Login Session mit Datenbankanbindung

Ziel

Wir wollen zeigen, wie man ein einfaches Authentifizierungs Login mittels einer Datenbankanbindung realisieren kann. Als Vorbereitung müssen wir zuerst eine Datenbank mit bereits registrierten Benutzern anlegen.

An diesem Beispiel soll gezeigt werden, wie mittels der superglobalen Variable **$_SESSION** während einer Folge von Aufrufen Informationen auf dem Server festgehalten werden können.

In einem Formular kann der Benutzer seine Daten, in diesem Falle **Benutzername** und **Passwort** eingeben. Diese Informationen werden mittels der POST-Methode dem Server übermittelt. Der Server validiert die eingegebenen Parameter. Das Passwort sollte dabei nie im Klartext gespeichert werden, sondern wird gerade nach dem Auslesen in einen sogenannten Hashwert umgewandelt (als Beispiel MD5, gilt aber inzwischen als nicht mehr genügen sicher).

Mittels eines Aufrufs an die Datenbank, wird mit dort abgespeicherten Daten verglichen, ob der entsprechende Benützer berechtigt ist in die internen Seiten zu gelangen.

Voraussetzungen

Voraussetzungen sind ein funktionstüchtiges PHP und eine MySQL-Datenbank. Die Datenbank wird in PHP wie folgt aufgerufen:

```
$host = "localhost";
$user = "root";
$pw = "";
$database = "loginsystem";
$table = "benutzerdaten";
```

Datenbank `loginsystem` muss schon bestehen.

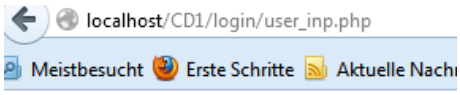
Anlegen der Datenbank

Datenbank: **loginsystem** mit Hilfe von **phpMyAdmin** anlegen

mit **phpMyAdmin** UserTabelle.sql importieren

Benutzer anlegen

Mit folgendem Skript werden die Benutzer `admin` pw:`admin` und `test` pw:`abc` angelegt

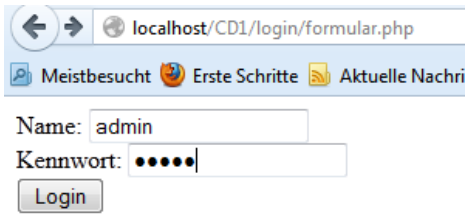


Benutzer erfolgreich angelegt.
Benutzer erfolgreich angelegt.

Formular zum Einlesen der Zugangsdaten

Jetzt legen wir eine Datei an, die das Loginformular enthält. Das könnte theoretisch auch eine reine HTML-Datei sein. Wir erstellen an dieser Stelle allerdings eine PHP-Datei da bei einem falschen Passwort über dem Formular eine Meldung ausgegeben werden soll.

Bitte beachten Sie, dass für den korrekten Ablauf der Funktionen alle hier erstellten Dateien im gleichen Verzeichnis liegen müssen.




```php
<?php session_start (); ?>
<html>
<head>
  <title>Login</title>
</head>

<body>
<?php
if (isset ($_REQUEST["fehler"]))
{
  echo "Die Zugangsdaten waren ungültig.";
}
?>
<form action="login.php" method="post">
  Name: <input type="text" name="name" size="20"><br>
  Kennwort: <input type="password" name="pwd" size="20"><br>
  <input type="submit" value="Login">
</form>
</body>
</html>
```

Login

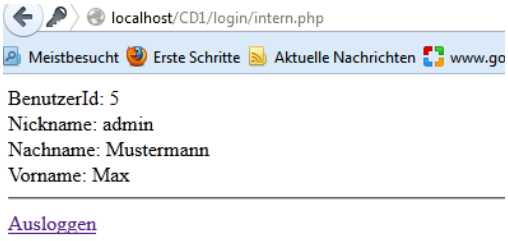
In dieser Datei werden die Zugangsdaten geprüft. Sind diese richtig, werden die Benutzerdaten in einer Session gespeichert und auf eine Inhaltsseite weitergeleiten, die nur zu sehen ist, wenn der Benutzer eingeloggt ist.

Beim Klicken auf den Login-Button wird das `login.php` Skript aufgerufen. In dieser Datei werden die Zugangsdaten geprüft.

`login.php`

```php
<?php
// Session starten
session_start ();

// Datenbankverbindung aufbauen
$connectionid = mysql_connect ("localhost", "root", "");
if (!mysql_select_db ("test", $connectionid))
{
  die ("Keine Verbindung zur Datenbank");
}

$sql = "SELECT ".
    "Id, Nickname, Nachname, Vorname ".
  "FROM ".
    "benutzerdaten ".
  "WHERE ".
    "(Nickname like '".$_REQUEST["name"]."') AND ".
    "(Kennwort = '".md5 ($_REQUEST["pwd"])."')";
$result = mysql_query ($sql);

if (mysql_num_rows ($result) > 0)
{
  // Benutzerdaten in ein Array auslesen.
  $data = mysql_fetch_array ($result);

  // Sessionvariablen erstellen und registrieren
  $_SESSION["user_id"] = $data["Id"];
  $_SESSION["user_nickname"] = $data["Nickname"];
  $_SESSION["user_nachname"] = $data["Nachname"];
  $_SESSION["user_vorname"] = $data["Vorname"];

  header ("Location: intern.php");
}
else
{
  header ("Location: formular.php?fehler=1");
}
?>
```

