Doing Web Apps

Formulare auswerten

Autor: Rüdiger Marwein
Letzte Änderung: 2012-10-18
Version: 0.9

Inhaltsverzeichnis

1.Einleitung

Mit HTML-Formularen kann man Eingabefelder, Buttons, Checkboxen und andere Eingabefelder erstellen. Ohne eine serverseitige Auswertung sind diese Elemente aber nahezu nutzlos.

Serverseitig war für solche Auswertung oft ein so genanntes CGI-Script vorhanden, dass dann beispielsweise die Daten in eine Datenbank geschrieben hat. Die gängige Sprache damals war Perl. Wir verwenden PHP.

2.Formulare erstellen

Ein einfaches HTML-Formular mit zwei Eingabefeldern und einem Button könnte mit HTML formuliert so aussehen:

```
<form name="daten" method="POST">
     <input type="text" name="email" value="" />
     <input type="text" name="username" value="" />
     <input type="submit" name="senden" value="Abschicken" />
</form>
```

Was passiert nun, wenn wir auf den Knopf drücken?

Unsere eingetragenen Daten werden hier im Beispiel mittels POST übergeben. Die mit POST übertragenen Datenmengen können unbegrenzt sein. Meist werden sie jedoch vom empfangenden Webserver limitiert auf unter 10 MB.

Empfangen wir die Daten mit einem PHP-Script, so finden wir die 3 Daten wieder in der Variable $_POST. Man kann dann mittels

$_POST['email'] bzw. $_POST['username'] bzw. $_POST['senden']

darauf zugreifen und die vom Benutzer eingetragenen Werte auslesen.

Gleiches gilt für Werte, die mittels GET übergeben werden. Hierbei ist jedoch zu beachten, dass alle Werte an die URL angehängt werden. Haben wir uns vor dem Absenden auf z.B. `http://www.webseite.de/formular.php` befunden, so sehen wir nach dem Abschicken die URL

```
http://www.webseite.de/formular.php?
email=meine.email@web.de&username=Thomas
%20Stein&senden=Abschicken
```

Die Länge so einer URL ist begrenzt. Man sollte schauen, dass sie nicht länger als 1024 Zeichen wird. Am besten bleibt man immer weit darunter. Denn: Zum einen gibt es Webserver, die nur eine gewisse Länge akzeptieren und den Rest abschneiden und zum anderen kann es passieren, dass ein Proxy auf dem Weg zum Webserver die URL beschneidet.

Des weiteren fällt das %20 auf, was in der URL stellvertretend für ein Leerzeichen steht. Mit PHP kann man für eine URL gültige Zeichen mit der Funktion urlencode() erzeugen.

Die Daten sind auf PHP-Seite dann mittels $_GET['email'] bzw.

$_GET['username'] bzw. $_GET['submit'] ansprechbar.

Bei der PHP-Installation wurden die automatischen globalen Variablen angesprochen. Ist der Schalter "register_globals" in der php.ini eingeschaltet, so kann man einfach auf $username zugreifen und hält die übergebene Variable. Warum das ziemlicher Mist ist, wurde ja bereits erklärt.

3.Elemente eines Formulars

Es gibt eine endliche Anzahl möglicher Formularfelder. Um sie auswerten zu können, muss jeweils die Option name= gesetzt sein. Sie müssen in <form>-Tags eingeschlossen sein.

Allgemeiner Hinweis: Soll in value="" ein beliebiger Text mittels einer PHP-Variable eingebaut werden, muss man darauf achten, dass keine **Gänsefüßchen** vorkommen. Was dann passiert, kann man sich leicht ausmalen. Der Text wird verstümmelt. Er ist mit **htmlspecialchars** (also " zu " oder > zu > usw.) zu bearbeiten, bevor er eingesetzt wird.

```
<?php $name = htmlspecialchars($name); ?>
<input type="text" name="name"
  size="5" maxlength="10" value="<?php echo $name; ?>" />
```

3.1.Einzeilige Eingabefelder

```
<input type="text" name="feld"
             size="5" maxlength="10" value="" />
<input type="password" name="feld"
             size="5" maxlength="10" value="" />
```

Der Unterschied besteht lediglich darin, dass bei Passwort-Feldern anstelle von Buchstaben nur Sternchen / Punkte angezeigt werden.

