Einführung in PHP

PHP – Scriptsprache für Webanwendungen

Dynamische Websites und PHP sind aus dem Internet nicht mehr wegzudenken. Im folgenden Kapitel machen wir Sie mit den Grundlagen von PHP vertraut.

Im Laufe des Buches wurde die Programmiersprache PHP schon mehrfach erwähnt. In den folgenden Kapiteln werden wir uns in erster Linie mit dieser Scriptsprache befassen und sie dafür zunächst etwas näher erläutern. Dieses Buch stellt jedoch keine vollständige Einführung in PHP dar. Wir werden nur sehr grob auf die nötigen Grundlagen eingehen. Für die Beispielübungen und -features sollte die jedoch ausreichen. Sie müssen in Dreamweaver MX 2004 nicht unbedingt mit dem Quelltext arbeiten. PHP-Kenntnisse in den Grundzügen sind jedoch eine Voraussetzung, um zu verstehen, wie Dreamweaver MX 2004 dynamischen Websites erstellt.

Was ist PHP?

PHP steht für Professional Hypertext Preprocessor und liegt aktuell in der Version 4.3.5 vor (Stand Januar 2004). PHP ist eine serverseitige Scriptsprache mit der speziellen Ausrichtung auf Webentwicklungen. Die Syntax ist an C++ angelehnt, jedoch wesentlich einfacher. PHP kann direkt in HTML-Dokumente eingebunden werden. PHP wird auf dem Webserver ausgeführt. Dazu wird der PHP-Interpreter benötigt. Damit wir mit PHP entwickeln können, ist es von großem Vorteil, wenn Sie PHP bzw. einen kompletten Webserver lokal auf Ihrem System installieren. In einem der folgenden Abschnitte werden wir detailliert beschreiben, wie das geht.

»Hallo Welt« in PHP

Zum hohen Verbreitungsgrad von PHP hat die breite und kostenlose Verfügbarkeit (Open Source) sowie die relativ einfache Erlernbarkeit geführt. Das unvermeidliche »Hallo Welt«-Script sieht in PHP folgendermaßen aus:

```
<html>
    <head>
        <title>PHP Script</title>
    </head>
    <body>
        <?php
        echo "Hallo Welt";
        ?>
    </body>
</html>
```

Listing 1 »Hallo Welt« in PHP

Wenn Sie im Browser den Quelltext betrachten, werden Sie feststellen, dass vom ursprünglichen Script nichts zu sehen ist. Stattdessen werden nur die HTML-Bestandteile ausgegeben:

```
<html>
    <head>
        <title>PHP Script</title>
    </head>
    <body>
Hallo Welt
    </body>
</html>
```

Listing 2 Aus dem PHP-Script generiertes HTML-Dokument

PHP wird nicht im Browser ausgegeben. Es ist in gewisser Weise vielmehr ein HTML-Generator, den man selbst programmieren kann. Eingefleischten PHP- Programmierern stehen bei dieser Definition sicherlich die Haare zu Berge. Um sich die Funktion von PHP zu verdeutlichen, ist es allerdings ein guter Vergleich.

Der Befehl `Echo` im Script bedeutet so viel wie »Gib mir zurück«, und zwar in diesem Fall »Hallo Welt«. Genau das macht dieses Script. Es gibt die Phrase »Hallo Welt« aus. Mehr passiert nicht in unserem ersten Beispiel.

Verzweifeln Sie nicht, wenn Sie versuchen, dieses Script lokal auszuführen. Das kann nicht gehen, solange wir keinen Webserver lokal installiert haben. Haben Sie bitte noch etwas Geduld, nach den Grundlagen werden wir einen Apache installieren (Seite XXX), und dann können Sie

auch alle Scripts nachvollziehen. Wenn Sie nicht so lange warten möchten, arbeiten Sie zuerst das entsprechende Kapitel durch, die Installation ist nicht schwierig.

