Datenbanken schnell und bequem sichern mit PHP

Dr. Volker Thormählen, 10. Dez. 2018

Inhalt

Abbildungen

Akronyme

CSS	Steht für „*Cascading Style Sheets*". Es ist eine *Gestaltungs- und Formatierungssprache* für Webseiten und -anwendungen.
HTML	Steht für „*HyperText Markup Language*". Es handelt sich um eine *Auszeichnungssprache* für Webseiten und -anwendungen.
PHP	Steht für „*Hypertext Preprocessor*" (ursprünglich „*Personal Home Page Tools*"). Es ist eine verbreitete (client- *und* serverseitige) Skriptsprache, die zur Programmierung von Webseiten und -anwendungen geeignet ist und in HTML (s. o.) eingebettet werden kann.
SQL	Steht für „*Structured Query Language*". Es ist eine *Datenbanksprache* zur Erstellung von Datenstrukturen in relationalen Datenbanken (wie z. B. MySQL) sowie zum Bearbeiten und Abfragen der darin enthaltenen Datentabellen.

Tabelle 1: Akronyme für weborientierte Programmiersprachen

Listings

Tabellen

1 Aufgabe

Im Folgenden wird ein modulares und strukturiertes PHP-Skript beschrieben, das sich zur Sicherung von einer oder mehreren Datenbanken eignet, wie sie in beliebten Datenbanksystemen wie MySQL[1] bzw. MariaDB verwendet werden. Die Besonderheiten sind:

- Der Sicherungsort der SQL-Dateien ist transparent, aber im PHP-Skript fest vorgegeben, um den Eingabeaufwand zu minimieren.
- Der Benutzer kann mittels Auswahllisten entscheiden, ob nur eine bestimmte Datenbank als SQL-Datei gesichert werden soll oder alle.
- Der Benutzer kann ebenfalls bestimmen, ob bereits gesicherte SQL-Dateien gelöscht werden sollen und zugleich festlegen, ab wieviel Kalendertagen eine gesicherte SQL-Datei als obsolet (d. h. überaltert) gelten soll[2].
- Die Eingabedaten des Formulars bleiben nach dem Absenden erhalten, auch wenn deren Validierung zu Fehlermeldungen führt.

Im Folgenden wird die Lösung der obigen Aufgabe beschrieben, beginnend mit der Präsentation des zugehörigen Eingabeformulars und dessen Gestaltung. Danach werden die alle Einzelheiten der Anwendung beschrieben:

1. Das HTML-Eingabeformulars und dessen internes CSS-Stylesheet zur Formatierung
2. Die benutzerdefinierten PHP-Funktionen zur Modularisierung der Anwendung
3. Die Validierung und der Erhaltung der Werte des Eingabeformulars nach dem Absenden
4. Die Weiterverarbeitung der fehlerfreien Formularwerte

Anschließend werden wichtige Details des PHP-Skripts erläutert (s. Abschnitt 3, Seite 10).

Grundkenntnisse über weborientierte Programmiersprachen (s. Tabelle 1) werden vorausgesetzt.

2 Lösung

2.1 Steuerelemente des Eingabeformulars

Abb. 1 zeigt das Eingabeformular des PHP-Skripts mit insgesamt 14 Steuerelementen, unterteilt in 4 Feldgruppen:

1.	Speicherort	(4 schreibgeschützte, einzeilige T*extfelder*)
2.	Anmeldung	(1 *Kombinationsfeld* und 1 Passwortfeld)
3.	Obsolete SQL-Dateien entfernen	(1 Kontrollkästchen und 1 *Kombinationsfeld*)
4.	Was sichern?	(2 *Optionsfelder* und 1 *Kombinationsfeld*)

Hinzu kommen noch 3 Befehlsschaltflächen unterhalb der genannten 4 Feldgruppen. Diese Steuerelemente und die jeweils dazugehörigen *Bezeichnungsfelder* sind aus Abb. 1 ersichtlich. Pflichtfelder sind im Formular jeweils durch ein hochgestelltes rotes Sternchen gekennzeichnet.

[1] **MySQ**L bzw. **MariaDB** ist der Name eines weit verbreiteten relationalen Datenbankmanagementsystems.
[2] Die am 25. Mai 2018 in Kraft getretene Datenschutz-Grundverordnung (**DSGVO**) sieht in Art. 17 ("*Recht auf Vergessenwerden*") strikte Löschpflichten für personenbezogene Daten vor.

