

Programmieren der Untersuchung

Während in HTML viele Tags zur Formatierung und Beschreibung eines Textes zur Verfügung stehen, bieten Programmiersprachen wie z.B. PHP oder Java Script Strukturen die einen komplexen Ablauf der Seite ermöglichen. Die wichtigsten Elemente einer solchen Sprache sind Variable, Ausdrücke, Abfragen, Schleifen und eine Reihe an mitgelieferten Funktionen. Die Codierung bzw. Syntax dieser Elemente ist zwischen den (3 GL) Sprachen sehr unterschiedlich, das Prinzip aber immer gleich. Im Folgenden wird auf die eben genannten Elemente näher eingegangen und die Unterschiede hinsichtlich der Verwendung von PHP und Java Script erläutert. Daran anschließend werden Programmbeispiele in PHP und Java Script besprochen.

Variable und Ausdrücke

Variable sind kleine Speicherplätze denen ein Namen zugewiesen wird. Der Inhalt der Speicherplätze kann innerhalb des Programmcodes manipuliert werden. So kann z.B. der Variable \$var der Wert 5 zugewiesen werden:

```
$var=5;
```

Dies nennt man dann einen Ausdruck. Im Laufe des Programms kann es sinnvoll sein, die Variable um den Wert 1 zu erhöhen. Dies geschieht mit Ausdruck wie z.B.

```
$var = $var + 1;
```

Manche Variablen erhalten schon vom System Inhalte wie z.B. die IP-Adresse:

```
$HTTP_SERVER_VARS['REMOTE_ADDR']
```

Abfragen

Abfragen oder Verzweigungen werden benötigt, um auf unterschiedliche Variableninhalte zu reagieren. Angenommen man speichert in der Variable \$alter das Alter der Versuchsperson und man möchte nur jene Personen den Fragebogen vorgeben, die älter sind als 15 Jahre. So kann man dies mit einer einfachen Abfrage programmieren, die so ähnlich aussieht:

```
if ($alter <16)
{
    ... Fehlermeldung ...
}
else
{
    ... Vorgabe des Fragebogens ...
}
```

Eine Abfrage besteht meist aus dem Schlüsselwort „if“ gefolgt von einem Bedingungsteil (in diesem Beispiel: (\$alter<16)). Dann erfolgt die Kennzeichnung des „then – Teils“ (in diesem Beispiel: { ... Fehlermeldung }) und des „else – Teils“ (in diesem Beispiel: else { Vorgabe des Fragebogens ... })

Schleifen

Mit Hilfe von Schleifen können Programmteile beliebig oft wiederholt werden. Hat man z.B. eine 10-stufige Antwortskala, die mit Radios abgebildet werden, so kann man diese entweder 10 mal kopieren oder eine Schleife mit dem entsprechenden Code 10 mal durchlaufen.

```
$i=0;
while ($i<10)
{
    ... Ausgabe des Radio - Eingabefeldes ...
    $i=$i+1;
}
```

In diesem Beispiel wird der Variable `$i` zu Beginn der Wert 0 zugewiesen. Die Schleife selbst beginnt mit dem „while“ gefolgt von einer Bedingung (im Beispiel: (`$i <10`)) und einem Durchführungsteil (`{ Ausgabe Radios ... $i=$i+1; }`). Im Klartext bedeutet dieser Befehl. Solange (while) in der Variable `$i` der Wert kleiner als 10 ist notiere ein Radio Eingabefeld und erhöhe die Variable `$i` um 1. Dadurch wird die Schleife genau 10 mal durchgeführt.

Funktionen

In den meisten Programmiersprachen werden verschiedene vordefinierte Funktionen mitgeliefert wie z.B. eine Funktion für das Auslesen des aktuellen Datums. Eine andere Funktion um Teile einer Zeichenkette zu extrahieren, ein File zu öffnen, zu schreiben und zu lesen, etc.