Wenn Sie direkt auf einem Webserver bei Ihrem Provider arbeiten und die Dateien mit FTP übertragen, muss sichergestellt sein, dass PHP und MySQL auf diesem Server installiert sind. Fragen Sie am besten nach, ob das der Fall ist. Webaccounts mit PHP und MySQL gibt es heute auch schon bei Billiganbietern, so dass die Kosten keine Rolle mehr spielen sollten.

Mit PHP ohne MySQL können Sie bereits einige Funktionen in Ihrer Website unterbringen. So können Sie z. B. Ihre User durch umfangreiche Berechnungen führen. So richtig interessant wird PHP aber erst, wenn Daten gespeichert, und später jederzeit wieder zum Abruf bereitstehen. Grundsätzlich unterstützt PHP dafür nahezu jede auf dem Markt vertretene Datenbank. Für unsere Zwecke haben wir uns wegen der hohen Verbreitung und der Leistungsfähigkeit für MySQL entschieden. Zudem ist es kostenlos unter http://www.mysql.com erhältlich.

Vorteile von PHP

Viele Vorteile sprechen für den Einsatz von PHP. Serverseitige Scripts, die lange nur mit Kenntnissen komplexer Programmiersprachen wie Perl möglich waren, kann man nun auch ohne große Vorkenntnisse in der Programmierung relativ schnell selbst realisieren:

- Wer jemals in Basic oder anderen Programmiersprachen entwickelt hat, wird feststellen, dass PHP sehr ähnlich ist. Erste Scripts sind sehr schnell erstellt, und die Syntax ist einfach zu verstehen. PHP ist zudem eine sehr tolerante Programmiersprache. Sie müssen keine Datentypen deklarieren und sich nicht mit verschiedenen Formaten herumschlagen.
- Für Webapplikationen zählt PHP zu den am weitesten verbreiteten Programmiersprachen überhaupt.
- Es gibt Unmengen an Open Source-Projekten zu PHP. Im Web finden Sie viele veröffentlichte PHP-Projekte, egal ob Sie ein Forum, ein Shopsystem oder ein WCMS suchen. Mit PHP werden Sie mit Sicherheit fündig.
- PHP ist bestens geeignet für dynamische Websites. Es ist für den Webeinsatz eingeführt worden und bietet als Open-Source-Standard alle notwendigen Funktionen an.

- Nahezu alle Datenbanken werden mittlerweile von PHP unterstützt.
- PHP enthält sehr umfangreiche Bibliotheken für nahezu jeden Anwendungsbereich.

PHP und HTML

PHP in HTML einbinden

PHP-Scripts können an beliebiger Stelle im HTML-Quelltext eingebunden werden. Das geschieht mit dem Tag:

```
<?php HIER STEHT DAS SCRIPT; ?>
```

Alternativ können Sie auch einfach schreiben:

```
<? HIER STEHT DAS SCRIPT; ?>
```

Es funktioniert beides. Jede PHP-Befehlszeile wird mit einem Semikolon abgeschlossen. Die Ausgabe eines einfachen Satzes sehen Sie z. B. in unserem »Hallo Welt«-Script.

PHP können Sie an jeder beliebigen Stelle und so oft im HTML-Dokument einbinden, wie Sie wollen. Das funktioniert auch innerhalb eines HTML-Tags. So können Sie beispielsweise die Hintergrundfarbe einer Tabelle aus einer PHP-Variablen generieren:

```
<table bgcolor="<? Echo "$farbe" ?>">
```

An den Browser wird anstelle des ganzen Befehles nur der Inhalt der Variablen `$farbe` ausgegeben. Der PHP-Befehl wird bereits auf dem Server ausgeführt. In unserem Beispiel muss er nicht mit einem Semikolon abgeschlossen werden, da es sich hier nur um eine einzige Befehlszeile handelt.

Schreibweise von Zahlen und Zeichen

Bei PHP werden zwar nicht detaillierte Datentypen, wie ganze Zahlen oder Fließkommazahlen, vorgegeben. Es wird jedoch in einem Script zwischen Zeichen (Strings) und Zahlen unterschieden.

