei wird *get_shorts()*
 aufgerufen. Darin steckt das Array und wir erhalten einen zufälligen Laut zur
‘her’. Diesen fügen wir der Variable *$password*
 an. Das Anfügen geschieht mit *.=*. Danach wird mit der PHP-Funktion
rand(1, 100)
 eine Zahl zwischen 1 und 100 generiert und ebenfalls angehängt. Die PHP-Funkt
ucfirst()
 verwandelt das erste Zeichen eines Strings in einen Großbuchstaben. Zum Schlu
return zurück.

give_passwords_low()

```
function give_passwords_low($arg1)
{
 $signs = $arg1;

 $password   = „";

 for ($i=0; $i<$arg1; $i++)
 {
  $numOrChar = rand(0, 1);
```

```
  if ($numOrChar == 0)
  {
   $password.= get_number();
  } else {
   $char   = get_char();
   $bigOrSmall = rand(0, 1);

   if ($bigOrSmall == 0){$char = strtoupper($char);}
   $password.= $char;
  }
 }

 return $password;
}
```

Ist etwas länger, aber kaum komplexer. Wir wollen ein einfaches Passwort mit z
$arg1 bzw. *$signs*
 betrifft nur die Anzahl der Zeichen. Wir können also sehr kurze, aber auch un

$numOrChar = rand(0, 1);
 entscheidet darüber, ob es eine Zahl oder ein Buchstabe wird. Bei der Zahl is

Spannender sind die Buchstaben. *$char = get_char();*
 holt einen der Buchstaben aus dem Array. *$bigOrSmall = rand(0, 1);*
 entscheidet darüber, ob der Buchstabe groß oder klein angezeigt wird. Wenn gr
strtoupper()
 den Buchstaben in einen Großbuchstaben um. Der Rest funktioniert, wie bekannt

give_passwords_save()

Jetzt zum sicheren Passwort. Die Sicherheit erhöht sich mit Sonderzeichen, wei

```
function give_passwords_save($arg1)
{
 $signs    = $arg1;

 $password   = „";

 for ($i=0; $i<$arg1; $i++)
 {
  $numCharOrSpecial = rand(0, 100);

  if ($numCharOrSpecial <=30)
  {
   $password.= get_number();
  } else if ($numCharOrSpecial <=80){
   $char   = get_char();
   $bigOrSmall = rand(0, 1);
```

```
   if ($bigOrSmall == 0){$char = strtoupper($char);}
   $password.= $char;
  } else {
   $password.= get_special();
  }
 }

 return utf8_encode($password);

}
```

Auch hier haben wir ein Argument für die Länge. Die Generierung des Passworts

- Die Chance liegt bei 30% (*$numCharOrSpecial <=30*), dass das Zeichen eine Zahl ist.
- Sie liegt bei 50% (*$numCharOrSpecial <=80*), dass es ein Buchstabe ist, was wiederum in Groß- und Kleinbuchstaben unterteilt wird.
- Sie liegt bei 20%, dass es ein Sonderzeichen wird.

Man kann also nicht ein wenig an der „Intensität” des Passworts schrauben. Am Ende wird es wieder zurückgegeben.

Ausgabe als HTML

Nun haben wir alle Arten von Passwörtern und können mit ihnen eine Party feiern. Die Einladung erfolgt per HTML und PHP. ?

Die erste Zeile, also *?>*
, brauchen wir nur, wenn wir wirklich alles in eine Datei schreiben und sich d
echo
 ausgebe. Dazwischen haben wir unsere Schleifen. Wir geben je fünf Passwörter

Date Created
27. November 2021
Author
sven