PHP vs. Java Script

Wie bereits erwähnt wird PHP serverseitig durchgeführt und Java Script clientseitig (am Rechner der Teilnehmer). Für die Programmierung ergeben sich daraus einige Konsequenzen:

1) Eingabefelder → Variable:

Möchte man auf den Inhalt der Eingabefelder zugreifen (also auf die Eingabe des Benutzers reagieren) so kann das mit Hilfe von Java Script direkt erfolgen. Verwendet man PHP, müssen die Inhalte zuerst auf den Server übertragen werden. Dies geschieht wenn z.B. ein „Submit“ durchgeführt wird beim Aufruf der nächsten Seite. Der Zugriff auf die Inhalte ist somit erst auf der nächsten Seite möglich!! Da Java Script clientseitig läuft ist der Zugriff während der Eingabe schon möglich.

2) Filetyp:

Während PHP – Code nur dann durchgeführt wird, wenn das File die Endung `.php` aufweist, verfolgt Java Script die gleichen Regeln wie HTML.

3) Sichtbarkeit des Codes / Blockierung

Während Java Script im HTML – Code auch für den Client sichtbar ist (z.B. mit Quelltext anzeigen), sieht der Teilnehmer (natürlich nur wenn das File auf einem Server mit PHP Installation mit der Endung `.php` liegt) den PHP – Code nicht, da dieser am Server durchgeführt wird und nur mehr der HTML – Code übertragen wird. Daher ist es auch für den Browser des Teilnehmers nicht möglich, die Ausführung von PHP – Code zu verhindern während ihm das bei Java Script möglich ist.

Programmbeispiele mit PHP

Um den PHP – Code durchzuführen, muss das File die Endung .php aufweisen und auf einem entsprechenden Server liegen. Der PHP – Code selbst wird mit den Tags <?php eingeleitet und mit ?> wieder geschlossen. Innerhalb dieser Tags gilt die PHP – Syntax.

Ausgabe von Variableninhalten in HTML - Text

Möchten wir z.B. den Inhalt einer Variable in einem HTML – Code ausgeben, so verwenden wir dazu den Befehl „echo“. An der gewünschten Stelle im HTML Code wird z.B. folgender Code notiert:

```
<?php
    echo "Ihre IP - Adresse ist:";
    echo $HTTP_SERVER_VARS['REMOTE_ADDR'];
?>
```

Zuerst wird der statistische Text „Ihre IP-Adresse ist:“ ausgegeben. Da es sich hierbei um eine Folge von Zeichen handelt (auch Zeichenfolge oder String genannt) muss dieser Text lt. PHP – Syntax unter Hochkomma gestellt werden. Jeder PHP – Befehl muss außerdem mit einem Semikolon (;) beendet werden.

Variablen beginnen in PHP mit einem \$ gefolgt vom Variablennamen. Einige Variable werden vom Betriebssystem bzw. PHP selbst befüllt. Eine solche Variable ist \$HTTP_SERVER_VARS['REMOTE_ADDR']. Sie enthält die IP – Adresse. Variable müssen nicht unter Hochkomma gesetzt werden. Der Strichpunkt am Ende ist wie bei jeder Anweisung in PHP auch hier nötig.

Ausgabe von Variableninhalten in ein Eingabefeld

Aus dem Kapitel „HTML“ kennen wir die Eingabebefehle und die Möglichkeit Initialwerte anzugeben. Das Eingabefeld

```
<input type="text" name="Seitennummer" value="1">
```

ist ein Texteingabefeld mit dem Namen Seitennummer, das zu Beginn gleich den Wert 1 enthält. Mit Hilfe von PHP ist es möglich hier statt dem Wert 1 eine PHP – Variable anzugeben:

```
<HTML>
...
<BODY>
...
    <?php
        $Startseite=1;
    ?>

    <input type="text" name="Seitennummer" value="<?php echo
    $Startseite;?>">
...
</BODY>
...
```

</HTML>

In diesem Beispiel wird zwar auch eine 1 in dem Eingabefeld erscheinen, es handelt sich aber jetzt um den Inhalt der Variable Startseite.