Variablen in der einfachen Schreibweise werden auch als Zahl behandelt. `<? Echo 100 ?>` erzeugt die Ausgabe der Zahl 100. Stehen Variablen oder Zahlen in Anführungszeichen, versteht PHP sie als Zeichenketten. So gibt `<? Echo "100" ?>` die Zeichenkette 100 aus. Mit Zeichenketten können keine Berechnungen durchgeführt werden.

Auch Vermischungen von Zeichenketten (Strings) und nummerischen Werten innerhalb eines Befehles sind möglich und werden häufig eingesetzt. Dafür müssen Sie dem PHP-Interpreter mitteilen, welcher Teil des Befehles als Zeichenkette und welcher als Zahl zu behandeln ist. Bei PHP bewirkt ein Punkt die Addition von Zeichenketten:

```
<? Echo "Bitte zahlen Sie". 100 ." Euro"; ?>
```

Mit dieser Schreibweise können Sie Zeichen und Zahlen innerhalb eines einzigen Befehles ausgeben. Als Beispiel soll eine Rechnungssumme ausgegeben und die Bezeichnung Euro hinter den Rechenwert gesetzt werden.

```
<? Echo "Bitte zahlen Sie". $rechnung ." Euro"; ?>
```

Würden Sie diese Zeichenaddition nicht vornehmen, gäbe PHP eine Fehlermeldung aus. Die Funktionsweise dieser Befehlszeile ist folgendermaßen zu verstehen:

`<?` (Jetzt kommt ein Befehl für den PHP-Interpreter)
`Echo` (Gib Folgendes aus:)
`Anführungszeichen` (Jetzt kommen Zeichen)
`Bitte zahlen Sie`
`Anführungszeichen` (Jetzt hören die Zeichen auf)
`Punkt` (Hänge das, was als Nächstes kommt, an das Vorherige)
`$rechnung` (Eine Zahl)
`Punkt` (Hänge das, was als Nächstes kommt, an das Vorherige)
`Anführungszeichen` (Jetzt kommen Zeichen)
`Euro`
`Anführungszeichen` (Jetzt hören die Zeichen auf)
`Semikolon` (Befehlszeile ist jetzt zu Ende)
`?>` (Hier ist das PHP-Script zu Ende – weiter mit HTML)

Kommata in Berechnungen

Achten Sie bei Berechnungen auf die korrekte Schreibweise des Fließkommas. Ein Komma in einer Zahl muss in PHP als Punkt geschrieben werden. Falsch ist etwa 3,14 * 300, richtig 3.14 * 300. Besonders wichtig ist diese korrekte Schreibweise bei Berechnungen durch Benutzereingaben. Der User weiß nicht, wie er eine Zahl schreiben muss. Daher muss eine Benutzereingabe für Berechnungen immer abgefangen und auf falsche Kommasetzung überprüft werden. Am einfachsten ist es, Kommata mit einem Script in Punkte umzuwandeln.

HTML in PHP einbinden

Neben der Einbindung von PHP in HTML ist es natürlich auch möglich, HTML in PHP einzubinden. Mit dem Befehl `Echo` können Sie komplette HTML-Zeilen ausgeben:

```
<? Echo "<table><tr><td> </td></tr></table>"; ?>
```

Diese Befehlszeile gibt eine Tabelle aus. Um PHP jetzt zu veranlassen, auch die für Attribute notwendigen Anführungszeichen auszugeben, müssen Sie die folgende Schreibweise anwenden:

```
<? Echo "<table bgcolor=\"#333366\"><tr><td> </td></tr>
</table>"; ?>
```

Der Backslash verhindert, dass der PHP-Interpreter das Anführungszeichen als PHP-Befehl interpretiert. Durch ihn wird mitgeteilt, dass das nachfolgende Zeichen einfach als Zeichen zu interpretieren ist.

Variablen in PHP

Wie Sie gesehen haben, werden Variablen mit einem vorangestellten Dollarzeichen $ gekennzeichnet. Es gibt in PHP mehrere Möglichkeiten, mit Variablen zu arbeiten.

