9. Auslesen von Formularinhalten

Im vorherigen Kapitel haben wir erfahren, welche Möglichkeiten uns für die Formulargestaltung zur Verfügung stehen. Nun werden wir sehen, wie eng PHP und HTML miteinander verbunden sind. Ein HTML-Formular bietet die Möglichkeit, Daten an ein PHP-Skript zu übergeben, damit es mit diesen Daten weiterarbeiten kann. In HTML werden in Verbindung mit PHP beispielsweise auch die Benutzeroberflächen, so genannte *Frontends*, für eine Datenbankanwendung erstellt. Ein gutes Beispiel hierfür ist das Administrationstool »*phpMyAdmin*«, das wir in Kapitel 14.10.1 besprechen werden.

In der Regel gibt es zwei Verfahren Formulare über PHP-Skripte auszuwerten. Die erste Möglichkeit besteht darin Formular und PHP-Skript in zwei verschiedenen Dateien unterzubringen. Die zweite Möglichkeit ist, das Skript, in dem sich das Formular befindet, wieder aufzurufen. Wir werden beide Möglichkeiten hier besprechen.

Grundsätzlich gibt es zwei Übertragungsmöglichkeiten, wie die Formularinhalte übertragen werden können, mit der »GET«- und mit der »POST«-Methode. Diese Methoden haben nichts direkt mit PHP zu tun, sie sind Elemente des HTTP-Protokolls (*Hypertext Transfer Protokolls*).

Beim Erstellen des Formulars geben wir im <form>-Tag die Übertragungsmethode mit Hilfe des Attributs »method« an, entweder »POST« oder »GET«. Des Weiteren benötigt jedes Formularobjekt einen eindeutigen Namen, damit später im PHP-Skript auf die Daten des Formularobjektes zugegriffen werden kann. Dies erfolgt mit Hilfe des »name«-Attributes. Nachdem der Submit-Knopf des Formulars gedrückt wurde, wird der Formularinhalt zum Server übertragen und kann dort ausgewertet werden. In Abhängigkeit vom gewählten Übertragungsverfahren kann man auf die übergebenen Werte zugreifen. Werden die Daten des Formulars mit der »POST«-Methode übertragen, so erfolgt der Zugriff über das »$_POST[]«-Array. In diesem Array befinden sich alle Daten des Formulars (siehe auch Kapitel 3.6). Findet die Übertragung hingegen mit der »GET«-Methode statt, so greift man darauf mit Hilfe des »$_GET[]«-Arrays zu.

- POST -> $_POST[]
- GET -> $_GET[]

Man bezeichnet die Arrays »$_GET[]« und $_POST[]« auch als superglobale Arrays. Ist die Methode im PHP-Skript nicht festgelegt, so kann die Übertragungsmethode mit Hilfe des Arrays »$_SERVER[]« ermittelt werden, in-

dem man die »REQUEST_METHOD« ausliest, dieser Wert kann dann entweder »POST« oder »GET« sein.:

listing0901.php

```
<?php
    $m = $_SERVER['REQUEST_METHOD'];
    if ($m =="GET")
        echo "Daten wurden mit der GET-Methode uebermittelt!";
    if ($m =="POST")
        echo "Daten wurden mit der POST-Methode uebermittelt!";
?>
```

In der anschaulichen Darstellung sieht die Übertragung der Daten wie folgt aus:

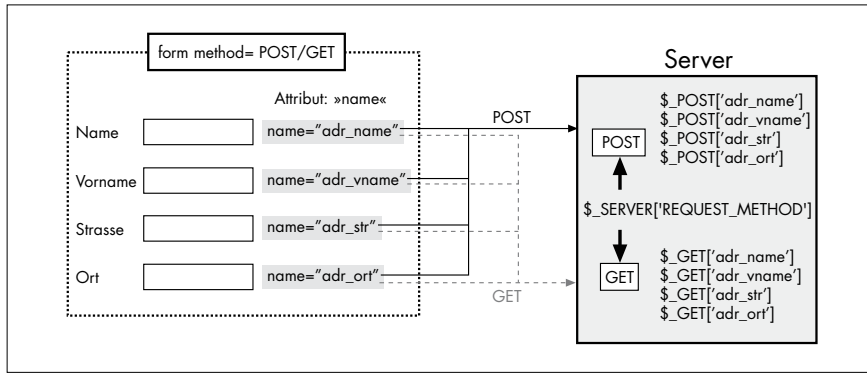

Abb. 9.1 Methoden zur Übertragung von Formularinhalten

Im Folgenden werden wir uns eingehend mit den verschiedenen Verfahren zur Formularauswertung beschäftigen.