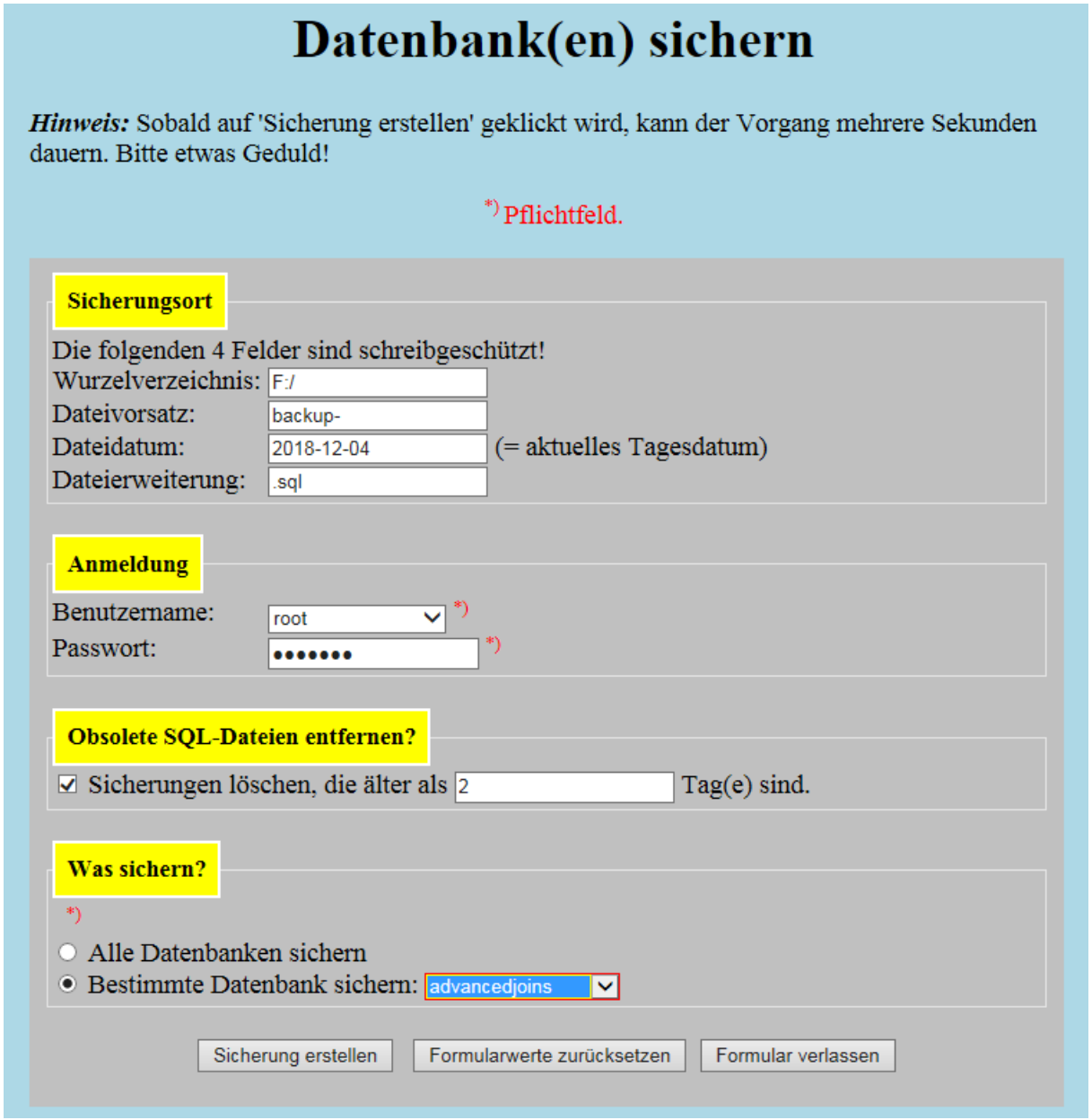

Abb. 1: HTML-Formular zur Sicherung aller Datenbanken oder einer best. Datenbank mit PHP

2.2 Formatierung des Eingabeformulars mit CSS-Anweisungen

Die Gestaltung des HTML-Eingabeformulars (s. Abb. 1) erfolgt mit CSS-Anweisungen innerhalb des <head>-Tag[3] der HTML-Seite (s. Listing 1):

```
<head>
    <title>Datenbank(en) mittels PHP-Skript sichern</title>
    <!-- Interne CSS Anweisungen -->
    <style type="text/css">
        /* Zeichenkodierung */
        @charset "UTF-8";
        /* HTML-Seite */
        body {
            width:630px;
            background-color:lightblue;
            margin-left:auto;
            margin-right:auto;}
        /* Formular */
        form {
            width:600px;
            height:570px;
            padding:10px 10px;
            background-color:silver;
        }
        /* Pseudo-Elemente */
        input[type=radio]:hover,input[type=checkbox]:hover,
        input[type=radio]:focus,input[type=checkbox]:focus {
            outline:1px solid red;
        }
        input:hover, input:focus, select:hover, select:focus {
            background-color:yellow;
            border:1px solid red;
            color:black;
        }
        /* Beschriftung einzelner Formularfelder */
        .lbl {
            float:left;   /* Beschriftung an der linken Seite des aktuellen Textflusses platzieren */
            width:150px;
        }
        /* Beschriftung für eine Gruppe von Formuarfeldern */
        legend {
            padding:0.5em 0.5em;
            border:2px solid white;
            color:black;
            font-size:90%;
            text-align:right;
            background-color:yellow;
        }
        /* Textanordnung aufheben */
            br{clear:left;}
        /* Fehlerklasse: Schriftfarbe rot */
            .error {color:red;}
        </style>
</head>
```

Listing 1: PHP-Skript, 1. Teil: Formatierungsanweisungen für das Eingabeformular

Ausführlich erklärt werden CSS-Anweisungen u. a. in [1].