Sind diese richtig, werden die Benutzerdaten in einer Session gespeichert und auf eine Inhaltsseite weitergeleitet, die nur zu sehen ist, wenn der Benutzer eingeloggt ist.



intern.php

```php
<?php
include ("checkuser.php");
?>
<html>
<head>
  <title>Interne Seite</title>
</head>
<body>
  BenutzerId: <?php echo $_SESSION["user_id"]; ?><br>
  Nickname: <?php echo $_SESSION["user_nickname"]; ?><br>
  Nachname: <?php echo $_SESSION["user_nachname"]; ?><br>
  Vorname: <?php echo $_SESSION["user_vorname"]; ?>
  <hr>
  <a href="logout.php">Ausloggen</a>
</body>
</html>
```

checkuser.php

```php
<?php
session_start ();
if (!isset ($_SESSION["user_id"]))
{
  header ("Location: formular.php");
}
?>
```

Logout

logout.php

```php
<?php
// Wird ausgeführt um mit der Ausgabe des Headers zu warten.
ob_start ();

session_start ();
session_unset ();
session_destroy ();

header ("Location: formular.php");
ob_end_flush ();
?>
```

Ganz zum Schluss wollen wir noch schnell die Datei für die Logout-Funktion schreiben. Diese Datei löscht die Inhalte der Session und leitet auf das Loginformular weiter.

Aufgaben:

- Code auf eigener Umgebung in Betrieb nehmen
- Testen Sie alle Funktionalitäten
- Code sollte soweit verstanden werden, dass er auf die eigene Website angewendet werden kann
- Registrieren Sie weitere Benutzer, die berechtigt sind einzuloggen. **(user_inp.php)** anpassen
- Validieren Sie die Formulardaten sowohl auf client- wie auch auf Serverseite
- Schreiben Sie ein eigenes Scpript für die Registrierung
- Cracken Sie die Passwörtern, indem Sie den generierten Hashwert aus der Datenbank nehmen und mit sogenannten Rainbow Tables im Netz auswerten lassen
- Verwenden Sie einen Hashalgorithmus mit mehr Sicherheit als md5

Anhang:

session_start()

Session(datei) auf dem Server erstellen. Wurde eine gültige Session-ID übergeben, werden die in den Sessiondaten gespeicherten Werte in dem $HTTP_SESSION_VARS-Hash wiederhergestellt und abhängig von der Einstellung der register_globals auch als Variablen reinitialisiert.

session_register()

Eine oder mehrere Variablen zur Speicherung in die Session(datei) vormerken. session_register() impliziert ein session_start(). Das bedeutet, dass man session_start() in diesem Fall auch weglassen könnte. Bei den Parametern der session_register() Funktion handelt es sich nicht um die Variablen selbst, sondern um ihre Namen. Diese Parameter besitzen kein führendenes "$".

session_register() merkt eine Variable vor (registriert sie), die dann am Ende des Scriptes in die Session(datei) geschrieben wird, und die zwischen dem Registrieren und dem Scriptende verschiedene Werte haben kann. Es wird jeweils der letzte Wert dieser Variablen in den Sessiondaten gespeichert. D.h. Die Werte der registrierten Variablen sind frühestens, nach der Beendigung des aktuellen Scriptes und beim nächsten session_start() wieder verfügbar.

session_unregister()

Eine oder mehrere Variablen aus der Session(datei) verwerfen. Dem session_unregister() muß ein session_start() oder session_register() vorangehen.

session_destroy()

Die Funktion session_destroy() veranlasst alle Variablen einer Session zu verwerfen und die Session(datei) löschen.

Session-ID

Die sog. Session-ID ist ein zufällig ausgewählter Schlüssel, der die Sessiondaten auf dem Server eindeutig identifiziert. Dieser Schlüssel kann z.B. über Cookies oder als Bestandteil der URL an ein Folgescript übergeben werden, damit dieses die Sessiondaten auf dem Server wiederfinden kann.

Fallback

Viele Browser-User haben in ihrem Browser die Cookies aus diversen Gründen deaktiviert, oder lassen sich jedes Cookie bestätigen. Sollte ein Cookie auf dem lokalen Rechnersystem nicht gesetzt werden können, muss ein Ersatzmechanismus her - ein sog. Fallback. In diesem Fall ist der Fallback (Rückfall / Atavismus) eine Ersatzmethode, die die Übergabe der Session-ID an ein Folgescript ohne Cookies erlaubt.

PHPSESSID ist der Default-Name der Session. Möchte man diesen Namen aus z.B. ästhetischen Gründen modifizieren - vor allem, wenn er als GET-Parameter als Teil der URL sichtbar wird - so kann man dies in der php.ini, der Webserverkonfiguration oder direkt mit PHP bewerkstelligen.

D. A. Waldvogel, 08.03.2017
M133_Login_V1.docx