9.1 POST-/GET-Methode

GET-Methode
Bei der GET-Methode wird ein URL-String erzeugt (*uniform ressource locator*). In diesem URL-String steht der Name des Skriptes, welches beim Absenden des Formulars aufgerufen wird, mit einem abschließenden Fragezeichen, gefolgt von dem zu übertragenden Formularinhalt, z.B.:

```
auslesen.php?f_name=Meyer&f_vname=Nicole
```

In diesem Beispiel heißt das aufzurufende Skript »auslesen.php« und es werden die Variablen »f_name« und »f_vname«, getrennt durch ein »&«, übergeben. Diese Variablen haben die Werte »Meyer« und »Nicole«. Das aufzurufende Skript wird über das »action«-Attribut des <form>-Tags definiert. Die Erzeugung des URL-Strings folgt definierten Regeln:

- Am Anfang steht das aufzurufende Skript, gefolgt von einem »?«.
- Name des Formularobjekts und Wert werden durch ein »=« getrennt.
- Einzelne Namen-/Daten-Paare werden durch ein »&« getrennt.
- Alle Zeichen aus dem erweiterten ASCII-Satz (128-255) werden durch einen hexadezimalen Wert ersetzt, eingeleitet durch ein »%«.
- Leerzeichen werden durch ein »+« ersetzt.
- Maximale Datenlänge beträgt üblicherweise ca. 2kByte (serverabhängig).

POST-Methode
Soll die POST-Methode zur Übertragung von Daten verwendet werden, so ist dieses im »method«-Attribut des form-Tags anzugeben. Im Allgemeinen wird diese Methode zur Übertragung von großen Datenmengen verwendet. Ein Vorteil bei der Verwendung der POST-Methode kann auch die unsichtbare Übertragung der Daten sein. Die POST-Methode sendet den Formularinhalt direkt in einem Block an die superglobale Serverumgebungsvariable »$_POST[]«, die dann ausgelesen werden kann.

Wann sollte man welche Methode verwenden?
Ungünstig auf die Verwendung der GET-Methode kann sich die serverseitige Längenbeschränkung des URL-Strings auswirken. Bei großen Formularen mit großen Datenmengen sollten Sie daher die POST-Methode verwenden. Dadurch dass bei der POST-Methode Variablen nicht wie bei der GET-Methode im URL-String übergeben werden, ist eine direkte manuelle Änderung in der Adresszeile des Browsers mit der POST-Methode nicht möglich. Nachdem Sie nun wissen, welche Vor- und Nachteile die Methoden besitzten, werden wir uns anschauen, wie eine Werteübertragung funktioniert.

9.2 Auslesen von Texteingabefeldern

Die prinzipielle Darstellung einer Formularübertragung sieht wie folgt aus: Zuerst einmal wollen wir mit zwei Dateien arbeiten (natürlich kann man auch beide Skripte zusammenfassen, aber dazu kommen wir später). Die erste Datei »formular .html« enthält das Formular mit den Eingabefeldern »adr_name«

und »adr_ vname«, deren Inhalt zum Server übertragen werden soll. Als Übertragungsmethode wurde »POST« gewählt, d.h. wir müssen später die Daten über das »$_POST[]«-Array auslesen. Außerdem enthält das Formular einen Submit-Knopf »ABSCHICKEN«, damit die Übertragung der eingegebenen Werte gestartet werden kann.

