Elvis-Shop

Elvis betreibt einen Online-Shop. Wenn jemand etwas kauft, nimmt Elvis die E-Mail-Adresse des Kunden in seine Kartei auf. Diese verwendet er, um Rundschreiben mit seinen Sonderangeboten zu verschicken. Momentan muss Elvis die E-Mails in seiner Liste noch manuell durchgehen, kopieren und in die Empfängerliste seiner Werbemails einfügen. Das funktioniert, kostet aber eine Menge Zeit und Mühe. Daher soll dies mittels HTML und PHP automatisiert werden.

Ziel:

- **Datenbank automatisch mit E-Mail-Daten füllen, die aus einem HTML-Formular stammen**
- **Formular mit Werbetext füllen und dies als E-Mail an alle auf einmal versenden können, die in der Datenbank gespeichert sind, z.B. für Werbeaktionen (Spam)**
- **Kunden per Formular aus der Datenbank entfernen**
- **Validierung der Eingabe, inkl. Fehlermeldung**

Die zu erstellende Elvis-Webanwendung besteht aus folgenden Grundkomponenten:

- einem Formular, über das sich neue Kunden für die Mailing-Liste eintragen können
- einem zweiten Formular, über das E-Mails vom Administrator Elvis an die Adressen auf Elvis E-Mail-Liste versendet werden, und
- einem Formular zum Schluss, zum Entfernen von Kunden aus der Liste, wenn der Kunde dies möchte.

To do Liste:

1. Datenbank und Tabelle für die Mailing-Liste erstellen. Die Tabelle speichert die Vornamen, Nachnamen und E-Mail-Adressen aller Personen auf Elvis Mailing-Liste.
2. Ein Formular und ein PHP-Skript, inkl. eingebetteter SQL-Abfragen, für die Anmeldung zur Mailing-Liste schreiben. Hier erstellen wir ein Formular und ein Skript, über die Kunden ihre Vornamen, Nachnamen und E-Mail-Adressen angeben und der Mailing-Liste hinzufügen können. (**anmelden.html und anmelden.php)**
3. Ein Formular und ein PHP-Skript zum Versenden einer Mail an alle Interessenten schreiben. Als Letztes erstellen wir das Formular, über das Elvis eine E-Mail verfassen kann, und das Skript, das das Rundschreiben entgegennimmt und an alle verschickt, die sich für die Liste eingetragen haben. (**mailsenden.html und mailsenden.php**)
4. Kunden aus der Datenbank entfernen. Über ein HTML-Formular (kunde_raus.html) kann der Kunde aus dem Postverteiler gelöscht werden. (**kunde_raus.html und kunde_raus.php**).
5. Validierung (Überprüfung) der Mail-Funktion auf leere Felder – if-Anweisung, else

Verwendete Befehle: (SQL-Befehle in Großbuchstaben)

SQL: CREATE DATABASE, CREATE TABLE

PHP: $_POST, SELECT * FROM (eingebettete SQL-Abfragen)
new MySQLi("localhost", "root", "", "elvis_laden");
INSERT INTO (eingebettete SQL-Abfragen)
mysqli_query(); mysqli_fetch_array() ; while (Bedingung) { Aktion(); }
mail(); DELETE FROM (eingebettete SQL-Abfrage)
empty(); if(), else()

1.)Datenbank und Tabelle für die Mailing-Liste erstellen

Wenn wir eine Datenbank und eine Tabelle für Elvis Mailing-Liste erstellen wollen, müssen wir uns zunächst um die Datenbank elvis_laden kümmern, die die Tabelle email_liste aufnehmen wird. Beide werden wir mit SQL-Anweisungen erstellen. Die SQL-Anweisung zur Erstellung einer Datenbank ist CREATE DATABASE.

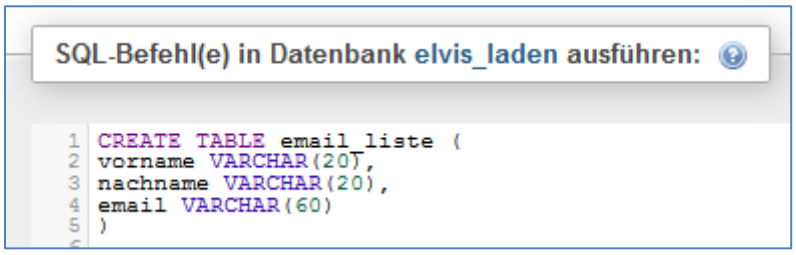
Bevor man eine Tabelle erstellen kann, muss man wissen, wie die Daten aussehen, die man in dieser Tabelle speichern will. Elvis möchte die Vor- und Nachnamen der Personen auf seiner Mailing- Liste einsetzen, um den Nachrichten, die er versendet, eine persönlichere Note zu geben. Fügt man dem noch die E-Mail-Adresse hinzu, muss Elvis Tabelle email_liste also für jeden Datensatz drei Datenteile speichern.
In einer Tabelle kommen alle Datenteile in eigene Spalten, die jeweils Namen tragen, die die enthaltenen Daten beschreiben. Nutzen wir vorname, nachname und email als Spaltennamen. Jede Zeile in der Tabelle enthält jeweils einen Datenteil für jede dieser Spalten. Eine Zeile bildet einen Eintrag in Elvis Mailing-Liste.

Nun muss man die Daten definieren und den **Datentyp** überlegen.
Es ist wichtig, dass man bei der Erstellung der Tabellenspalten geeignete Datentypen verwendet. Das macht Tabellen genauer und effizienter. Beispielsweise ist für die Speicherung von Textdaten mehr Platz erforderlich als für die Speicherung von ganzzahligen Daten. Werden in einer Spalte nur ganzzahlige Werte gespeichert, ist es also empfehlenswert, für sie einen ganzzahligen Datentyp zu wählen. Außerdem sorgt der Datenbankserver dafür, dass in eine Spalte nur Werte des gewählten Datentyps eingefügt werden. Soll eine Spalte per Tabellendefinition einen Datumsdatentyp speichern, erhält man dann entsprechend eine Fehlermeldung, wenn man versuchen, in diese Spalte etwas anderes als einen Datumswert einzufügen.

Erstellen der Tabelle

Mit der CREATE TABLE-Anweisung wird eine neue Tabelle in einer Datenbank erstellt. Unsere Datenbank ist die „elvis_laden“ Datenbank.