[3] Ein **Tag** ist eine HTML-Anweisung in spitzen Klammern, auch *HTML-Markup* genannt.

2.3 Benutzerdefinierte PHP-Funktionen

Die in PHP eingebauten Funktionen müssen oft für spezielle Aufgabenstellungen durch benutzerdefinierte Funktion ergänzt werden. Das vorliegende PHP-Skript umfasst 5 solche Funktionen:

Aufgabe der Funktion	Name und Argument(e) der Funktion
Formulareingaben filtern	test_input($data) **$data:** Übergebener Formularwert
Datenbanken ermitteln für das entsprechende Kombinationsfeld im Formular	getDatabases($host, $usr, $pwd) **$host:** Name des (lokalen) Webservers (hier: *localhost*) **$usr:** Name eines berechtigten Benutzers der jeweiligen Datenbank **$pwd:** Passwort dieses Benutzers
Benutzer ermitteln für das entsprechende Kombinationsfeld im Formular	getUsers($host, $usr, $pwd, $dbname) **$host:** Name des (lokalen) Webservers (hier: *localhost*) **$usr:** Name eines berechtigten Benutzers der jeweiligen Datenbank **$pwd:** Passwort dieses Benutzers **$dbname:** Name der zu sichernden Datenbank
Benutzernamen bereinigen und bestätigen.	filterName($usr){ **$usr** : Name eines berechtigten Benutzers der jeweiligen Datenbank.
Veraltete SQL-Dateien aus einem vorgegebenen Verzeichnis entfernen	delete_obsolete_sql_files($days, $folder, $ext) **$days:** Abstand in Tagen zur Bestimmung veraltender (d. h. obsoleter) SQL-Dateien im Verzeichnis mit dem Namen $folder **$folder:** Verzeichnis mit datierten SQL-Dateien **$ext:** Dateierweiterung (hier: *.sql) der datierten SQL-Dateien

Tabelle 2: Benutzerdefinierte PHP-Funktionen

```
// Formulareingaben filtern
function test_input($data){
        $data = trim($data);                    // Entfernt nicht benötigte Zeichen.
        $data = stripslashes($data);            // Entfernt Maskierungszeichen.
        $data = htmlspecialchars($data);        // Wandelt best. Sonderzeichen in HTML-Entitäten um.
        return $data;

}
```

Listing 2: Formulareingaben filtern

```
// Datenbanken ermitteln für das entsprechende Kombinationsfeld im Eingabeformular
function getDatabases($host,$root,$pwd){
    $conn= mysqli_connect('localhost','root','mvt1940');
    if (mysqli_connect_errno()){
        die("Verbindungsfehler: ".mysqli_connect_error());
    }
    // Ausgenommene Datenbanken
    $exclude_db = array('mysql','information_schema','performance_schema','phpmyadmin');
    $dbs = array();
    $sql = "SHOW DATABASES;";
    if ($result=mysqli_query($conn,$sql)){
        while($row = mysqli_fetch_row($result)){
            if(!in_array($row[0], $exclude_db)) {
                $dbs[] = $row[0];
            }
        }
        mysqli_free_result($result);
    }
    mysqli_close($conn);
    $cnt_dbs = count($dbs);
    if($cnt_dbs==0) {
        die("Abbruch: Keine Datenbanken für den DB-Export gefunden.");
    } else {
        return $dbs;
    }
}
```

Listing 3: Datenbanken ermitteln für das entsprechende Kombinationsfeld im Formular

```php
// Benutzer ermitteln für das entsprechende Kombinationsfeld im Eingabeformular
function getUsers($host, $usr, $pwd, $dbname){
    // Verbindung zur MySQL-Datenbank herstellen.
    $conn = mysqli_connect($host, $usr, $pwd, $dbname)
        or die("Verbindungsfehler: ".mysqli_connect_error());
    $sql = "SELECT DISTINCT user FROM user";
    $result = mysqli_query($conn,$sql);
    $users = array();
    if (!$result) {
        echo "Keine Benutzernamen gefunden!";
    } else {
        if (mysqli_num_rows($result) > 0) {
            while ($row = mysqli_fetch_assoc($result)) {
                $users[] = $row['user'];
            }
        }
        mysqli_free_result($result);
    }
    mysqli_Close ($conn);
    return $users;
}
```