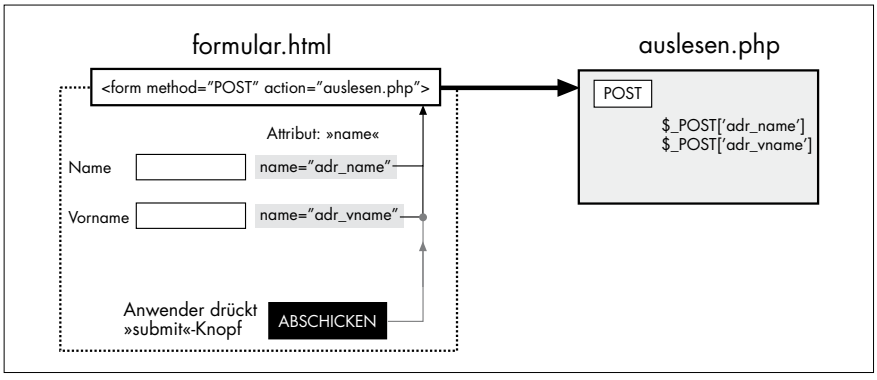

Abb. 9.2 Prinzipielles Schema der Übertragung von Formulardaten zum Auswertungsskript

Jedes Formular muss mindestens einen Submit-Knopf besitzen. Das im action-Attribut definierte Skript wird nach dem Drücken des Submit-Knopfes aufgerufen. In unserem Fall ist es das Skript »auslesen.php«. Die übertragenen Werte können dann mit »$_POST['adr_ name']« bzw. »$_POST['adr_ vname']« ausgelesen werden. Das Listing zu »formular. html« sieht folgendermaßen aus:

listing0902a.php

```
<html>
<body>
<form name ="mf" action ="auslesen.php" method ="POST">
    Name<input type ="text" name ="adr_name" value=""><br />
    Vorname<input type="text" name="adr_vname" value=""><br />
    <input type="submit" name="sb" value="ABSCHICKEN">
</form>
</body>
</html>
```

Das Skript »auslesen.php« sieht wie folgt aus:

listing0902b.php

```
<?php
    $f_name   = $_POST['adr_name'];
    $f_vname  = $_POST['adr_vname'];
    echo "Sie haben als Namen: $f_name eingegeben<br>";
    echo "Im Feld Vorname haben Sie: $f_vname eingegeben";
?>
```

Beide Skripte können auch zu einem Skript zusammengefasst werden. Dazu ruft sich das Skript selbst auf, sobald der Anwender den submit-Knopf drückt. In diesem Zusammenhang wollen wir auch die Bedeutung der Variablen »$_SERVER['PHP_SELF']« klären. Schauen wir uns folgendes Skript an:

listing0903.php

```
<html>
<head></head>
<body>
<?php
    if (isset($_POST['sb'])){
       $f_name   = $_POST['adr_name'];
       $f_vname  = $_POST['adr_vname'];
       echo "Sie haben als Namen: $f_name eingegeben.<br />";
       echo "Im Feld Vorname haben Sie: $f_vname eingegeben.<br />";
       echo "<a href='{$_SERVER['PHP_SELF']}'>zur Eingabe..</a>";
    }
    else{
       echo "<form name ='mf' action ='{$_SERVER['PHP_SELF']}'
       method ='POST'>
       <table  border='0' cellpadding='1' cellspacing='1'
                                    bgcolor='#000000' align='center'>
          <tr bgcolor='#FFFFFF'>
             <td>Name</td>
             <td><input type ='text' name ='adr_name' value=''></td>
          </tr>
          <tr bgcolor='#FFFFFF'>
             <td>Vorname</td>
             <td><input type='text' name='adr_vname' value=''></td>
          </tr>
          <tr bgcolor='#999999' height='45'>
             <td colspan='2'align='center'>
             <input type='submit' name='sb' value='ABSCHICKEN'></td>
          </tr>
       </table>
       </form>";
    }
?>
</body></html>
```