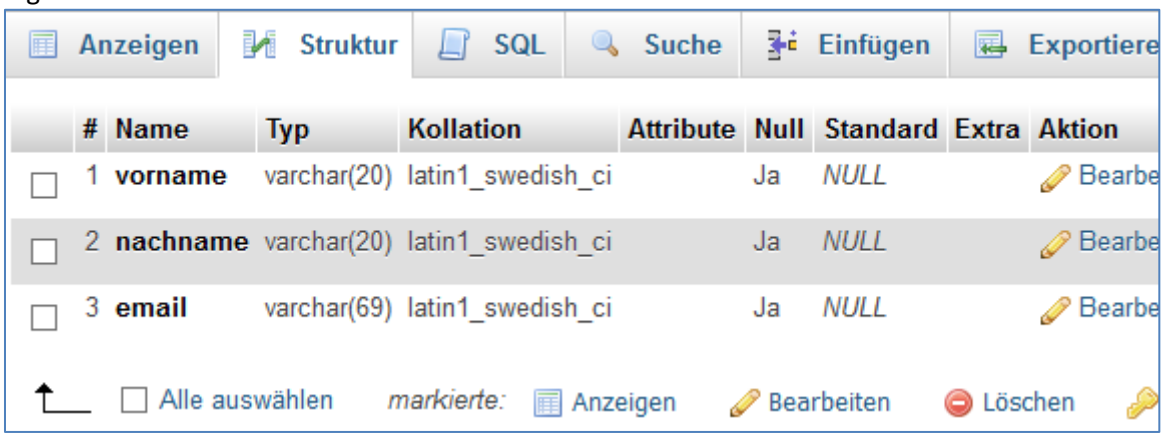
```
CREATE TABLE email_liste (
vorname VARCHAR(20),
nachname VARCHAR(20),
email VARCHAR(60)
)
```



Beachte:
Da bei der Erstellung der Datenbank bereits mittels „USE elvis_laden“ die Datenbank angesprochen wurde, gibt es nun keine Probleme. Ansonsten würde eine Fehlermeldung erzeugt, wenn man keine Datenbank angibt.

Wir haben hier keinen Primärschlüssel vergeben.

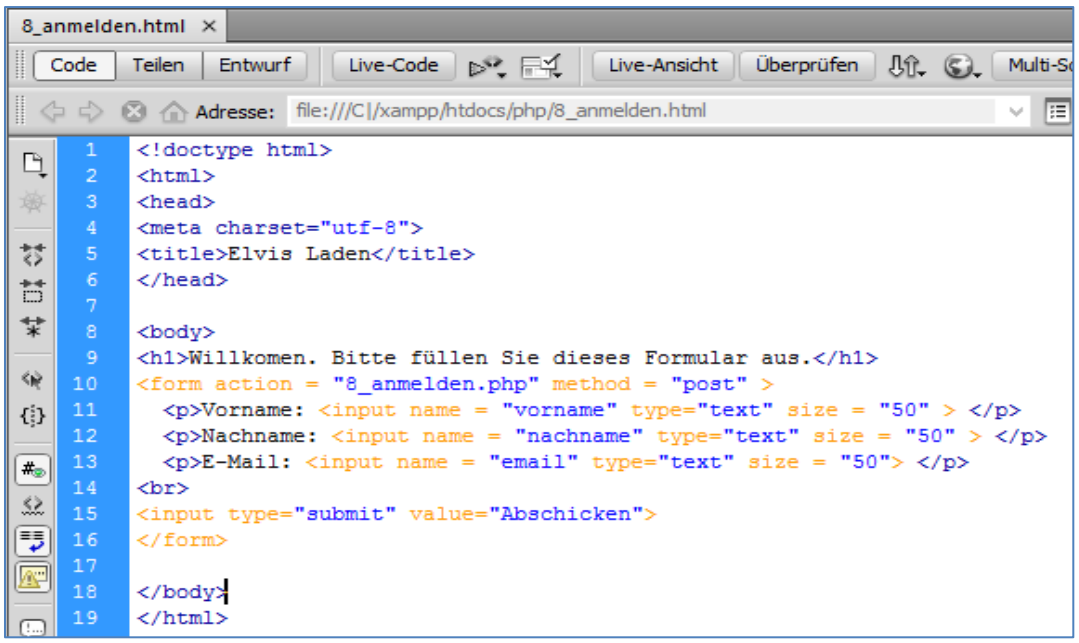
Ergebnis:



2)Das Anmeldeskript erstellen

Ein Formular und ein PHP-Skript für die Anmeldung zur Mailingliste schreiben.
Elvis braucht ein HTML-Formular, über das seine Kunden ihre Namen und E-Mail-Adressen angeben können. Hat er diese, kann er sie mit einem PHP-Skript in Empfang nehmen und in der Tabelle email_liste speichern. Das Webformular (anmelden.html) benötigt drei Eingabefelder und einen Button. Der Teil des Formulars, der am interessantesten ist, ist die Formularaktion, da sie dafür sorgt, dass die Formulardaten an das PHP-Skript übergeben werden, das wir gleich erstellen müssen.

a)Erstellen des Formulars in HTML (8_anmelden.html):



Code:

```
<body>
<h1>Willkommen. Bitte füllen Sie dieses Formular aus.</h1>
<form action = "8_anmelden.php" method = "post" >
 <p>Vorname: <input name = "vorname" type="text" size = "50" > </p>
 <p>Nachname: <input name = "nachname" type="text" size = "50" > </p>
 <p>E-Mail: <input name = "email" type="text" size = "50"> </p>
<br>
<input type="submit" value="Abschicken">
</form>
</body>
```

Ergebnis:

b)Erstellen der anmelden.php:

Das Skript anmelden.php verarbeitet die Daten aus dem Anmeldeformular. Das Skript sollte die vom Formular übergebenen Daten nehmen, eine Verbindung mit der Datenbank elvis_laden herstellen und die Daten mit INSERT in die Tabelle email_liste einfügen.

```php
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Elvis Laden</title>
</head>

<body>
<h1>Elvis Laden</h1>
<?php

$vorname = $_POST["vorname"];
$nachname = $_POST["nachname"];
$email = $_POST["email"];

/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "elvis_laden");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}

$sql = "INSERT INTO email_liste (vorname, nachname, email) VALUES ('$vorname', '$nachname', '$email')";

$ergebnis = mysqli_query($con, $sql)
    or die("Fehler bei der Datenbankabfrage.");

mysqli_close($con);

?>
</body>
</html>
```

Code:

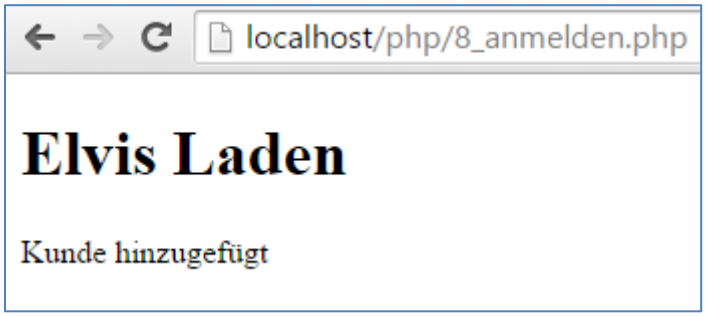
```
<body>
<h1>Elvis Laden</h1>
<?php
$vorname = $_POST["vorname"];
$nachname = $_POST["nachname"];
$email = $_POST["email"];
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "elvis_laden");
if ($con->connect_error) {
        echo "Fehler bei der Verbindung: " . mysqli_connect_error();
        exit();
}
$sql = "INSERT INTO email_liste (vorname, nachname, email) VALUES ('$vorname', '$nachname',
'$email')";
$ergebnis = mysqli_query($con, $sql)
        or die("Fehler bei der Datenbankabfrage.");
mysqli_close($con);
?>
</body>
```

Sinnvoll ist auch diese Bestätigung in „8_anmelden.php“:
Dass der neue Kunde der Mailing-Liste hinzugefügt wurde, wird von anmelden.php auf diese Weise bestätigt.

```
26      or die("Fehler bei der Daten
27
28  echo "Kunde hinzugefügt";
29
30  mysqli_close($con);
```

Füge diese hinzu!

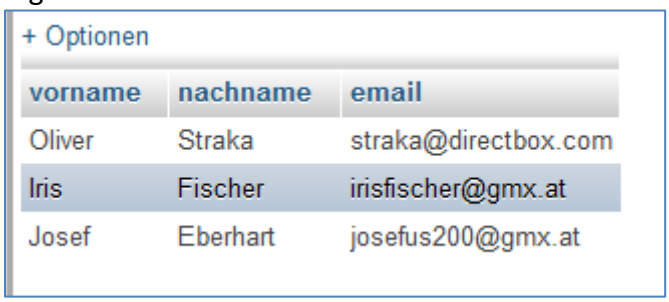
Ergebnis:



Testen:
Gib in das HTML-Formular „8_anmelden.html“ drei Kunden ein und kontrolliere in der Datenbank mittels phpMyAdmin deren Eintrag.

SELECT * FROM email_liste

Ergebnis:



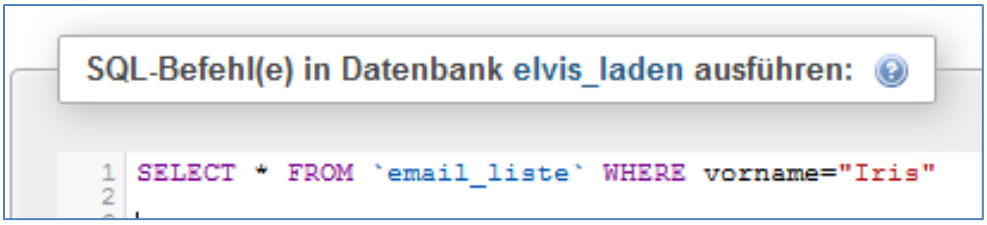
vorname	nachname	email
Oliver	Straka	straka@directbox.com
Iris	Fischer	irisfischer@gmx.at
Josef	Eberhart	josefus200@gmx.at

Übung: Abfragen durchführen

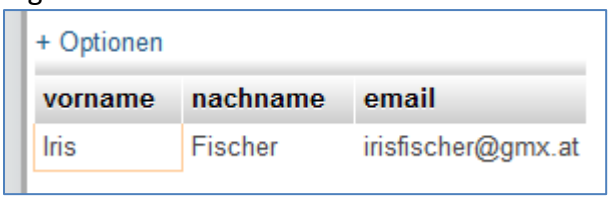
Die WHERE – Klausel schränkt die Abfrageergebnisse ein.

- Alle Daten der Kunden mit dem Vornamen Iris wählen:

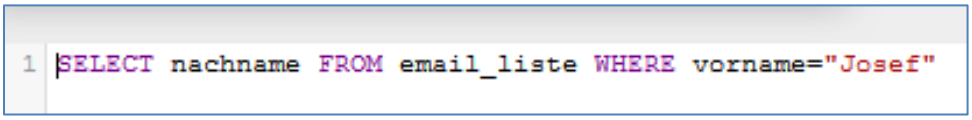
SELECT * FROM email_liste WHERE vorname=“Iris“



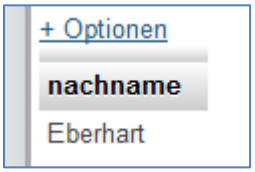
Ergebnis:



- Nur den Nachnamen aller Kunden mit dem Vornamen Josef wählen:

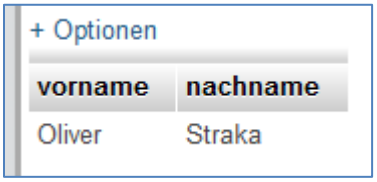


Ergebnis:



- Vor- und Nachname des Kunden mit der E-Mail-Adresse straka@directbox.com

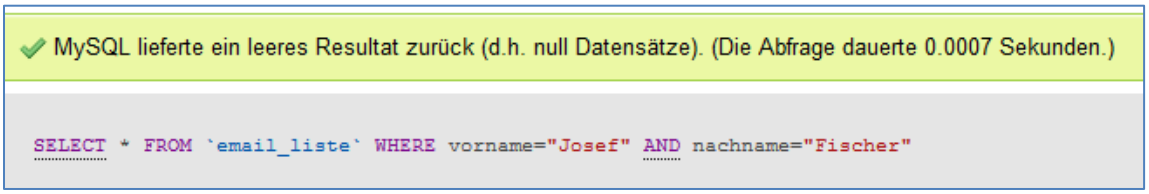
Ergebnis:



- Die Daten aller Kunden mit dem Vornamen Josef und dem Nachnamen Fischer wählen:

```
1 SELECT * FROM `email_liste` WHERE vorname="Josef" AND
nachname="Fischer"
```

Ergebnis:
ein leeres Ergebnis



Besser:
Vorname „Josef“ und Nachname „Eberhart“
Ergebnis:

3.)Ein Formular und ein PHP-Skript zum Versenden einer E-Mail an alle interessierten Kunden erstellen

Zum Unterschied von „anmelden.php“ muss man sich beim Versenden des Rundschreibens nicht nur mit einer Zeile der Tabelle „email_liste“ befassen, sondern mit dem gesamten Inhalt.

- mailsenden.php: liest die Kunden aus der Datenbank und sendet jedem das Rundschreiben.
- mailsenden.html: Hier ist ein Formular zu erstellen, das einen Betreff aufweist und den Inhalt des Schreibens abgibt und dann an alle auf der Liste versendet.

3a) mailsenden.php:

muss Daten aus zwei Quellen vereinen:
Einerseits die Namen und E-Mail-Adressen der Empfänger aus der Tabelle „email_liste“ und andererseits muss es mittels Formular den Betreff und Inhalt des Versenders Elvis entgegennehmen, die aus dem „mailsenden.html“ – Formular kommen.