Zugreifen auf Eingaben der Teilnehmer in die Eingabefelder

Wie bereits mehrfach erwähnt kann der Zugriff auf Teilnehmereingaben erst erfolgen, wenn die Daten an den Server übermittelt wurden. Die Übermittlung der Daten kann als POST oder als GET erfolgen. GET bedeutet, dass die Daten in der URL als Parameter (als für den Teilnehmer sichtbar) übergeben werden. Mit POST werden die Daten im Standardeingabekanal des nachfolgenden Skripts zur Verfügung gestellt.

Diese Angabe mit der Information, welche Seite als nächstes aufgerufen werden soll, wird im <form> - Tag der HTML Syntax bekannt gegeben. Zwischen öffnendem und schließendem Form - Tag müssen alle Eingabefelder angeführt werden auf deren Inhalt auf der nächsten Seite zugegriffen werden soll. Außerdem wird in diesem Tag mit dem Attribut ACTION angegeben, was mit den Daten gemacht werden soll, wenn ein Submit (= wenn auf einen „submit - Button geklickt wird oder mit Hilfe von Java Script die Aktion „Submit“ initiiert wird) durchgeführt wird. In unserem Fall wollen wir die nächste Seite aufrufen und Ihr die Daten zur Verfügung stellen. Es wäre aber auch denkbar, die Daten via E-Mail zu versenden. In diesem Fall würde man bei dem ACTION Attribut den Wert „<mailto:email.adresse@univie.ac.at>“ angeben. Konkret sieht ein typischer Aufruf folgendermaßen aus:

Seite1.php:

```
<HTML>
...
<BODY>
<FORM METHOD="POST" ACTION="Seite2.php">
    ...
    <input type="text" name="Seitennummer" value="1">
    .....
    <input type=submit name="naechsteSeite" value="weiter">
</FORM>
</BODY>
</HTML>
```

Auf der zweiten Seite stehen sodann die Eingaben des Benutzers als Variable zur Verfügung. Der Variablenname richtet sich nach dem Namen des Eingabefeldes. Im obigen Beispiel würde auf der Seite "Seite2.php" in der Variable \$Seitennummer der Wert 1 stehen bzw. jener Wert den der Benutzer eingegeben hat. Z.B.

Seite2.php

```
<?php
    echo „In der Variable Seitennummer steht:“;
    echo $Seitennummer;
?>
```

dynamische Vergabe von Seitennummern

Die Seitennummer ist ein wichtiger Hinweis für die Teilnehmer. Oft wird diese einfach statisch auf den jeweiligen Seiten notiert. Kommt dann eine Seite hinzu, müssen alle

nachfolgenden Seiten verändert werden. Einfacher wäre es, wenn man mit Hilfe eines kleinen Programmteils die Seitennummer dynamisch ermitteln würde.

Das funktioniert z.B. folgendermaßen: wir wissen bereits, wie man eine Variable um einen Wert erhöht (`$var = $var + 1 ;`), wie man den Wert eines Eingabefeldes auf die nächste Seite überträgt, wie man den Inhalt einer Variable in einem Eingabefeld anzeigt und wie man eine Variable als Text ausgibt. Damit haben wir auch schon fast alle Zutaten die wir für dieses Beispiel brauchen. Es fehlt uns lediglich die Information, dass Eingabefelder auch unsichtbar sein können. Diese Angabe erfolgt im type Attribut des input tags mit dem Wert "hidden":

Seite1.php

```
<form method=post action="Seite2.php" name="f1">
...
    <input type="hidden" name="Seitennummer" value="1">
...
</form>
```

Diese Angabe wird auf der ersten Seite zwischen die öffnenden und schließenden Form-Tags kopiert. Klickt der Teilnehmer auf den Submit-Button, so wird die nächste Seite aufgerufen und die Werte des Eingabefeldes (in dem Beispiel 1) stehen als Variable zur Verfügung. Auf der zweiten Seite steht nun in der Variable `$Seitennummer` der Wert 1. Da wir uns auf der nächsten Seite befinden, erhöhen wir diesen Wert um 1 und geben ihn bei der Seitennummernanzeige aus:

Seite2.php

```
...
<?php
    $Seitennummer = $Seitennummer+1;
?>
...
... Seite <?php echo $Seitennummer; ?> von 10
...
```

Damit wir auf der nächsten Seite die Seitennummer wieder zur Verfügung haben, brauchen wir wieder das versteckte Eingabefeld, in dem wir aber diesmal nicht 1 sondern 2 vermerken. Praktischer Weise steht in der Variable `$Seitennummer` der 2er und so geben wir in das versteckte Eingabefeld diese Variable aus:

Seite2.php

```
...
<?php
    $Seitennummer = $Seitennummer+1;
?>
...
... Seite <?php echo $Seitennummer; ?> von 10
...
<input type="hidden" name="Seitennummer" value="<?php echo
$Seitennummer; ?>">
...
```

Auf der nächsten Seite (Seite3.php) steht in der Variable `Seitennummer` der Wert 2. Wir müssen also die Variable wieder um eins erhöhen, diese ausgeben und für die vierte Seite ein

verstecktes Eingabefeld erstellen. Zum Programmieren der dynamischen Seitenangaben brauchen wir also pro Seite 3 Befehle.

Randomisierte Zuweisung

Wir möchten jetzt die Teilnehmer zufällig in 2 Gruppen einteilen. Die erste Gruppe gelangt von der Startseite auf Seite2a.php, die zweite Gruppe gelangt auf Seite2b.php. Natürlich gibt es hier viele Möglichkeiten das zu programmieren. In diesem Beispiel verwenden wir das Datum zur Zuweisung. Wir ermitteln also die Sekunde in der der Teilnehmer die Seite aufruft (das sollte ein sehr zufälliges Maß sein) und überprüfen, ob es sich um eine gerade oder ungerade Zahl handelt. Je nach dem wird entweder Seite2a.php oder Seite2b.php aufgerufen. Wir benötigen dafür:

- eine Funktion, die das Datum liefert
- die Information wo die nächste Seite aufgerufen wird (das wissen wir bereits: beim action attribut des Form-Tags)
- den Operator mit dem man zwischen gerade und ungeraden Zahlen unterscheiden kann
- die Syntax für eine Abfrage

Funktion zum Ermitteln der Sekunde: date(„s“)

Operator zur Unterscheidung zw. gerade und ungeraden Zahlen: %

Abfrage: if (Bedingung) { ... then Teil } else { else Teil }

Startseite.php:

```
<?php
// Lässt sich die Uhrzeit (die Sekunden) durch 2 teilen
    if ((date("s")%2)==0)
    {
// wenn gerade (==0) dann merke Dir Seite2a.php
        $naechsteSeite="Seite2a.php"
    }
    else
    {
// sonst merke Dir Seite2b.php
        $naechsteSeite="Seite2b.php";
    }

// wenn die ersten beiden Zeichen des nachfolgenden Befehls
geloescht werden wird die ausgegeben welche Seite gemerkt
worden ist (nur zu Testzwecke!)
    // echo "Als nächste Seite wird
aufgerufen:$naechsteSeite";
?>
<html>
<head>
<title>Randomisierte Zuweisung</title>
</head>
<form name="f1" action="<?php echo $naechsteSeite;?>"
method="post" >
<body>
....
```

Adaptive Befragung

Ähnlich der Randomisierten Zuweisung ist auch eine adaptive Befragung zu realisieren (wenn man also aufgrund von Benutzereingaben auf andere Seiten verzweigen möchte).