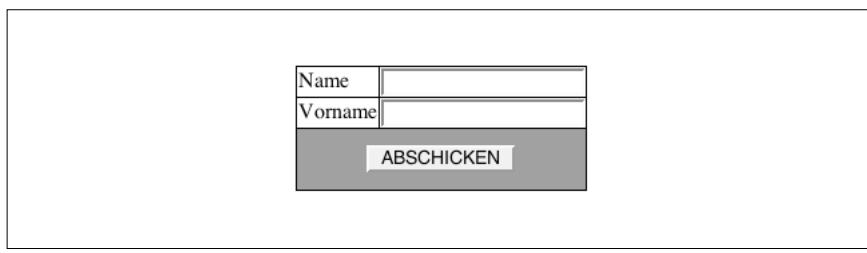

Abb. 9.3 Formular zur Texteingabe

Das Skript teilt sich in zwei Teile auf. Der »if«-Teil und der »else«-Teil. Beim ersten Aufruf wird der »else«-Teil abgearbeitet. Die »if«-Bedingung prüft mit Hilfe der »isset()«-Funktion, ob die Variable »sb« existiert. Die Bezeichnung »sb« haben wir dem Submit-Knopf gegeben, der den Wert »ABSCHICKEN« hat. Beim ersten Aufruf existiert »sb« nicht, da das Skript ja nicht durch das Drücken des Submit-Knopfes aufgerufen wurde, also wird der »else«-Teil des Skriptes abgearbeitet, der das Formular darstellt.

Hat der Benutzer das Formular ausgefüllt und den Submit-Knopf gedrückt, wird dasselbe Skript aufgerufen, da im »action«-Attribut des form-Tags »$_SERVER['PHP_SELF']« steht. Hier ist der Name des aktuellen Skriptes abgespeichert. Nach dem wiederholten Aufruf ist nun der Wert für »sb« gleich »ABSCHICKEN«, jetzt wird der »if«-Teil des Skriptes abgearbeitet, was beim ersten Durchlauf nicht der Fall war, da der Submit-Knopf nicht gedrückt wurde. Dieses Mal wird der HTML-Code für das Formular mit einer »echo«-Anweisung ausgegeben. Zusätzlich wird ein Hyperlink erzeugt, der es uns ermöglicht, das Formular wieder aufzurufen, um eine neue Eingabe zu machen.

Die feinen schwarzen Linien um die Eingabefelder erzeugt man dadurch, dass man den Hintergrund der Tabelle mit Schwarz definiert, den Zellenabstand und den Zellinhaltsabstand auf »1« setzt und den Zellen eine Farbe gibt:

```
<table border='0' cellpadding='1' cellspacing='1' bgcolor='#000000'>
```

Achten Sie bei den HTML-Tags auf die einfachen Hochkommata, ein doppeltes Hochkomma würde das Ende der auszugebenen Zeichenkette markieren! Natürlich können Sie auch das doppelte Hochkomma entwerten, indem Sie einen Backslash »\"« voranstellen.

Eine weitere Besonderheit ist die Variable »$_SERVER ['PHP_SELF']«. In ihr ist der Name des aktuellen Skriptes abgespeichert. Beachten Sie, hierbei handelt es sich um ein Array und da dürfen die »{ }« geschweiften Klammern bei der Ausgabe in der »echo«-Anweisung nicht vergessen werden.

9.3 Auslesen von Radio-Buttons und Checkboxen

Die Funktion und der Zweck von Radio-Buttons wurde in Kapitel 8 besprochen. Wie heißt es doch so schön, jeder Knopf hat seinen eigenen Wert, jede Knopf-Gruppe hat einen gemeinsamen Namen. So lesen wir im folgenden Skript den Wert einer Radio-Button-Gruppe aus:

listing0904.php

```
<html><head></head><body>
<?php
    if (isset($_POST['sb'])){
       $hgf = $_POST['hg_col'];
       echo "<body bgcolor=\"#$hgf\">";
    }
    else{
       echo "<body bgcolor=\"#FFFFFF\">";
    }
?>
<form name="mf" action="<?php $_SERVER['PHP_SELF'] ?>" method="POST">
    <table width="350" border="0" cellpadding="1" cellspacing="1"
                    bgcolor="#000000" align="center">
    <tr valign="middle" align="center">
       <td width="35" bgcolor="#93C0C5" align="left">
          <font color="#333333">Farbauswahl: </font>
       </td>
       <td width="36" bgcolor="#006666" align="center">
          <input type="radio" name="hg_col" value="#006666"
          onfocus="if(this.blur)this.blur()">
       </td>
       <td width="36" bgcolor="#990000" align="center">
          <input type="radio" name="hg_col" value="#990000"
          onfocus="if(this.blur)this.blur()">
       </td>
       <td width="36" bgcolor="#3333CC">
          <input type="radio" name="hg_col" value="#3333CC"
          onfocus="if(this.blur)this.blur()">
       </td>
       <td width="36" bgcolor="#000000">
          <input name="hg_col" type="radio"
          onfocus="if(this.blur)this.blur()" value="#000000" checked>
       </td>
    </tr>
    <tr>
       <td colspan="5" align="center" bgcolor="#999999">
       <input type="submit" name="sb" value="Farbe Ausw&auml;hlen">
       </td>
    </tr>
</table>
</form>
</body></html>
```