Listing 4: Benutzer ermitteln für das entsprechende Kombinationsfeld im Formular

```php
// Benutzernamen bereinigen und bestätigen.
function filterName($usr){
    $user = filter_var(trim($usr), FILTER_SANITIZE_STRING);
    if(filter_var($usr, FILTER_VALIDATE_REGEXP,
           array("options"=>array("regexp"=>"/^[a-zA-Z\s]+/")))){
        return $usr;
    } else{
        return FALSE;
    }
}
```

Listing 5: Benutzernamen bereinigen und bestätigen

```php
// Veraltete SQL-Dateien aus einem vorgegebenen Verzeichnis entfernen.
function delete_obsolete_sql_files($days, $folder, $ext) {
    $count = 0;                          // Zähler für gelöschte SQL-Dateien.
    $files = glob($folder."*".$ext);  // SQL-Dateien bestimmen.
    foreach($files as $file){         // Alle SQL-Dateien durchlaufen.
        if(is_file($file)) {
            // filetime: Liefert Datum und Uhrzeit der letzten Dateiänderung.
            if (filemtime($file) < (time() - ($days * 24 * 60 * 60))) {
                unlink($file);           // Datei löschen.
                $count ++;               // Dateizähler erhöhen.
            }
        }
    }
    return $count;                       // Zahl der entfernten SQL-Dateien zurückgeben.
}
```

Listing 6: Veraltete SQL-Dateien aus dem vorgegebenen Verzeichnis entfernen

2.4 Validierung der Formulardaten

Die Prüfung der Formularwerte erfolgt nach dem Absenden mit den in Listing 7 benummerten PHP-Anweisungsblöcken:

```
// Ist das Eingabeformular abgesandt worden?
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Schalter für die Gültigkeit der erfassten Formularwerte setzen.
    $valid = true;

    // (1) Dateidatum = aktuelles Tagesdatum
    if(!empty($_POST["sql_date"])){
        $sql_date = test_input($_POST["sql_date"]);
    }

    // (2) Benutzername validieren.
    if(empty($_POST["username"])){
        $usernameErr = "Benutzername fehlt.";
        $valid = false;
    }else{
        $username = test_input($_POST["username"]);
        $username = filterName($username);
        if($username == false){
            $usernameErr = "Benutzername ist ungültig.";
            $valid = false;
        }
    }

    // (3) Passwort validieren.
    if (empty($_POST['pwd'])) {
        $pwdErr = "Passwort fehlt.";
        $valid = false;
    } else {
        $pwd = test_input($_POST["pwd"]);
    }

    // (4) Datenbank validieren.
    if(!empty($_POST['database'])) {
        $database = test_input($_POST["database"]);
    }

    // (5) Tage validieren.
    $days  = test_input($_POST['days']);
    $days  = (empty($days) || !is_numeric($days)) ? "" : $days;
    $isChecked = isset($_POST['c_box']) ? true : false;
    if ($isChecked) {
        if ($days == "") {
            $daysErr = "Tage ungültig.";
            $valid = false;
        }
    } else {
        if ($days > 0) {
            $daysErr = "Tage unnötig.";
            $valid = false;
        }
    }

    // (6) Datenbankauswahl validieren.
    if(empty($_POST['backupall']) ) {
        $backupallErr = "Keine Sicherungsoption ausgewählt!";
        $valid = false;
    } else {
        $backupall = $_POST['backupall']; // $backupall ist eine Zeichenkette!
        if ($backupall == 'true') {
            if (!empty($_POST['database'])) {
                $backupallErr = "Ausgewählt wurde, alle Datenbanken zu sichern. Aber ein besonderer
                    Datenbankname wurde gefunden.";
                $valid = false;
            }
        } else {
            if (empty($_POST['database'])) {
                $backupallErr = "Ausgewählt wurde, eine bestimmte Datenbank zu sichern. Aber ein
                    Datenbankname wurde nicht gefunden.";
                $valid = false;
            }
        }
    }
}
```