Arbeitsschritte und Struktur:
a)Mithilfe von $_POST werden der Betreff und der Inhalt aus dem Formular übernommen.
b)SELECT holt die Daten aus der Tabelle „email_liste“
c)Die Ausführung der Abfrage gibt noch keinen Zugriff auf die Daten, daher muss man die einzelnen Datenzeilen im „ergebnis“ abrufen.
d)Erstellen von mysqli_fetch_array() und einer while-Schleife
e)mit der Funktion „mail()“ eine Mail an alle Kunden versenden. Dazu wird eine Schleife notwendig sein, die alle Datenzeilen durchläuft.
f)mysqli_close() schließen.

a)Wir nutzen einfach $_POST, um Betreff und Inhalt der Mail in Variablen zu speichern. Zusätzlich können wir vorsorglich schon einmal Elvis E-Mail-Adresse in einer Variablen speichern, weil wir diese später zum Versenden der E-Mails benötigen.

```php
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Mail senden</title>
</head>

<body>
<h1>Elvis Laden</h1>
<?php

$von = "josefeberhart@gmx.at";
$betreff = $_POST["betreff"];
$text = $_POST["elvismail"];

$msg = "Liebe(r) $vorname $nachname, \n $text";
$an =
mail($an, $betreff, $msg, 'FROM:' . $von);

echo "Gemailt an: " . $an . "<br />";


?>
</body>
</html>
```

$betreff: das Formularfeld in der „mailsenden.html“ heißt „betreff“
$text: der Inhalt der Nachricht in der „mailsenden.html“ wird in das Feld „elvismail“ eingegeben

Darunter wird gleich begonnen, die Versendung per **„mail()“** vorzubereiten:
$msg: der Inhalt der E-Mail wird aus dem Namen des Kunden und dem übermittelten Formular-Text zusammengebaut.
$vorname: Diese Variable wird später erzeugt
$nachname: Diese Variable wird später erzeugt
$an: wird später erstellt – wird aus der „email“-Spalte der Datenbank genommen werden
mail(): hier werden Empfänger, Betreff, Inhalt der Mail und Elvis E-Mail-Adresse an die Funktion mail() übergeben

echo: Es wird eine Bestätigungsmeldung in die Seite ausgegeben, die die E-Mail-Adressen aller Kunden enthält, an die die E-Mail versandt wurde. Anhand der Ausgabe des Skripts kann er sogar prüfen, ob die Nachrichten erfolgreich versandt wurden

Verbindung zur Datenbank herstellen:

Natürlich muss die Verbindung hergestellt werden. Für diese mittels Xampp verwendet man:

Als Datenbank wähle „elvis_laden“.

```
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "elvis_laden");
if ($con->connect_error) {
        echo "Fehler bei der Verbindung: " . mysqli_connect_error();
        exit();
}
```

```
12  $von = "josefeberhart@gmx.at";
13  $betreff = $_POST["betreff"];
14  $text = $_POST["elvismail"];
15
16  /* Verbindung aufnehmen*/
17  $con = new MySQLi("localhost", "root", "", "elvis_laden");
18  if ($con->connect_error) {
19      echo "Fehler bei der Verbindung: " . mysqli_connect_error();
20      exit();
21  }
```

b+c)SELECT:

Um die Tabellendaten aus email_liste in das Skript zu bekommen, benötigen wir eine SELECT-Abfrage. Bislang haben wir unsere SELECT-Anweisungen immer aus dem mysql-Client durchgeführt.
Dieses Mal werden wir das im Skript mailsenden.php machen, indem wir mit mysqli_query() eine SELECT-Abfrage durchführen.

Zuerst erzeuge die Variable, dann die echte mysqli-Abfrage:

$sql = „SELECT * FROM email_liste“;
$ergebnis = mysqli_query($con, $sql);

Die Variable $sql enthält die SQL-Abfrage in Form eines Text-Strings.
In dem wird die Abfrage geschrieben, die alle Spalten aus der Tabelle email-Liste auswählt.
Mysqli_query() führt die Abfrage $sql auf der erstellten Verbindung „$con“ aus.
In Klammer wird mit „$con“ die Datenbankverbindung gespeichert und dann die Variable $sql.

Füge noch die „or die()“ Funktion ein, die eine Meldung bei Problemen ausgibt:
Füge den Code ein:

```
16  /* Verbindung aufnehmen*/
17  $con = new MySQLi("localhost", "root", "", "elvis_laden");
18  if ($con->connect_error) {
19      echo "Fehler bei der Verbindung: " . mysqli_connect_error();
20      exit();
21  }
22  $sql = "SELECT * FROM email_liste";
23  $ergebnis = mysqli_query($con, $sql)
24    or die("Fehler bei der Datenbankabfrage.");
```

Beachte:

Wenn man versucht, den »Wert« der Variablen $ergebnis direkt auszugeben, erhält man eine Fehlermeldung, die besagt, dass ein Objekt der Klasse mysqli_result nicht in einen String umgewandelt werden konnte. Die Variable $ergebnis speichert also nur eine Referenz auf ein Objekt. MySQL liefert auf eine mysqli_query()-Abfrage dieser Form zwar die Ergebnismenge an den Client, PHP, zurück, aber diese wird von einem Objekt gekapselt. Wenn wir auf sie zugreifen wollen, müssen wir diese Referenz an eine Funktion wie **mysqli_fetch_array()** übergeben, die uns bei jedem Aufruf die aktuelle Datenzeile zurückliefert.

d)mysqli_fetch_array()

Der Variable $ergebnis wird als Parameter an die Funktion mysqli_fetch_array() übergeben, um die Ergebnismenge zeilenweise auszulesen. Die Datenzeilen werden als Array zurückgeliefert, das wir in einer neuen Variablen namens $zeile speichern.

$zeile = mysqli_fetch_array($ergebnis);

- Der Variablen $zeile wird das Array zugewiesen, das die Funktion aus der Ergebnismenge zurückliefert.
- Fetch_array(): Die Funktion liest die aktuelle Zeile der Ergebnismenge und liefert sie als Array zurück.
- $ergebnis: Das Abfrageergebnis wird als Objekt zurückgeliefert und die Referenz darauf haben wir der Variablen $ergebnis zugewiesen.