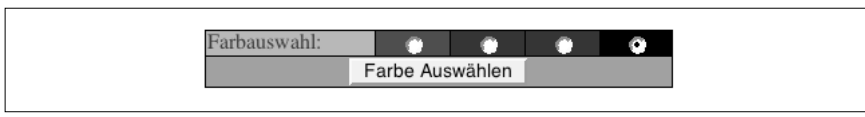

Abb. 9.4 Tabelle mit einer Radio-Button-Gruppe zum Wählen einer Hintergrundfarbe

Schauen wir uns das Skript noch einmal im Detail an. Das Formular wird in einer Tabelle angeordnet, damit eine präzise Ausrichtung möglich ist. Das Ausse-

hen dieser Tabelle mit den Formularobjekten wird in Abbildung 9.4 gezeigt. Im <form>-Tag verwenden wir wieder die PHP-Variable »$_SERVER['PHP _SELF']«

```
<form name="mf" action="<?php $_SERVER['$PHP_SELF'] ?>" method="POST">
```

Dann erfolgt die Definition der Radio-Button-Gruppe »hg_col«. Jeder Knopf dieser Gruppe hat einen anderen Wert und zwar den entsprechenden Farbwert, z.B. »#3333CC«. Wenn also der Knopf gedrückt wird, der den Wert »#3333CC« hat, bekommt die Radio-Button-Gruppe »hg_col« den entsprechenden Wert. Drückt der Anwender jetzt den Submit-Knopf, ruft sich das Formular selbst auf und das PHP-Skript wird abgearbeitet:

listing0905.php

```
<?php
    if (isset($_POST['sb']) ){
        $hgf = $_POST['hg_col'];
        echo "<body bgcolor=\"#$hgf\">";
    }
    else{
        echo "<body bgcolor=\"#FFFFFF\">";
    }
?>
```

Der einzige Wert, der variabel ist, ist die Hintergrundfarbe, und die wird im <body>-Tag definiert, also muss diese Definition über das PHP-Skript gesteuert werden. Es wird geprüft, ob der Submit-Knopf gedrückt wurde; wenn ja, dann wird der Wert der Radio-Button-Gruppe »hg_col« ausgelesen und in der Variablen »$hgf« abgespeichert. Mit Hilfe einer echo-Anweisung wird dann der <body>-Tag ausgegeben, in dem die Farbe über die Variable »$hgf« definiert wird. Es kann sein, dass Sie das Geheimnis der Anweisung

```
onfocus="if(this.blur)this.blur()"
```

nicht kennen. Sie ist dafür da, dass bei Browsern auf Macintosh-Rechnern kein blauer Rand um die Formularobjekte erscheint. Also eine rein optische Perfektionierung der Darstellung!

In diesem Beispiel haben wir einen klassischen Fall einer PHP-gesteuerten Ausgabe kennengelernt. Die Abschnitte des HTML-Dokuments, die variabel sind, werden durch das entsprechende PHP-Skript gesetzt bzw. generiert.

Checkbox

Das Auslesen der Checkboxen verhält sich ähnlich wie bei den Radio-Buttons. Wie wir wissen, hat jede Checkbox im HTML-Code einen eigenen Namen, auf den man sich im PHP-Skript beziehen kann.