Listing 7: PHP-Skript, 2. Teil: Validierung der Formulareingaben

```
<center><h1>Datenbank(en) sichern</h1></center>
<p><em><strong>Hinweis:</strong></em> Sobald auf 'Sicherung erstellen' geklickt wird,
       kann der Vorgang mehrere Sekunden dauern. Bitte etwas Geduld!</p>
<center><p><span class="error"><sup>*) </sup>Pflichtfeld.</span></p></center>
<form name="dbbackup" method="post" action="<?php echo htmlentities(THISPAGE,ENT_QUOTES,'UTF-8');?>"
    accept-charset="utf-8">
    <fieldset>
        <legend><b>Sicherungsort</b></legend>
        <label>Die folgenden 4 Felder sind schreibgeschützt!</label>
        <br>
        <label for "root_fldr" class="lbl">Wurzelverzeichnis:</label>
        <input type="text" name="root_fldr" value="<?php echo BACKUPDIR;?>" readonly />
        <br>
        <label for "sql_prefix" class="lbl">Dateivorsatz:</label>
        <input type="text" name="sql_prefix" value="<?php echo PREFIX;?>" readonly />
        <br>
        <label for "sql_date" class="lbl">Dateidatum:</label>
        <input type="text" name="sql_date" value="<?php echo date('Y-m-d');?>" readonly />
        (= aktuelles Tagesdatum)
        <br>
        <label for "sql_ext" class="lbl">Dateierweiterung:</label>
        <input type="text" name="sql_ext" value="<?php echo EXT;?>" readonly />
        <br>
    </fieldset><br>

    <fieldset>
        <legend><b>Anmeldung</b></legend>
        <?php
        $users = getUsers(HOST, USR, PWD, DB);
        ?>
        <label for "username" class="lbl">Benutzername: </label>
        <select name="username" />
        <option selected disabled>Benutzer wählen</option>
        <?php
        foreach ($users as $usr) {
            $sel = ($usr==$username) ? " selected" : "";
            echo '<option value = '.$usr.$sel.'>'.$usr.'</option>';
        }
        echo "</select>";
        ?>
        <span class="error"><sup>*) </sup><?php echo $usernameErr;?></span>
        <br>
        <label for "pwd" class="lbl">Passwort:</label>
        <input type='password' name='pwd' placeholder = 'Passwort'
            value='<?php echo isset($_POST['pwd']) ? $_POST['pwd'] : ''; ?>' />
        <span class="error"><sup>*) </sup><?php echo $pwdErr;?></span>
        <br>
    </fieldset><br>

    <fieldset>
        <legend><b>Obsolete SQL-Dateien entfernen?</b></legend>
        <input type="checkbox" name="c_box" value="" <?php if(isset($_POST['c_box'])) echo
            "checked='checked'";?> />
        <label for "c_box">Sicherungen löschen, die älter als</label>
        <input type="text" name="days" maxlength="2" placeholder="Tag(e)" value="<?php echo $days ?>" />
        <label>Tag(e) sind.</label>
        <span class="error"><?php echo $daysErr;?></span>
        <br>
    </fieldset><br>
```

Listing 8: Fortsetzung auf der nächsten Seite

```
    <fieldset>
        <legend><b>Was sichern?</b></legend>
        <span> </span>
        <span class="error"><sup>*) </sup><?php echo $backupallErr;?></span>
        <br>
        <input type="radio" name="backupall" value="true" id="backupallTrue"
            <?php echo (isset($_POST['backupall'])
                && $_POST['backupall']=="true") ? 'checked' : '';?> />
        <label for="backupallTrue">Alle Datenbanken sichern</label>
        <br>
        <input type="radio" name="backupall" value="false" id="backupallFalse"
            <?php echo (isset($_POST['backupall'])
                && $_POST['backupall']=="false") ? 'checked' : '';?> />
        <label for="backupallFalse">Bestimmte Datenbank sichern:</label>
        <?php
        //  Datenbanken ermitteln.
        $dbs= getDatabases(HOST, USR, PWD);
        //  Datenbanken zählen.
        $cnt_dbs = count($dbs);
        // Datenbanken zur Auswahl anzeigen.
        echo '<select name="database" id="database">';
        echo'<option selected disabled>Datenbank wählen</option>';
        echo '<option value =" " ></option>';
        for($i=0; $i < $cnt_dbs; $i++) {
            $db = $dbs[$i];
            $sel = ($database==$db) ? " selected" : "";
            echo '<option value = '.$db.$sel.'>'.$db.'</option>';
        }
        echo "</select><br>";
        ?>
    </fieldset><br>

    // Schaltflächen zentriert einrichten
    <center>
        <input type="submit" name="submit" value="Sicherung erstellen">  
        <input type="reset" value="Formularwerte zurücksetzen"> 
        <input type="button" name="btn" value="Formular verlassen" onclick="self.close()">
    </center>
    <br>
</form>
```

