Session Handling

Das vorliegende PHP Projekt verwendet Sessions um den Endbenutzer eindeutig zu identifizieren. Um den Zugriff von JavaScript auf die Session zu verhindern wird der PHP Parameter „session.use\_only\_cookies“ auf TRUE gesetzt. Auf das im nachfolgenden Schritt erzeugte Cookie „sec\_session\_id“ kann nur mit dem http Protokoll zugegriffen werden.[[1]](#footnote-1) Die Erstellung des Cookies ist in der PHP Funktion „sec\_session\_start()“ gekapselt und wird auf jeder PHP Seite zum Erstellen einer Session verwendet.

Login Verfahren

Die Veranstaltungsmanagement Software verwendet einen sha512 Hashing-Algorithmus[[2]](#footnote-2) für die Benutzerpasswörter. Zusätzlich werden die Passwörter in der Datenbank mit einem Salt[[3]](#footnote-3) abgelegt um die Sicherheit weiter zu erhöhen. Im Folgenden wird das eigentliche Login Verfahren anhand des Schaubilds näher erklärt.

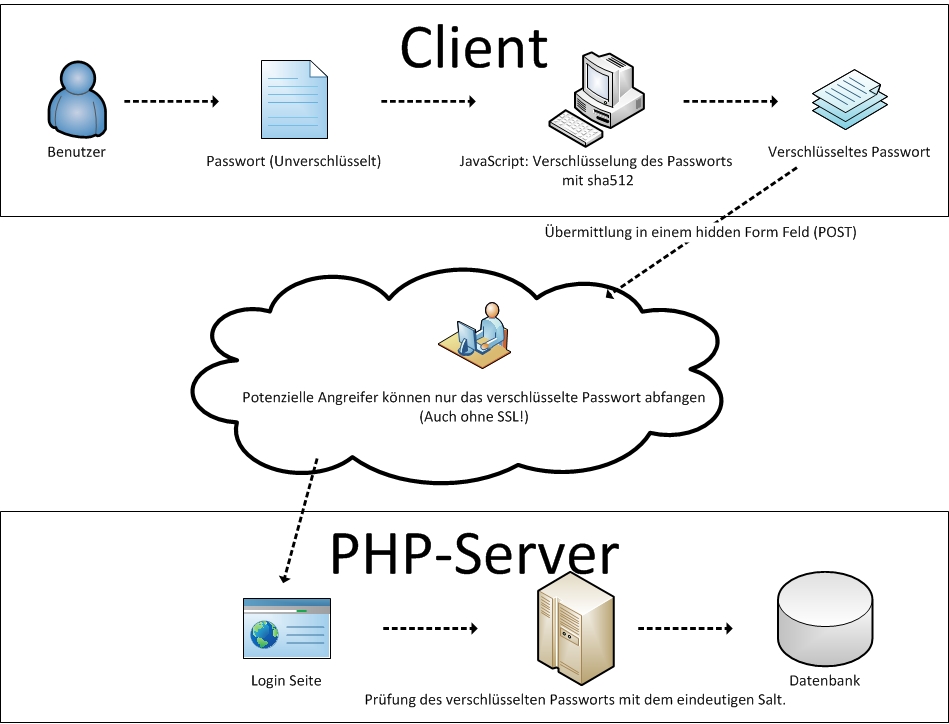


Abbildung Client / Server Login Verfahren

Der Benutzer gibt seine Login Daten in der HTML Form „login\_form“ ein. Nach dem Bestätigen per Tastendruck auf „Login“ wird die JavaScript Methode „formhash()“ ausgeführt.

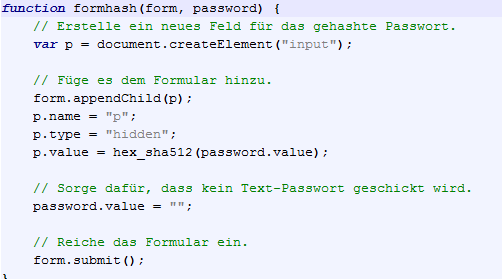


Abbildung JavaScript Methode formhash()

Die Methode erzeugt ein Form Feld in dem das Passwort mit einer sha512 Verschlüsselung abgelegt wird. Das Klartext Passwort Feld wird danach geleert und die Form bestätigt. Durch das leeren des Klartext Feldes wird das Passwort nicht mehr mit POST übertragen.

Nach erfolgreicher Übertragung des verschlüsselten Passwortes an den Server wird die PHP Funktion „login()“ ausgeführt. Diese Methode ermittelt anhand der eingegebenen E-Mail Adresse das in der Datenbank hinterlegte verschlüsselte Passwort mit zugehörigen Salt. Zusätzlich wird geprüft ob der Benutzer innerhalb der letzten zwei Stunden versucht hat sich mehr als fünf Mal erfolglos Anzumelden. Dies dient dem Schutz gegen Bruteforce Angriffe. Falls ein Bruteforce Angriff ausgeschlossen werden kann, wird das in der Datenbank hinterlegte Passwort mit dem übertragenen Passwort abgeglichen. Falls der Login Versuch erfolgreich ist wird der Benutzer direkt auf die geschützte interne Seite weitergeleitet. Um sicherzustellen dass der Benutzer Zugriff auf die internen Seiten hat, wird mittels der in PHP implementierten Funktion „login\_check()“ geprüft ob der Benutzer angemeldet ist. Dabei wird geprüft ob die SESSION Variable eine gültige Benutzerkennung enthält und ob der verwendete Browser mit dem in der SESSION hinterlegtem Browser übereinstimmt.