Gibt man diese mittels echo aus so ist das Ergebnis ist also eine Datenzeile wie z.B. Julian Kraml: juliankraml@gmx.at

echo $zeile[“vorname“] . “ “ . $zeile[“nachname“] . “ : “ . $zeile[“email“] . “
“;

Der Zeilenumbruch sorgt dafür, dass jede Datenzeile auf einer eigenen Zeile steht.
Jedes Mal, wenn dieser Code vom Webserver ausgeführt wird, wird eine neue Zeile aus dem Abfrageergebnis zurückgeliefert. Dabei werden die Spalten der Zeile jeweils als Elemente des Arrays gespeichert, das $zeile zugewiesen wird. Die ersten drei Aufrufe von mysqli_fetch_array() liefern also die Werte der ersten drei Zeilen der Ergebnismenge zurück.

Um alle Datenzeilen der Liste sinnvoll ausgeben zu können, benötigt man eine Schleife.

Ansonsten müsste man diese zwei Codezeilen permanent wiederholen:

```
$zeile = mysqli_fetch_array($ergebnis);
echo $zeile['vorname'] . ' ' . $zeile['nachname'] . ' : ' . $zeile['email'] . '<br />';
$zeile = mysqli_fetch_array($ergebnis);
echo $zeile['vorname'] . ' ' . $zeile['nachname'] . ' : ' . $zeile['email'] . '<br />';
$zeile = mysqli_fetch_array($ergebnis);
echo $zeile['vorname'] . ' ' . $zeile['nachname'] . ' : ' . $zeile['email'] . '<br />'; .....
```

Schleifen

Die while-Schleife ist eine Schleife, die Code wiederholt, **solange eine bestimmte Bedingung erfüllt ist.**

```
while (Bedingung) {
Aktion();
}
```

Die Bedingung (Code in der Klammer) wird geprüft. Sie wird durch einen Ausdruck angegeben, dessen Wert als ein Wahr/Falsch-Wert betrachtet werden kann. Wird er mit Wahr, d.h. true, ausgewertet, wird die Schleife ausgeführt. Wird er mit Falsch, also false, ausgewertet, wird die Schleife verlassen.
Die Schleifenaktion Aktion() wird bei jedem Schleifendurchlauf einmal ausgeführt.

Daten mit while durchlaufen:
In einer while-Schleife können wir mysqli_fetch_array() so einsetzen, dass alle Zeilen der Ergebnismenge nacheinander durchlaufen werden, bis ihr Ende erreicht wird.

```
while($zeile = mysqli_fetch_array($ergebnis)) {
        echo $zeile["vorname"] . " " . $zeile["nachname"] . ": " . $zeile["email"] . " <br />";
        }
```

Die while-Schleife durchläuft zeilenweise die Ergebnismenge. Hat sie das Ende der Daten erreicht, bricht sie ab.

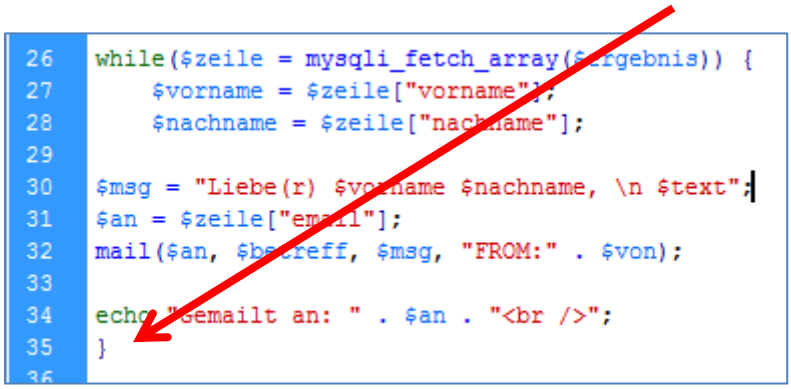
Beachte:
Da nun aber die einzelnen Zeilen ja nicht ausgegeben werden sollen, sondern nur für das Mail übergeben werden sollen, brauchen wir das „echo“ nicht. Stattdessen sollen nur die Vornamen und Nachnamen als Variable erstellt werden. Diese sollen dann im Mail verwendet werden:

```
while($zeile = mysqli_fetch_array($ergebnis)) {
        $vorname =  $zeile["vorname"];
        $nachname =  $zeile["nachname"];
        }
```

Füge diesen Code ein:

```
22  $sql = "SELECT * FROM email_liste";
23  $ergebnis = mysqli_query($con, $sql)
24    or die("Fehler bei der Datenbankabfrage.");
25
26  while($zeile = mysqli_fetch_array($ergebnis)) {
27      $vorname = $zeile["vorname"];
28      $nachname = $zeile["nachname"];
29      }
30
31  $msg = "Liebe(r) $vorname $nachname, \n $text";
```

Beachte:
Da auch das Mail und echo jedes Mal bei einem neuen Kunden versendet werden soll, muss die geschwungenen Klammer der while-Funktion **erst nach dem echo enden**.



```
26  while($zeile = mysqli_fetch_array($ergebnis)) {
27      $vorname = $zeile["vorname"];
28      $nachname = $zeile["nachname"];
29
30  $msg = "Liebe(r) $vorname $nachname, \n $text";
31  $an = $zeile["email"];
32  mail($an, $betreff, $msg, "FROM:" . $von);
33
34  echo "Gemailt an: " . $an . "<br />";
35  }
```

e)mail() vervollständigen
Nachdem die while-Schleife fertig ist und somit $vorname und $nachname zur Verfügung stellen kann man nun auch den Empfänger „$an“ ausfüllen:

$an = $zeile["email"];

```
31  $msg = "Liebe(r) $vorname $nachname, \n $text";
32  $an = $zeile["email"];
33  mail($an, $betreff, $msg, 'FROM:' . $von);
34
35  echo "Gemailt an: " . $an . "<br />";
```

f)Schließen von mysqli

mysqli_close($con);

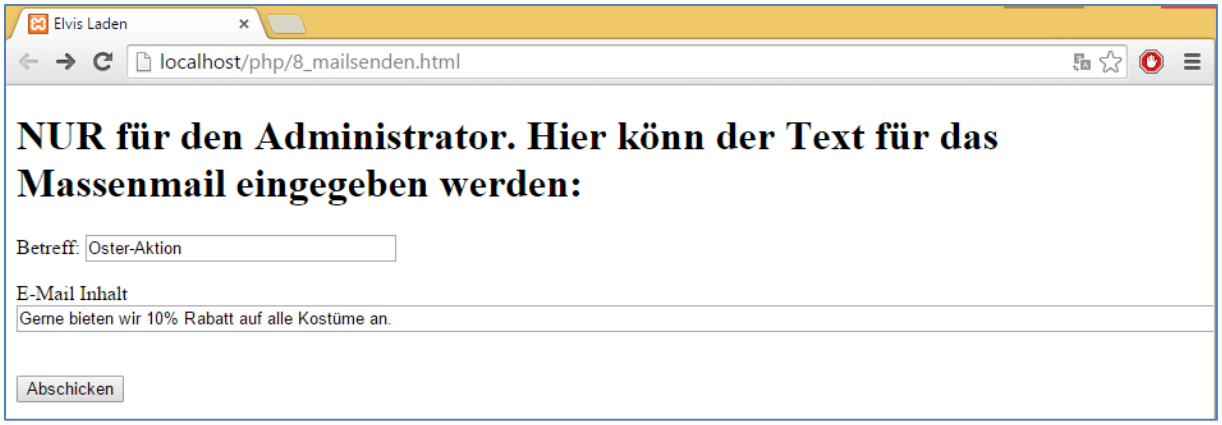
```
31  $msg = "Liebe(r) $vorname $nachname, \n $text";
32  $an = $zeile["email"];
33  mail($an, $betreff, $msg, 'FROM:' . $von);
34
35  echo "Gemailt an: " . $an . "<br />";
36
37  mysqli_close($con);
38  ?>
39  </body>
40  </html>
```