Listing 8: PHP-Skript, 3. Teil: HTML-Formular mit PHP-Skript einrichten

2.5 Formulardaten ausgeben und auswerten

Der vierte und letzte Teil des PHP-Skripts beinhaltet die Verarbeitung der zuvor validierten Formulardaten. Listing 9 enthält alle Einzelheiten:

```
<?php
if((isset($_POST['submit'])) && ($valid == true)) { // Formular abgeschickt und Formulardaten gültig?
    // Veraltete SQL-Dateien entfernen.
    if ($isChecked && ($days > 0)) {
        $cnt = delete_obsolete_sql_files($days, BACKUPDIR, EXT);
        echo "Zahl der gelöschten SQL-Dateien, die älter als ".$days." Tage sind: ".$cnt;
    }
    // Datenbank(en) sichern (nein/ja)?
    $backupall = ($backupall == 'false') ? false : true;
    //  Wenn alle Datenbanken gesichert werden sollen,
    //  wird die Option -A (steht für 'all-databases') gesetzt,
    //  sonst wird der Name der zu sichernden, speziellen Datenbank (d.h. $database) gesetzt.
    $dbargument = ($backupall == true) ? '-A' : $database;
    // Voller Name der aktuellen SQL-Datei.
    $backupfile = BACKUPDIR.PREFIX.$sql_date.EXT;
    // Aktuelle SQL-Datei entfernen.
    if(file_exists($backupfile)) {
        unlink($backupfile);
    }
    //  Die Dateideskriptoren für stdin, stdout und stderr lauten 0, 1 bzw. 2.
    //  Die Umleitung von stderr nach stdout erfolgt durch den Ausdruck: 2>&1
    $cmd = "mysqldump ".$dbargument." -u ".$username." -p".$pwd." -r \"".$backupfile."\" 2>&1";
    ?>
    <h2>Sicherung läuft, bitte warten ...</h2>
    <?php
    //  Sicherung der Datenbank(en) durchführen.
    exec($cmd);
    // Existenzprüfung der Sicherungsdatei.
    if (!is_file($backupfile)) {
        die("Fehler: Die Zieldatei '$backupfile' wurde NICHT erstellt!<br>");
    } else if (filesize($backupfile) == 0) {
        die("Fehler: Die Zieldatei ". $backupfile. " enthält ".filesize($backupfile) . " Byte <br>");
    }
    ?>
    <h2>Sicherung erfolgreich durchgeführt.</h2>
    <!-- Rückkehr zum Eingabeformular -->
    <button onclick="location.href='<?php echo THISPAGE ?>'" type="button">Zurück zum Formular</button>
    <?php
    exit();
}
?>
```

Listing 9: PHP-Skript, 4. Teil: Formulardaten ausgeben und auswerten

3 Erläuterung wichtiger Details

3.1 Fehlerbehandlung und Variablen definieren

Am Anfang des <**body**>-Tag[4] wird festgelegt,

- wie Fehler des PHP-Skripts behandelt,
- Variablen definiert,
- und diese mit Anfangswerten belegt werden.

Listing 10 beinhaltet die Einzelheiten:

```
<body>
<?php
    // Fehlerbehandlung definieren
    error_reporting(E_ERROR | E_WARNING | E_PARSE);// Nur einfache Fehler melden.
    ini_set('display_errors', TRUE);               // Fehler angezeigen.
    // Variablen definieren und Anfangswerte setzen.
    define('THISPAGE',$_SERVER['PHP_SELF']);       // Link zu dieser Seite (Formularaktion etc.)
                                                   // $_SERVER['PHP_SELF'] ist eine super-globale Variable,
                                                   //     die den Dateinamen des aktuell ausgeführten PHP-
                                                   //     Skripts zurückgibt.
    define('HOST','localhost');                    // Name des lokalen Webservers.
    define('BACKUPDIR','F:/');                     // Speicherort der Sicherung (ggf. anpassen).
    define('PREFIX','backup-');                    // Präfix für den Namen einer SQL-Datei.
    define('EXT','.sql');                          // Dateizusatz für SQL-Dateien.
    define('USR','root');                          // Name des Standard-Benutzers.
    define('PWD','***');                           // Passwort des Standard-Benutzers (hier zur Sicherheit
                                                   //     durch 3 Sternchen ersetzt, bitte anpassen).
    define('DB','MySQL');                          // Name einer System-Datenbank.

    $root_fldr   =$sql_date   =$username   =$pwd   =$backupall   =$days="";
    $root_fldrErr=$sql_dateErr=$usernameErr=$pwdErr=$backupallErr=$daysErr="";
…
?>
</body>
```

Listing 10: Variablen und Zuweisungen im <body>-Tag der Anwendung

[4] Ein **<body>**-Tag markiert Anfang und Ende des sichtbaren Inhalts einer Webseite.

3.2 Obsolete SQL-Dateien entfernen?

Die dritte Feldgruppe des Eingabeformular (s. Abb. 1) bietet die Möglichkeit, veraltete SQL-Dateien im vorgegebenen Speicherort zu entfernen, wenn das jeweiliges Änderungsdatum mehr als X Kalendertage zurück liegt. Dazu muss das entsprechende Kontrollkästchen markiert und die Zahl der Kalendertage erfasst werden (s. Abb. 2).

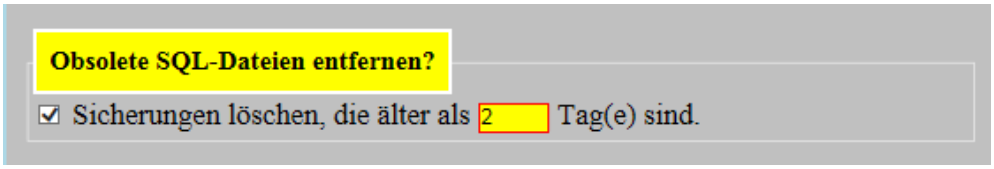

Abb. 2: Obsolete SQL-Dateien entfernen?