Z.B. könnte man, wenn als Alter ein Wert kleiner 15 eingegeben wurde, gleich zur Endseite springen. Dafür kennen wir schon alle benötigten Elemente: Zugriff auf Benutzereingaben, Info wie die nächste Seite aufgerufen wird und eine Abfrage:

Demographischdaten.php:

```
<?php
// Lässt sich die Uhrzeit (die Sekunden) durch 2 teilen
    if ($alter<15)
    {
// wenn ja
        $naechsteSeite="Ende.php"
    }
    else
    {
// sonst
        $naechsteSeite="Seite3.php";
    }
// wenn die ersten beiden Zeichen des nachfolgenden Befehls
geloescht werden wird die ausgegeben welche Seite gemerkt
worden ist (nur zu Testzwecke!)
    // echo "Als nächste Seite wird
aufgerufen:$naechsteSeite";
?>
<html>
<head>
<title>Randomisierte Zuweisung</title>
</head>
<form name="f1" action="<?php echo $naechsteSeite;?>"
method="post" >
<body>
....
```

Programmbeispiele mit Java Script

Da Java Script auf dem Client ausgeführt wird kann man direkt auf die Benutzereingaben reagieren. Dafür benötigen wir das Event Handling und Java-Script-Anweisungen. Event Handling meint, dass wir - während die Seite angezeigt wird - bei bestimmten Ereignissen (=Events) Funktionen aufrufen können.

Für die Eingabeüberprüfung heißt das, dass wir - wenn der Event „Benutzer drückt auf Weiter-Button“ ausgelöst wird - eine kleine Javascript Funktion aufrufen werden, die die Eingabewerte überprüft. Diese Funktion müssen wir uns selbst schreiben. Sie beinhaltet die einfache Abfrage „wenn im Feld (z.B. Alter) nichts steht, dann gib eine Meldung aus, gehe zu dem Feld (z.B. Alter) hin und gehe nicht zur nächsten Seite (was ja normalerweise passiert wenn der Weiter-Button geklickt wird). Formell sieht das dann so aus:

Befehle	Bedeutung
if(document.f1.alter.value == "")	Wenn Alter leer ist

<pre>{ alert("Bitte geben Sie das Alter ein!"); document.f1.alter.focus(); return false; }</pre>	<p>führe alle Befehle ab hier gibt diese Meldung aus gehe zum Feld bleibe auf der Seite bis hier durch</p>
--	--

Ein HTML File könnte dann wie folgt aussehen. Alle Werte in gleicher Farbe müssen gleich sein.

```
<html>
<head>
<title>Eingabeüberprüfung</title>
</head>
<script language="JavaScript">
function CheckEnter()
{
  if(document.f1.alter.value == "")
  {
    alert("Bitte geben Sie das Alter ein!");
    document.f1.alter.focus();
    return false;
  }
  if(document.f1.alter.value > 70)
  {
    alert("Sie sind für diesen Fragebogen leider zu alt!");
    document.f1.alter.focus();
    return false;
  }
  return true;
}
</script>

<link rel="stylesheet" href="layout.css" type="text/css">
<form name="f1" action="Beispiel8.php" method="post"
onSubmit="return CheckEnter();" >
<body>

<p>Bitte machen Sie folgende Angaben:</p>
<table border=0 cellspacing=10>
<tr>
  <td>Alter:</td>
  <td><input type="text" name="alter" maxlength=2 size=2</td>
</tr>
<tr>
  <td>Geschlecht:</td>
  <td>
    <input type="radio" name="geschl" value="1">
    männlich<br>
    <input type="radio" name="geschl" value="2"> weiblich
  </td>
</tr>
</table>
```

```
<tr>
  <td>h&ouml;chste abgeschlossene Ausbildung:</td>
  <td>
    <select name="ausbild" size="1">
      <option value=1>Pflichtschule</option>
      <option value=2>Matura</option>
      <option value=3 >Universitätsstudium</option>
      <option value=4 >Doktoratsstudium</option>
    </select>
  </td>
</tr>
</table>
<input type=submit value="Absenden">
</body>
</form>
</html>
```

Anhand dieses Beispiels sieht man, dass Java Script Befehle entweder beim Event direkt (siehe im FORM – Tag `onSubmit="return CheckEnter();"`) oder mit einem Tag `<script language="JavaScript">` eingeleitet wird. Auf die Eingabefelder zugegriffen wird in Javascript mit `document.<name des Formulars>.<name des Eingabefeldes>.value`. Mit `alert(„TEXT“);` wird eine Meldung in einer kleinen Box ausgegeben.