3b)HTML-Formular „mailsenden.html“

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Elvis Laden</title>
</head>
<body>
<h1>NUR für den Administrator. Hier könn der Text für das Massenmail eingegeben
werden:</h1>
<form action = "8_mailsenden.php" method = "post" >
  <p>Betreff: <input name = "betreff" type="text" size="30" ></p>
  <p>E-Mail Inhalt <input name = "elvismail" type="text" size = "400" > </p>
<br>
<input type="submit" value="Abschicken">
</form>
</body>
</html>
```

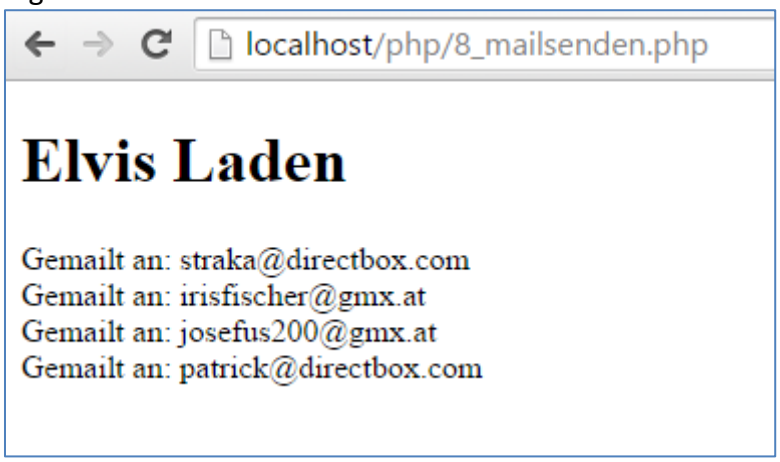
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Elvis Laden</title>
</head>

<body>
<h1>NUR für den Administrator. Hier könn der Text für das Massenmail eingegeben werden:</h1>
<form action = "8_mailsenden.php" method = "post" >
  <p>Betreff: <input name = "betreff" type="text" size="30" ></p>
  <p>E-Mail Inhalt <input name = "elvismail" type="text" size = "400" > </p>
<br>
<input type="submit" value="Abschicken">
</form>

</body>
</html>
```

Ergebnis:



Testen:

Ergebnis:



Aufgaben:

- Ändere das HTML-Formular so ab, dass auch die Postleitzahl und der Ort (ohne Straße und Hausnummer) eingegeben werden muss.
- Natürlich soll die PLZ und der Ort auch als echo ausgegeben werden. Dies in der Form wie hier angeführt:
  Gemailt an: irisfischer@gmx.at aus 2130 Mistelbach

4.)Kunden aus der Datenbank entfernen

Wenn wir aus einer Tabelle Daten löschen wollen, brauchen wir eine neue SQL-Anweisung, DELETE. Diese werden wir in einem neuen Skript einsetzen, um die Daten eines Kunden aus Elmars Postverteiler zu entfernen. Wir brauchen also mal wieder ein neues Skript und eine neue HTML-Seite.
DELETE entfernt eine Datenzeile aus einer Tabelle. Es ist also eine Anweisung, die man mit äußerster Vorsicht einsetzen sollte, da sie sehr schnell sämtliche Daten aus einer Tabelle löschen kann.

DELETE FROM Tabellenname

Um genau die Zeile oder die Zeilen anzugeben, die mit DELETE gelöscht werden sollen, muss man ans Ende eine WHERE-Klausel anhängen.

Beispiele:
DELETE FROM email_liste WHERE nachname = “Berger“;
DELETE FROM email_liste WHERE email = “franzberger@gmx.at“;

Da es mehrere Kunden mit denselben Vornamen und denselben Nachnamen geben kann, ist es besser, die eindeutige E-Mail Adresse zur Bereinigung mittels DELETE heranzuziehen.

Um nicht in der Datenbank mittels phpMyAdmin die Datensätze löschen zu müssen ist es sinnvoller eine webbasierte Schnittstelle mittels Formular zu verwenden. Erstelle folgende HTML-Seite (kunde_raus.html), welche mit der kunde_raus.php Seite verknüpft ist. Diese PHP-Seite nimmt die Löschung vor.

Aufgabe:
Erstelle eine Datei „Kunde_raus.html“ als Formular, die auf die „kunde_raus.php“ verlinkt und folgendes Aussehen haben soll:
Ein Klick auf den Abschicken-Button sendet das Formular als POST-Anfrage an das PHP-Skript.

Ergebnis:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Elvis Laden - E-Mail löschen</title>
</head>

<body>
<h1>Bitte geben Sie die zu löschende E-Mail Adresse ein:</h1>
<form action = "8_kunde_raus.php" method = "post" >
  <p>E-Mail Adresse <input name = "email" type="text" size = "60" > </p>
<br>
<input type="submit" value="Abschicken">
</form>

</body>
</html>

```

Die „kunde_raus.php“:

Am Beginn muss die Verbindung zur Datenbank erstellt werden und danach die Variable, die wir öfters benötigen werden:

```
<title>Kundde löschen</title>
</head>

<body>
<h1>Elvis Laden - Kunde löschen</h1>
<?php

/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "elvis_laden");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}

$email = $_POST["email"];

```

Die eingebettete SQL-Abfrage DELETE FROM ist dann zu erstellen.
Beachte beim Vergleich auf die einfachen und dann das doppelte Anführungszeichen!
Darunter kann eine Bestätigungsausgabe nicht schaden.

```

$sql = "DELETE FROM email_liste WHERE email = '$email'";
mysqli_query($con, $sql)
  or die("Fehler bei der Datenbankabfrage.");

echo "Kunde gelöscht:" . $email;

```

Schließe die Verbindung mit

```
mysqli_close($con);
?>
```

Code:

8_kunde_raus.html | 8_kunde_raus.php*

Code | Teilen | Entwurf | Live-Code | Live-Ansicht | Überprüfen

Adresse: file:///C|/xampp/htdocs/php/8_kunde_raus.php

1 / 2 Keine Syntaxfehler.