Angenommen,

- das Tagesdatum der aktuellen Sicherung ist der 04.12.2018,
- die Feldgruppe *Obsolete SQL-Dateien entfernen?* wurde wie vorstehend (s. Abb. 2) belegt,
- im vorgegebenen Speicherort (s. 1. Feldgrupp in Abb. 1) befinden sich bereits fünf SQL-Dateien mit dem jeweils dazugehörigen Änderungsdatum (s. Abb. 3).

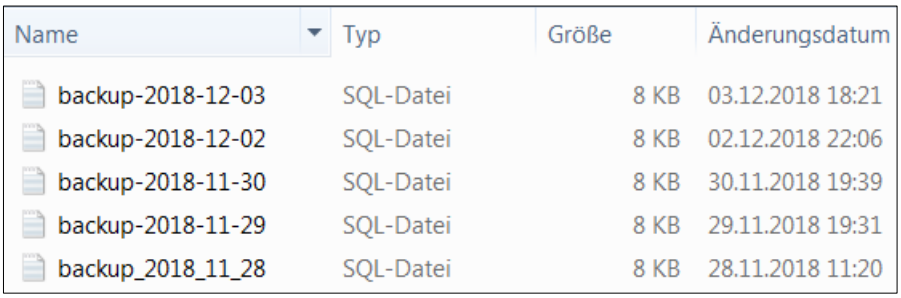

Abb. 3: Gespeicherte SQL-Dateien mit Änderungsdatum im Windows Explorer

Das zum Eingabeformular gehörige PHP-Skript würde in diesem Fall die Erfolgsmeldung ausgeben, dass 3 überalterte SQL-Dateien entfernt wurden (s. Abb. 4 in Verbindung mit Abb. 3).

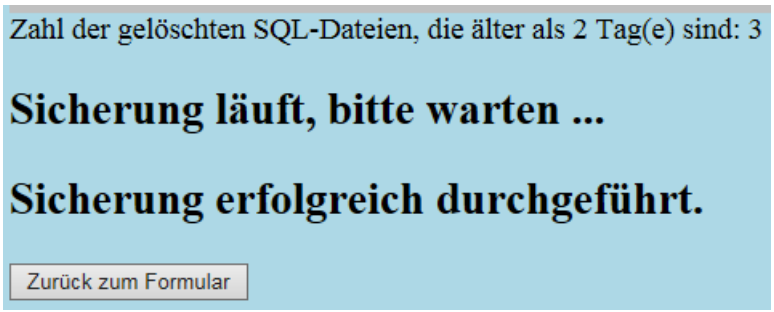

Abb. 4: Erfolgsmeldung des PHP-Skripts

3.3 Was sichern?

3.3.1 Alle Datenbanken sichern

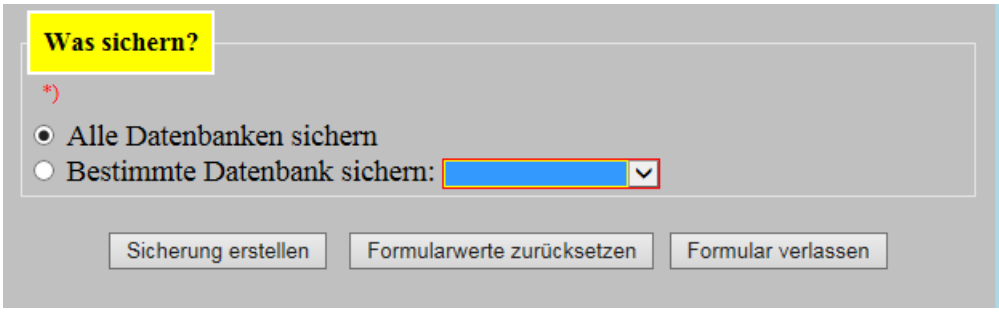

Abb. 5: Alle Datenbanken sichern

Wenn der Benutzer das Optionsfeld *„Alle Datenbanken sichern“* markiert, wird im PHP-Skript folgender Befehl erzeugt und ausgeführt:

mysqldump -A -u root –p* -r "F:/backup-2018-12-03.sql" 2>&1**

Die Kürzel in diesem Befehl bedeuten:

Kürzel	Bedeutung
mysqldump	Ein **Befehl**, der die logische Sicherung einer Kombination von Datenbanken, Tabellen und Tabellendaten ausführt. Das Ergebnis sind SQL-Anweisungen, welche die ursprünglichen Schemaobjekte, Daten oder beides wiedergeben.
-A	Filter: Steht für ***all-databases*** (alle Datenbanken).
-u	Option: Steht für ***user*** (Benutzername, der beim Herstellen einer Verbindung zum lokalen Web-Server verwendet werden soll.)
-p	Option: Steht für ***password*** (Passwort, das zum Herstellen einer Verbindung zum lokalen Web-Server verwendet werden soll. Oben durch drei Sternchen ersetzt.)
-r	Option: Steht für ***result-file*** (SQL-Zieldatei)
2>&1	Umleitung von ***stderr*** nach ***stdout***. Die Dateideskriptoren für *stdin*, *stdout* und *stderr* lauten 0, 1 bzw. 2.