listing0906.php

```
<html>
<body>
<?php
    if (isset($_POST['sb'])){
        $f_land1 = $_POST['land1'];
        $f_land2 = $_POST['land2'];
        $f_land3 = $_POST['land3'];
        echo "Im Feld Land haben Sie: <br /><br />";
        if (isset($f_land1))
            echo "$f_land1<br>";
        if (isset($f_land2))
            echo "$f_land2<br>";
        if (isset($f_land3))
            echo "$f_land3<br>";
        echo "<br /> ausgegeben.<p>";
        echo "<a href=\"{$_SERVER['PHP_SELF']}\">zur Eingabe</a>";
    }
    else{
        echo "<form name =\"mf\" action =\"{$_SERVER['PHP_SELF']}\"
            method =\"POST\">
        <table border=\"0\" cellpadding=\"1\" cellspacing=\"1\"
            bgcolor=\"#000000\" align=\"center\">
            <tr bgcolor=\"#999999\" height=\"45\">
                <td colspan=\"2\" align=\"center\">
                &Ouml;sterreich
                <input type=\"checkbox\" name=\"land1\"
                       value=\"&Ouml;sterreich\"><br />
                   Schweiz <input type=\"checkbox\" name=\"land2\"
                       value=\"Schweiz\"><br />
                   Deutschland <input type=\"checkbox\" name=\"land3\"
                       value=\"Deutschland\"><br />
                <input type=\"submit\" name=\"sb\" value=\"Lesen >\">
                </td>
            </tr>
        </table>
        </form>";
    }
?>
</body>
</html>
```

In diesem Skript haben wir drei Checkboxen definiert: »land1«, »land2«, »land3«. Über das »$_POST[]«-Array lesen wir den der Checkbox zugewiese-

nen Wert »value« aus und weisen diesen jeweils den Variablen »$f_land1«, »$f_land2« und »$f_land3« zu.

```
$f_land1 = $_POST['land1'];
$f_land2 = $_POST['land2'];
$f_land3 = $_POST['land3'];
```

Der Wert dieser Variablen ist bei ausgewählter Box der Wert, der im Formular definiert wurde. Ist die Checkbox nicht angewählt worden, so existiert die Variable nicht. Bevor wir den Variablenwert »$f_land1«, $f_land2« oder »$f_land3« ausgeben, überprüfen wir deren Existenz mit der Funktion »isset()«, um keine sinnlose Ausgabe zu erzeugen. Die Funktion »isset()« gibt entweder ein »false« oder »true« zurück, um die »if«-Verzweigung zu steuern.

9.4 Auslesen von Listen mit Einfach- und Mehrfachauswahl

Listen werden mit dem <select>-Tag erzeugt. Man unterscheidet zwischen Listen, in denen man *einen* Listenpunkt auswählen kann und Listen mit Mehrfachauswahl. Hierbei gibt es einige Hürden zu nehmen, um in einem PHP-Skript auf die Elemente zugreifen zu können. Um auf den ausgewählten Listeneintrag zugreifen zu können, verwenden wir wieder die »$_POST[]«-Variable. Das Prinzip gleicht dem des Auslesens eines Texteingabefeldes. Schauen wir uns einmal ein einfaches Listenfeld an:

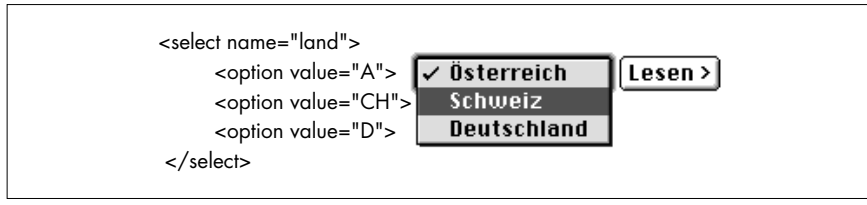

Abb. 9.5 Aufbau einer Select-Box

listing0907.php

```
<html><body>
<?php
    if (isset( $_POST['sb']) ){
        $f_land = $_POST['land'];
        echo "Im Feld Land haben Sie: $f_land angegeben.<p>";
        echo "<a href=\"{$_SERVER['PHP_SELF']}\">zur Eingabe..</a>";
    }
    else{
        echo "<form name =\"mf\" action =\"{$_SERVER['PHP_SELF']}\"
        method =\"POST\">
```

```
        <table  border=\"0\" cellpadding=\"1\" cellspacing=\"1\"
        bgcolor=\"#000000\" align=\"center\">
        <tr bgcolor=\"#999999\" height=\"45\"><td colspan=\"2\"
        align=\"center\">
        <select name=\"land\">
           <option value=\"A\">&Ouml;sterreich</option>
           <option value=\"CH\">Schweiz</option>
           <option value=\"D\">Deutschland</option>
        </select>
        <input type=\"submit\" name=\"sb\" value=\"Lesen >\">
        </td> </tr>
        </table>
        </form>";
    }
?>
</body>
</html>
```