```php
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Kundde löschen</title>
</head>

<body>
<h1>Elvis Laden - Kunde löschen</h1>
<?php

/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "elvis_laden");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}

$email = $_POST["email"];

$sql = "DELETE FROM email_liste WHERE email = '$email'";
mysqli_query($con, $sql)
  or die("Fehler bei der Datenbankabfrage.");

echo "Kunde gelöscht:" . $email;

mysqli_close($con);
?>
</body>
</html>
```

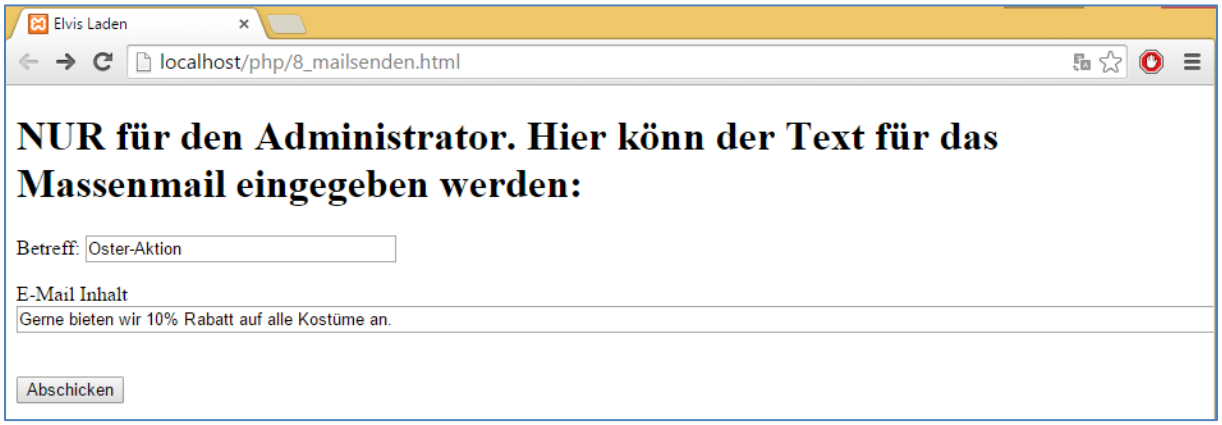
Teste dieses Formular und PHP-Skript.

5.)Validierung – Überprüfung von Feldern

Es kann aber vorkommen, dass Kunden leere E-Mails erhalten oder das gleiche E-Mail doppelt. Das eigentliche Problem hier ist also der »**Anwenderfehler**« – Elvis klickt unüberlegt auf Senden, obwohl er **keinen E-Mail-Inhalt eingegeben** hat, und veranlasst so, dass leere E-Mails an die gesamte Liste geschickt werden.
Daher muss man die E-Mail Funktion „mail()“ genauer ansehen.

a)Das Formular beinhaltet eine „Betreff“ (=$betreff) und ein „Inhalt“ (=$text) Feld.

mailsenen.html:



b)Hier sollte man den Code in der betreffenden „mailsenden.php“ überprüfen, ob sie nicht versehentlich leer sind.

```
10  <?php
11
12  $von = "josefeberhart@gmx.at";
13  $betreff = $_POST["betreff"];
14  $text = $_POST["elvismail"];
15
16  /* Verbindung aufnehmen*/
17  $con = new MySQLi("localhost", "root", "", "elvis_laden");
18  if ($con->connect_error) {
19      echo "Fehler bei der Verbindung: " . mysqli_connect_error();
20      exit();
21  }
22  $sql = "SELECT * FROM email_liste";
23  $ergebnis = mysqli_query($con, $sql)
24    or die("Fehler bei der Datenbankabfrage.");
25
26  while($zeile = mysqli_fetch_array($ergebnis)) {
27      $vorname = $zeile["vorname"];
28      $nachname = $zeile["nachname"];
29
30  $msg = "Liebe(r) $vorname $nachname, \n $text";
31  $an = $zeile["email"];
32  mail($an, $betreff, $msg, "FROM:" . $von);
33
34  echo "Gemailt an: " . $an . "<br />";
35  }
```

Das Problem ist, dass man $text und $betreff einfach verwenden, egal ob die beiden Variablen Text enthalten oder nicht.

Als Validierung bezeichnet man die Prüfung der Sauberkeit von Formulardaten, bevor diese zu etwas verwendet werden.

Beispiel:
Elvis nutzt bereits einen Validierungsschritt, ohne dafür diesen Namen zu verwenden. Wenn er eine Bestellung für Elvis-Waren erhält, verschickt er nicht sofort die Waren, sondern prüft erst die Bestellung! Bei einer Bestellung prüft Elmar, ob die Kreditkarte des Kunden **gültig** ist. Ist das der Fall, erledigt er die Bestellung und macht die Ware für den Versand fertig. Aber bevor er sie versenden kann, muss er noch prüfen, ob die Lieferadresse vollständig ist. Erst wenn auch diese Prüfung bestanden ist, verschickt er die Ware. Die erfolgreiche Ausführung einer Bestellung ist immer von einer Prüfung der Bestellungsdaten abhängig.

In „mailsenden.php“ dürfen daher die Mails erst versendet werden, wenn sichergestellt ist, dass die Felder validiert sind und daher sauber sind, d.h. keine leeren Inhalt haben.

Um die Validierung durchzuführen benötigt man eine IF-Anweisung:

```
if (istGueltig($kreditkarte)) {
  bestellungAusführen();
  }
```

1. **Das Schlüsselwort if**
   Mit diesem Schlüsselwort wird die Anweisung eingeleitet.

2. **Die Testbedingung**
   Die Testbedingung oder der **Bedingungsausdruck** folgt in **Klammern** unmittelbar auf das Schlüsselwort if. An dieser Stelle gibt man den Ausdruck an, dessen Gültigkeit oder Wahrheit man prüfen will.

3. **Die Aktion**
   Die Aktion einer if-Anweisung folgt in **geschweiften Klammern** auf die Testbedingung. Hier gibt man den PHP-Code an, der ausgeführt werden soll, wenn die Bedingung tatsächlich true ist.

Prüfung mit empty()

Man kann leere Strings mit == prüfen (=auf Gleichheit), PHP bietet allerdings eingebaute Funktionen, die dazu besser geeignet sind. Die Funktion **empty()** prüft, ob eine Variable einen **leeren Wert** enthält. Leere Werte in PHP sind 0, der leere String (' ' oder " ") sowie die Werte false oder NULL.
Beispiel:
empty() liefert true, wenn eine Variable auf 0, einen leeren String, false oder NULL gesetzt wurde.

Den Variablen $betreff und $text werden Werte aus $_POST['betreff'] und $_POST['elvismail'] zugewiesen. empty() prüft tatsächlich, ob eine Variable leer ist, und genau das benötigen wir für die Validierung des Formulars.