Installation der Datenbank

Grundlegende Konfigurations Parameter

Ordner: *Project/functions/mysql.php*

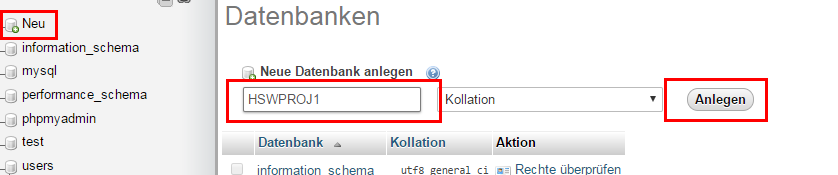
|  |  |  |
| --- | --- | --- |
| Parameter | Standardwert | Beschreibung |
| error\_reporting(); | E\_ALL | Alle Fehler ausgeben. Im Produktiv System deativieren! |
| HOST | localhost | Datenbank Server |
| USER | root | Datenbank Benutzer |
| PASSWORD |  | Passwort für Datenbank Benutzer |
| DATABASE | HSWPROJ1 | Datenbankname |
| CAN\_REGISTER | any | Registrierung für alle erlauben |
| DEFAULT\_ROLE | member | Standardrolle der Benutzer |
| SECURE | FALSE |  |

Datenbank Import

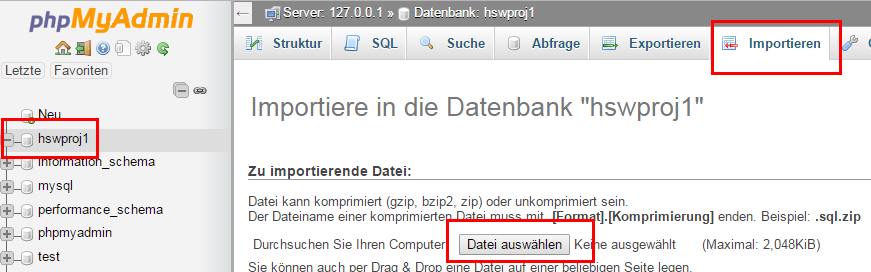
Datenbankstruktur Template: *Project/*doc/*HSWPROJ1\_DB.sql*

PHPMyAdmin: <http://localhost/phpmyadmin>

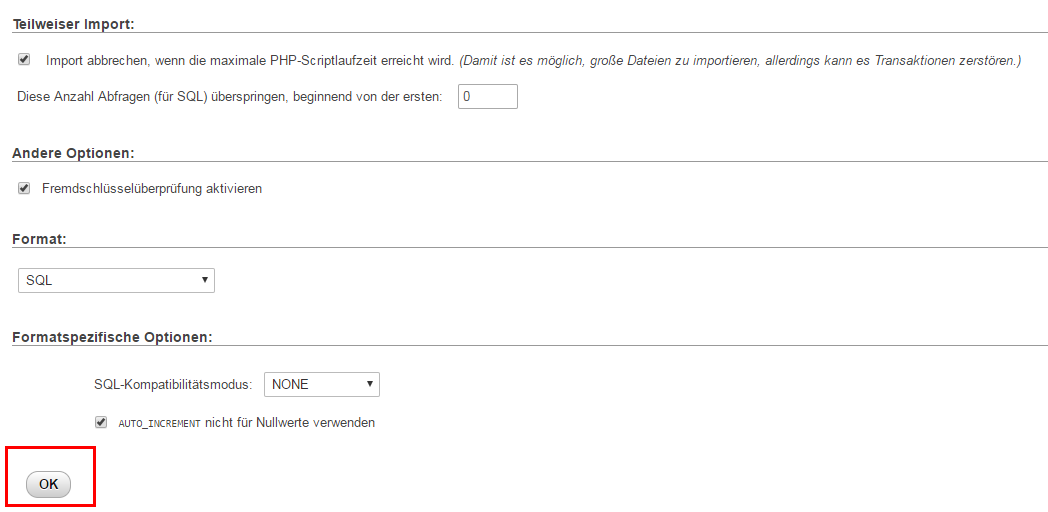
Datenbank HSWPROJ1 Anlegen:

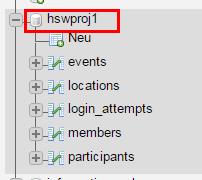


Datenstrukturen importieren:



Datei „*Project/*doc/*HSWPROJ1\_DB.sql“* auswählen.





Die Datenbank ist nun vollständig importiert und das Veranstaltungsmanagement kann genutzt werden.

1. <http://php.net/manual/de/function.session-get-cookie-params.php> (2017) [↑](#footnote-ref-1)
2. JavaScript Implementierung übernommen von http://pajhome.org.uk/crypt/md5/sha512.html [↑](#footnote-ref-2)
3. Salt bezeichnet in der Kryptographie eine zufällig gewählte Zeichenfolge, die an einen gegebenen Klartext vor der Verwendung als Eingabe einer Hashfunktion angehängt wird. [↑](#footnote-ref-3)