Bitte achten Sie hier ebenfalls auf die geschweiften Klammern »{ }« bei der Ausgabe eines Arrays in einer echo-Anweisung. Kommen wir nun zum Auslesen einer Liste, die dem Anwender eine Mehrfachauswahl ermöglicht. Diese Art von Listen hat meistens folgendes Aussehen:

Abb. 9.6 Aussehen einer Mehrfachauswahl-Liste

Während Sie die einzelnen Listenpunkte mit dem Mauscursor anwählen, halten Sie die »ctrl«-Taste auf einem Windows-Rechner bzw. »«-Taste auf einem Apple-Macintosh-Rechner gedrückt. So können Sie mehrere Punkte anwählen. Doch nun zum Listing. Wir konstruieren wieder ein Skript, welches sich selbst aufruft. Der Selbstaufruf erfolgt durch »$_SERVER['PHP_ SELF']«. Das Geheimnis liegt allerdings in dem Namen für die Select-Box:

```
<select name=\"land[]\" multiple size=\"3\">
```

Wir haben hier »land[]« gewählt. Dieser Name zeigt an, dass es sich hierbei um ein Array handelt. Ebenso muss man das Attribut »multiple« setzen, um eine Mehrfachauswahl zu ermöglichen, »size« gibt die Anzahl der sichtbaren Listenpunkte an, hier »3«. Wenn das Formular ausgelesen werden soll, muss jetzt

natürlich auf ein Array zugegriffen werden, und das erfolgt am einfachsten mit Hilfe der foreach-Schleife:

```
foreach( $_POST['land'] as $y){
        echo "$y<br>";
}
```

Abschließend noch einmal das komplette Listing:

listing0908.php

```
<html><body>
<?php
    if (isset($_POST['sb'])){
        foreach( $_POST['land'] as $y){
            echo "$y<br>";
        }
        echo "ausgewaehlt!";
        echo "<a href=\"{$_SERVER['PHP_SELF']}\">zur Eingabe</a>";
    }
    else{
        echo "<form name =\"mf\" action =\"{$_SERVER['PHP_SELF']}\"
        method =\"POST\">
        <table  border=\"0\" cellpadding=\"1\" cellspacing=\"1\"
            bgcolor=\"#000000\" align=\"center\">
            <tr bgcolor=\"#999999\" height=\"45\">
                <td colspan=\"2\" align=\"center\">
                <select name=\"land[]\" multiple size=\"3\">
                    <option value=\"A\">&Ouml;sterreich</option>
                    <option value=\"CH\">Schweiz</option>
                    <option value=\"D\">Deutschland</option>
                </select>
                <input type=\"submit\" name=\"sb\" value=\"Lesen >\">
                </td>
            </tr>
        </table>
        </form>";
    }
?>
</body>
</html>
```

Die zweite Möglichkeit für das Auslesen einer Liste mit Mehrfachauswahl ist die Auswertung der Variablen »QUERY_STRING«. Das Aussehen der Select-Box ist identisch mit der im vorherigen Skript. Als Übertragungsmethode haben wir hier »GET« gewählt.

Nachdem das Formular abgeschickt wurde, wird mit Hilfe der Funktion »getenv()« auf den »QUERY_STRING« zugriffen; er wird der Variablen »$f_

land« zugewiesen. Diese Funktion zeigt den Wert einer Umgebungsvariable an, in unserem Fall »QUERY_STRING« (siehe auch DIN-A3-Einleger am Ende des Buches).

```
$f_land = getenv("QUERY_STRING");
```

Nachdem wir den »QUERY_STRING« ausgelesen haben, müssen wir die Werte aufspalten, denn wie wir in Kapitel 9.1 besprochen haben, liegen die Werte in einer Zeichenkette vor. Diese Werte sind voneinander durch ein »&« getrennt. Wenn z.B. »Schweiz« und »Deutschland« ausgewählt wurden, sieht der String wie folgt aus:

```
land=CH&land=D&sb=lesen+>
```