Wie oben erwähnt, müssen Sie sich bei PHP nicht um die Deklaration von Variablen kümmern. Dennoch gibt es Einiges zu beachten: Im Internet werden Variablen mit den im Kapitel »Formulare«(Seite XXX)

beschriebenen Aktionen GET und POST übermittelt. Möchten Sie zum Beispiel ein Dokument aufrufen und gleichzeitig eine Variable übertragen, um zum Beispiel die ID eines Datensatzes zu übergeben, sieht die Angabe in der Adressleiste des Browsers folgendermaßen aus:

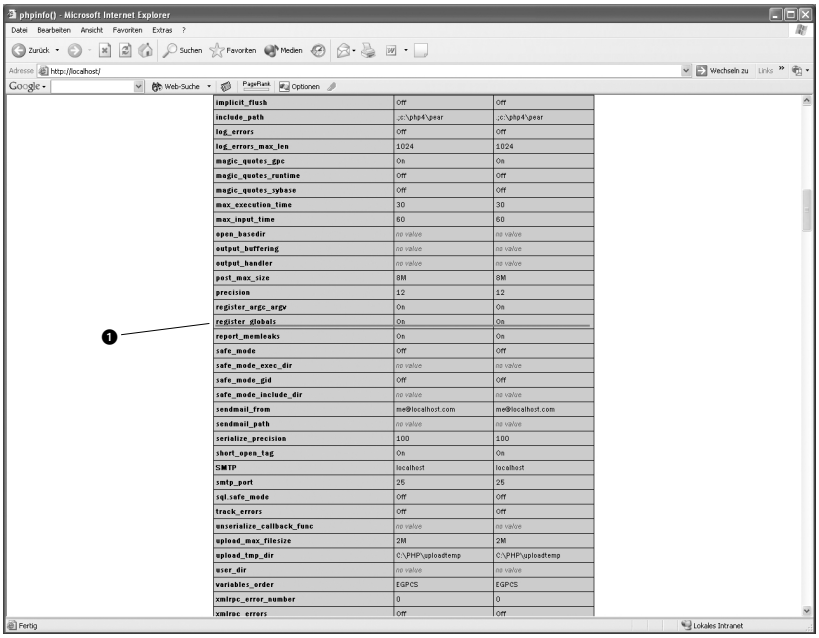
```
http://www.website.de/produkte.php?PRODID=234
```

In diesem Fall heißt die Variable `PRODID` und hat den Inhalt 234. Diese Variable können Sie sich im Dokument produkte.php mit `echo $PRODID` ausgeben lassen oder andere Aktionen damit auslösen. Das Fragezeichen in der URL bedeutet sinngemäß: Jetzt kommen Variablen.

Mehrere Variablen in einer URL werden mit `&` verkettet:

```
http://www.website.de/produkte.php?PRODID=234&SUBID=2
```

Bei der einfachen Schreibweise: $Variablenname spielt die Art der Übertragung keine Rolle. Unter PHP 4.2.0 oder höher funktioniert diese Schreibweise nur noch innerhalb eines Dokumentes, nicht jedoch, wenn die Variablen übertragen werden.



▲ **Abbildung 1**
Eintrag in php.ini REGISTER_GLOBAL

In PHP 4.2.0 ist die Standardanweisung in der php.ini REGISTER_GLOBALS off (Abbildung 1). Wenn Sie dennoch mit der bisherigen Schreibweise der Variablen arbeiten möchten, editieren Sie die php.ini und setzen Sie die Vorgabe auf REGISTER_GLOBALS on. Welche Einstellungen bei Ihnen vorliegen, können Sie mit phpinfo() überprüfen ❶.

Falls Sie keinen eigenen Webserver betreiben, werden Sie die php.ini nicht bearbeiten können. Verwenden Sie daher am besten die neuere Schreibweise für Variablen. Auch Dreamweaver MX 2004 benutzt diese, so dass Sie sicher sein können, dass Ihre Websites auf allen Servern lauffähig sind.