Auswertung: $_POST['feld']

3.2.Mehrzeilige Eingabefelder

```
<textarea name="feld" rows="2" cols="10">Langer Text
Mehrzeiliger Langer
Text</textarea>
```

Hierbei werden Zeilen (rows) für die Höhe und Spalten (cols) für die Breite angegeben. Es hängt vom verwendeten Zeichensatz ab, wie breit/hoch das im Endeffekt ist. Mit Stylesheets kann man diese Ungenauigkeit umgehen.

Damit keine führenden Leerzeichen auftauchen, muss direkt an das Tag-Ende wie im Beispiel angelegt werden. Gleiches gilt für Leerzeichen am Ende.

Auswertung: $_POST['feld']

3.3.Checkboxen und Radioboxen

```
<input type="checkbox" name="feld" value="Send" checked />
<input type="radio" name="feld" value="Send" />
```

Radioboxen sind ausschließend. Haben Sie den gleichen Namen, kann man immer nur eines aktiviert sein. Ist keine Radiobox ausgewählt, wird das Feld nicht an den Server übertragen.

Bei Checkboxen sind unterschiedliche Namen zu wählen, da man sie einzeln auswerten muss.

Ist eine Checkbox nicht ausgewählt, wird das Feld **nicht** an den Server übergeben.

Auswertung: $_POST['feld']

3.4.Dropdowns und Auswahlboxen

```
<select name="feld" size="5" multiple="multiple">
    <option value="wert1">Wert Eins</option>
    <option value="wert2">Wert Zwei</option>
</select>
```

Lässt man hier size weg oder setzt es auf 1, erhält man ein Dropdown. Ist es auf einen Wert größer 1 und multiple gesetzt ist eine Mehrfachauswahl mit gedrückter Shift-Taste möglich.

Das value der Option wird dem Select-Namen zugeordnet.

Auswertung: $_POST['feld']

3.5.Buttons

```
<input type="submit" name="feld" size="5" maxlength="10" />
<input type="image" name="feld"
       src="image.gif" size="5" maxlength="10" />
```

Der Wert eines solchen Elements ist serverseitig nur auswertbar, wenn der Knopf betätigt wurde.

Beim Submit-Image wird nicht der Name sondern eine Abwandlung mit den x-/y-Koordinaten des Bilder, auf die geklickt wurde übertragen.

Auswertung: $_POST['feld'] oder bei Image $_POST['feld_x']

4.Die Auswertung

Bei der Auswertung ist natürlich alles möglich. Die Daten könnten auf Korrektheit geprüft werden, mit einer Datenbank verglichen, per E-Mail versendet, in einer XML-Datei abgelegt werden, in eine Datenbank eingetragen werden und vieles mehr.

Wir nehmen unser erstes Beispiel und beschränken uns auf die semantische Prüfung der Korrektheit.

So könnte eine Prüfung aussehen:

```
<?php
$fehler = array();
if(isset($_POST['senden']) ||
   (isset($_POST['username']) && isset($_POST['email']))) {

  if(strpos($_POST['email'],'@')===false) {
    $fehler[] = 'E-Mail falsch.<br />';
  }
  if(strlen($_POST['username'])==0) {
    $fehler[] = 'Name nicht ausgefüllt.<br />';
  }
}
if(count($fehler)>0) {
  echo '<h1>Fehler</h1>';
  echo '<ul><li>'.join('</li><li>',$fehler).'</li></ul>';
}
?>
[... Formular ...]
```

Die E-Mail muss also ein @ enthalten um korrekt zu sein. Das ist natürlich ein blöder Test. Ein sinnvolleres Beispiel gibt es bei der Einführung zu **Regulären Ausdrücken**.

Drückt man in einem Eingabefeld die Eingabetaste, wird das Formular **automatisch abgeschickt**. Der Knopf wurde also nicht gedrückt und sein Wert ist serverseitig nicht zwingend verfügbar. Es kann aber auf die Feldinhalte geprüft werden, um festzustellen, ob das Formular abgeschickt wurde. Formulare, die keinen Senden-Button haben lassen sich in manchen Browsern nicht mit der Eingabetaste absenden.