Tabelle 3: Bedeutung der Kürzel im Befehl zur Sicherung aller Datenbanken

3.3.2 Bestimmte Datenbank sichern

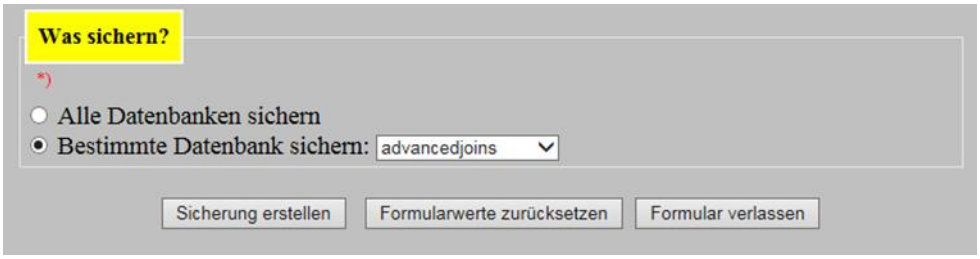

Abb. 6: Bestimmte Datenbank sichern

Wenn der Benutzer das Optionsfeld *„ Bestimmte Datenbank sichern“* markiert und dann den Namen der Datenbank *advancedjoins* aus dem dazugehörigen Kombinationsfeld auswählt, wird im PHP-Skript folgender Befehl erzeugt und ausgeführt:

mysqldump advancedjoins -u root –p* -r "F:/backup-2018-12-03.sql" 2>&1**

Die Kürzel in diesem Befehl sind ebenfalls in Tabelle 3 enthalten, lediglich die Option –A (*alle Datenbanken*) ist gegen den Namen der gewählten, speziellen Datenbank (hier: *advancedjoins*) ausgetauscht worden.

4 Alternative Lösungen

Wer nach einer alternativen Lösung zur Sicherung von MySQL-Datenbanken sucht, sei auf den englischsprachigen Beitrag *„10 Ways to Automatically & Manually Backup MySQL Database“* verwiesen [2]. Dort wird u. a. Schritt für Schritt beschrieben, wie das Werkzeug *PhpMyAdmin*[5] zum Sichern (d. h. ***Export***) und zum Wiederherstellen (d. h. ***Import***) von MySQL-Datenbanken genutzt werden kann.

5 Wiederherstellung

Wenn eine MySQL-Datenbank korrupt ist oder andere undefinierte Probleme bereitet ist meistens eine Wiederherstellung notwendig. Der Grundbefehl zur Rücksicherung einer bestehenden Datenbank lautet:

mysql -h{Hostname} -u{Benutzername} -p{Passwort} {Datenbankname} **<** {SQL-Datei}

Sind alle Datenbank mit der *"--all-databases*"-Option gesichert worden und soll daraus nur eine ganz bestimmte Datenbank zurückgesichert werden, kann das gewünschte Objekt mit Hilfe der *"--one-database*"-Option importiert werden.

Kurz, die Wiederherstellung einer bestehenden Datenbank kann bequem mit

- einem PHP-Skript (s. [3])
- dem Softwarewerkzeug *phpMyAdmin (s.* [4]),
- oder einem anderen Tool vorgenommen werden.

[5] *phpMyAdmin* ist eine freie Webanwendung zur Administration von MySQL-Datenbanken.

Literatur

[1] P. Müller, Einstieg in CSS, Bonn: Galileo Press, 2014.

[2] B. Schwartz, „10 Ways to Automatically & Manually Backup MySQL Database,“ 03 2009. [Online]. Available: https://www.jotform.com/blog/how-to-backup-mysql-database/. [Zugriff am 05 12 2018].

[3] o. V., „MySQL-Datenbank mittels PHP sichern und wiederherstellen,“ 2018. [Online]. Available: https://www.ionos.de/hilfe/hosting/mysql-datenbank-fuer-web-projekt-verwenden/mysql-datenbank-mittels-php-sichern-und-wiederherstellen/. [Zugriff am 05 12 2018].

[4] o. V., „MySQL 8.0 Reference Manual/... /mysqldump — A Database Backup Program,“ MySQL, [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html. [Zugriff am 05 12 2018].

[5] P. Upfold, „Do a MySQL backup from a PHP script,“ 21 01 2008. [Online]. Available: https://fosswire.com/post/2008/01/do-a-mysql-backup-from-a-php-script/. [Zugriff am 05 01 2018].