Mit Hilfe der »explode()«-Funktion spalten wir diese Zeichenkette auf:

```
$f_auswahl = explode("&",$f_land);
```

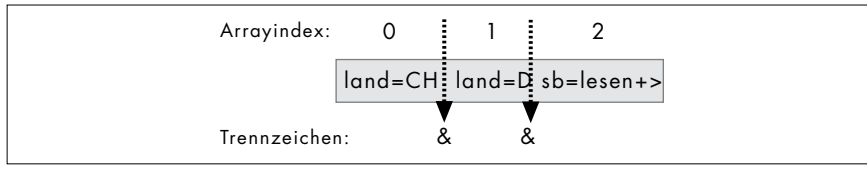

Abb. 9.7 Aufspaltung eines »QUERY_STRING«

Nachdem wir den »QUERY_STRING« aufgespalten haben und die einzelnen Elemente im Array »$f_auswahl« abgespeichert haben, können wir das Array mit Hilfe einer foreach-Schleife auslesen. Hier folgt das komplette Skript:

listing0909.php

```
<html><body>
<?php
    if ( isset($_GET['sb'] )){
        $f_land = getenv("QUERY_STRING");
        $f_auswahl = explode("&",$f_land);
        echo "Im Feld Land haben Sie: <br />";
        foreach ($f_auswahl as $w){
            echo "$w<br>";
        }
        echo "ausgewaehlt!";
        echo "<a href=\"{$_SERVER['PHP_SELF']}\">zur Eingabe</a>";
    }
```

```
    else{
        echo "<form name =\"mf\" action =\"{$_SERVER['PHP_SELF']}\"
            method =\"GET\">
        <table  border=\"0\" cellpadding=\"1\" cellspacing=\"1\"
            bgcolor=\"#000000\" align=\"center\">
            <tr bgcolor=\"#999999\" height=\"45\">
                <td colspan=\"2\" align=\"center\">
                <select name=\"land\" multiple size=\"3\">
                    <option value=\"A\">&Ouml;sterreich</option>
                    <option value=\"CH\">Schweiz</option>
                    <option value=\"D\">Deutschland</option>
                </select>
                <input type=\"submit\" name=\"sb\" value=\"Lesen >\">
                </td>
            </tr>
        </table>
        </form>";
    }
?>
</body>
</html>
```

9.5 Dynamische Erzeugung von Formularelementen

Wenn sich Formularelemente im Laufe der Zeit ändern, ist es von Vorteil, wenn sie je nach aktuellem Datenbestand neu generiert werden können. Man kann sich z.B. Select-Boxen mit einer Liste von Werten vorstellen, die variiren.

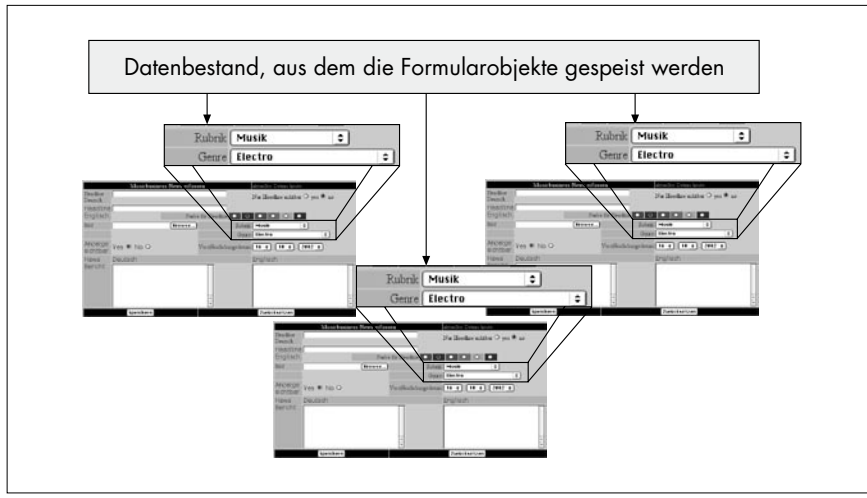

Abb. 9.8 Dynamische Formularobjekte