```
13  $betreff = $_POST["betreff"];
14  $text = $_POST["elvismail"];
```

- Die Prüfung bezieht sich auf zwei Formularfelder, nämlich auf $betreff und $text. Beide Felder müssen Daten enthalten:

WENN $betreff enthält Text UND $body enthält Test
DANN E-Mail senden

- Oder man kann auch umgekehrt schauen, ob eins der Felder leer ist

WENN $betreff ist leer UND $body ist leer
DANN Fehler melden

Eine bessere Lösung wäre die Vergleiche mit AND und OR zu verknüpfen:.

UND-Operator: && bzw. AND
ODER-Operator: II bzw. OR

```
if (empty($berteff)) && (empty($text)) {
    Dann Fehler melden
}
```

Das würde bedeuten, dass wenn beide Variablen leer sind, dieser Code mit „Datenbank verbinden und mail()“ nicht angewendet werden würde. Das ist zwar ok, besser wäre das Gegenteil: Wenn beide nicht leer sind dann soll ganz normal das Mail abgesendet werden.

Die Logik einer Testbedingung kann umgekehrt werden – mit dem **Negativoperator**:

Dieser wandelt ein „true“ in „false“ um und umgekehrt. Setze dafür ein Rufzeichen (als Kurzschreibweise) vor den Operator bzw. verwende NOT. Es dient der Negation (Verneinung) einer bestimmten Bedingung.

```
if (!empty($berteff)) && (!empty($text)) {
    …
}
```

d.h. Wenn die beiden Felder NICHT LEER sind, dann soll er ganz normal die Mails absenden:

Beachte:
Die geschweifte Klammer der if-Anweisung umklammert den ganzen Teil der Datenbank-Verbindung bis zur mail()-Funktion und dem Schließen der Datenbank-Verbindung:

```
<body>
<h1>Elvis Laden</h1>
<?php

$von = "josefeberhart@gmx.at";
$betreff = $_POST["betreff"];
$text = $_POST["elvismail"];

if((!empty($betreff)) && (!empty($text)))  {

/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "elvis_laden");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}
$sql = "SELECT * FROM email_liste";
$ergebnis = mysqli_query($con, $sql)
  or die("Fehler bei der Datenbankabfrage.");

while($zeile = mysqli_fetch_array($ergebnis)) {
    $vorname = $zeile["vorname"];
    $nachname = $zeile["nachname"];

$msg = "Liebe(r) $vorname $nachname, \n $text";
$an = $zeile["email"];
mail($an, $betreff, $msg, "FROM:" . $von);

echo "Gemailt an: " . $an . "<br />";
}

mysqli_close($con);

}

?>
</body>
```

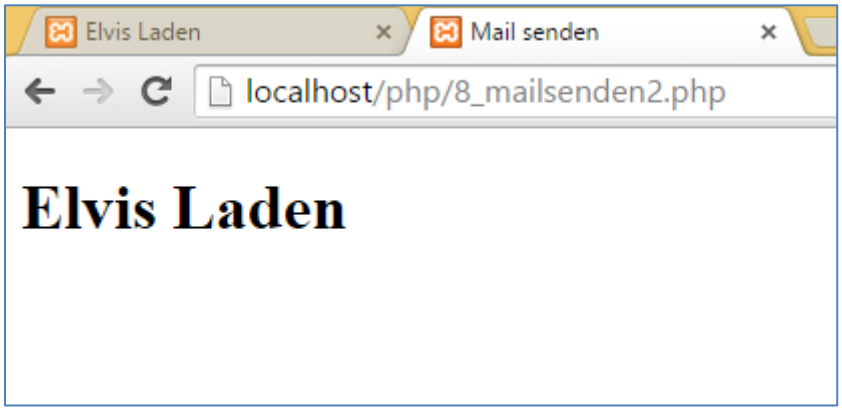
Speicher diese neue Version als „8_mailssenden2.php“.

Speicher daher auch die „8_mailsenden.html“ als „8_mailsenden2.html“ und ändere darin den Verweis auf „8_mailsenden2.php“ damit du die neue if-Funktion kontrollieren kannst.

```
<body>
<h1>NUR für den Administrator. Hier könnte der Text f
<form action = "8_mailsenden2.php" method = "post" >
  <p>Betreff: <input name = "betreff" type="text" siz
```

Testen:
gibt keinen Betreff und keinen Inhalt ein:

Ergebnis: nichts passiert, es sind keine Mails versendet worden:



Ergebnis:
Wenn die Validierung scheitert, sagt das Skript Elvis nicht, was passiert ist. Er erhält bloß eine leere Webseite.

Es fehlt eine Warnmeldung, wenn die if-Anweisung „false“ ist.

Der Code nach der if-Anweisung wird nämlich immer ausgeführt.
Eine fehlende echo-Anweisung „Sie haben keine Daten in Betreff und Inhalt eingegeben“ soll **nur** dann ausgeben werden, wenn die Testbedingung der if-Anweisung falsch ist. Wir benötigen also so etwas:

WENN Betreff nicht leer UND Inhalt nicht leer
DANN Mail senden
SONST Fehlermeldung ausgeben

Zu einem solchen Zweck bietet die if-Anweisung eine optionale **else-Klausel**, die Code ausführt, wenn die Testbedingung falsch ist. Die echo-Anweisung für unsere Fehlermeldung kann also in einer else-Klausel angegeben werden, damit sie nur ausgeführt wird, wenn beide Formularfelder nicht gefüllt wurden. **Hänge einfach nach der if-Anweisung das Schlüsselwort else** an und lasse darauf einen Codeblock in geschweiften Klammern folgen:

```
else {
  echo “Sie haben keine Daten in Betreff und Inhalt eingegeben. <br  />“;
}
```

d.h.: Dieser Code wird nur ausgeführt, wenn die if-Testbedingung mit false ausgewertet wird.

```
36  echo "Gemailt an: " . $an . "<br />";
37  }
38
39  mysqli_close($con);
40
41  }
42  else {
43      echo "Sie haben keine Daten in Betreff und Inhalt eingegeben. <br />";
44  }
45
46  ?>
47  </body>
```

Ergebnis, bei fehlenden Inhalten:

Quellen:

Lynn Beighley und Michael Morrison in: PHP & MySQL von Kopf bis Fuß;
Verlag =O´Reilly, 2009

Pröll, Zangerle, Gassler in MySQL, Das umfassende Handbuch;
Verlag Rheinwerk, 2015