5.Dateien hochladen (Upload)

Da unter Umständen binäre Daten übertragen werden sollen (Bilddateien oder ähnliches), brauchen wir eine andere Kodierung der Daten zur Übertragung.

```
<form method="post" enctype="multipart/form-data">
[...]
</form>
```

Passend dazu gibt es das Eingabefeld, mit dem man eine Datei auf der Festplatte selektieren kann.

```
<input type="file" name="datei" size="50">
```

Mit der Auswertung funktioniert es bei diesem Feld etwas anders. Die Details sind in der Variablen $_FILES zu finden.

```
$_FILES['datei']['name']  // Name der Datei (lokal)
$_FILES['datei']['type']  // Mime-Typ, z.B. image/gif
$_FILES['datei']['size']  // Grösse in Bytes
$_FILES['datei']['tmp_name'] // Pfad im Temp-Verzeichnis
$_FILES['datei']['error'] // Fehlercode
```

Die Fehlercodes kann man in der PHP-Hilfe nachlesen. Wenn alles geklappt hat ist **error=0**.

Mit der Funktion

```
move_uploaded_file (
     $_FILES['datei']['tmp_name'], 'c:/test/' );
```

kann die Datei dann an einen öffentlichen oder nicht-öffentlichen Ort verschoben werden.

5.1.Mehrere Dateien hochladen

Angenommen, man möchte x Bilder gleichzeitig anstatt einzeln hochladen, kann man eine Liste mit vielen solcher Eingabefeldern erstellen.

Wählt man als Feldnamen „datei[]“, kommen diese Daten als **Array** an. Allerdings haben sie die ungewöhnliche Form

```
$_FILES['datei']['name'][0]
$_FILES['datei']['size'][0]
...
$_FILES['datei']['name'][1]
$_FILES['datei']['size'][1]
...
```

Das muss also bei der Auswertung **unbedingt beachtet** werden.

6.Tipps

Will man mehrere Checkboxen, die der selben Logik folgen (z.B. zu löschende E-Mails) kann man auch hier mit [] arbeiten. Die Daten kommen ebenfalls als Array an

```
$_POST['feld'][0]
$_POST['feld'][1]
```

Es ist immer sinnvoll die Daten eines Formulars an der frühest möglichen Stelle auszuwerten, um Seiteneffekte zu verhindern und die Ergebnisse schnell zur Hand zu haben.

Werden übergebene Werte z.B. aufgrund eines Fehlers zur Korrektur automatisch in das Formular eingetragen, dann sollte man nicht vergessen, sie zuvor mit `htmlspecialchars()` behandelt zu haben.

Eingabewerten, die über die Benutzerschnittstelle ins System kommen ist

NIEMALS (!!)

zu trauen.

Das gilt für $_GET, $_POST, $_COOKIE und auch $_SERVER.

7.All-In-One-Beispiel

Anmerkung: Das Beispiel soll lediglich die HTML-Felder und die Auswertung demonstrieren und erhebt keinen Anspruch auf Vollständigkeit.

```
<?php
/**
 *  Formular auswerten
 */
 $msg = '';
 if($_POST['senden']) {
   $msg .= '<h3>Ihre Daten</h3>';
   $msg .= "<b>E-Mail</b>:";
   $msg .= htmlspecialchars($_POST['email'])."<br />\n";
   $msg .= "<b>Username</b>:";
   $msg .= htmlspecialchars($_POST['username'])."<br />";
   $msg .= '<hr />';
   if(move_uploaded_file($_FILES['datei']['tmp_name'],
        dirname(__FILE__).'/'
        .$_FILES['datei']['name'] )) {
     $msg .= '<img src="'.htmlspecialchars($_FILES['datei']['name']).'" border="0"
/>';
   }
 }
?>
<html>
<head>
<title>Formulare auswerten</title>
</head>
<body>
<?php echo $msg; ?>
<form name="daten" method="POST" enctype="multipart/form-data">
      <p><input type="text" name="email" value="<?php echo
htmlspecialchars($_POST['email']); ?>" /></p>
      <p><input type="text" name="username" value="<?php echo
htmlspecialchars($_POST['username']); ?>" /></p>
      <p><select name="feld[]" size=2 multiple>
             <option value="wert1">Wert Eins</option>
             <option value="wert2">Wert Zwei</option>
      </select></p>
      <p>Eins <input type="radio" name="fell" value="Send1"
               checked /><br />
      Zwei <input type="radio" name="fell" value="Send2" /><br />
      Drei <input type="radio" name="fell" value="Send3" /></p>
      <p><input type="file" name="datei" size="50"></p>
      <p><input type="submit" name="senden" value="Abschicken" /></p>
</form>
</body>
</html>
```