PHP überträgt Variablen zwischen URLs in Arrays. Sie werden als superglobale Arrays bezeichnet.

Die folgende Variable beinhaltet einen Verweis zu jeder Variable im laufenden Script:

```
$GLOBALS
```

Dieses ist das Array mit den Servervariablen:

```
$_SERVER
```

Das folgende wichtige Array beinhaltet alle Variablen, die mit HTTP GET übertragen wurden:

```
$_GET
```

Die einzelnen Variablen können ausgelesen werden über:

```
$_GET['VARIABLENNAME']
```

Das folgende wichtige Array beinhaltet alle Variablen, die mit HTTP POST übertragen wurden:

```
$_POST
```

Die einzelnen Variablen können ausgelesen werden über:

```
$_POST['VARIABLENNAME']
```

Das folgende Array beinhaltet Cookie-Variablen:

```
$_COOKIE
```

Dieses Array enthält alle Variablen, die dem Script über HTTP Post-Datei-Uploads angeliefert werden:

```
$_FILES
```

Das folgende Array enthält alle Umgebungsvariablen:

```
$_ENV
```

Dieses Array enthält alle Variablen, die auf anderen Wegen in das Script gelangen und keiner der gängigen Sicherheitsanforderungen entsprechen:

```
$_REQUEST
```

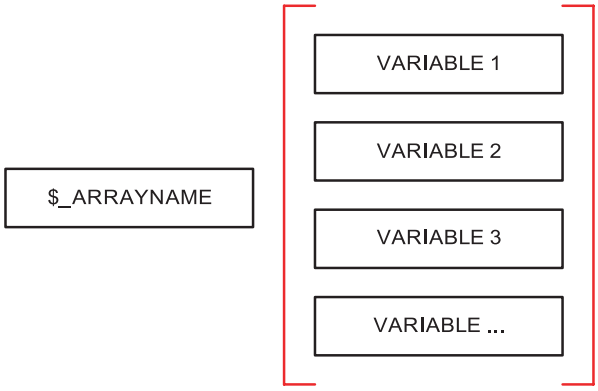
Variablen, die aktuell in der Session eines Scripts registriert sind, enthält das folgende Array. Mehr dazu erfahren Sie im Teil über Sessions in diesem Kapitel:

```
$_SESSION
```

Jedes dieser Arrays setzt sich nach dem Schema in Abbildung 2 zusammen.

Vergleichsoperatoren

Schleifendurchläufe und Bedingungen sind die grundlegenden Elemente jeder Programmiersprache. Für beides benötigen Sie die Möglichkeit, Daten zu vergleichen, um Aktionen durch die daraus hervorgehenden Ergebnisse zu steuern. PHP unterstützt viele Arten des Datenvergleichs. Vergleichsoperatoren können auch mit booleschen Funktionen verknüpft werden.



▲ **Abbildung 2**
Array-Variablen

Das Ergebnis eines Vergleiches ist immer `true` oder `false`. Wird eine Bedingung erfüllt, also `true`, wird die nachfolgende Aktion im Script ausgeführt:

Die wichtigsten Vergleichoperatoren werden in der folgenden Tabelle aufgeführt:

Vergleichsoperatoren in PHP

A == B	Bedingung ist erfüllt (gibt true zurück), wenn A und B gleich sind. Dieser Operator darf nicht mit dem mathematischen Gleichzeichen verwechselt werden.
A != B	Bedingung ist erfüllt (gibt true zurück), wenn A und B ungleich sind.
A >= B	Bedingung ist erfüllt (gibt true zurück), wenn A größer oder gleich B ist.
A <= B	Bedingung ist erfüllt (gibt true zurück), wenn A kleiner oder gleich B ist.
A > B	Bedingung ist erfüllt (gibt true zurück), wenn A größer B ist.
A < B	Bedingung ist erfüllt (gibt true zurück), wenn A kleiner B ist.
A === B	Bedingung ist erfüllt (gibt true zurück), wenn A und B identisch sind.
A !== B	Bedingung ist erfüllt (gibt true zurück), wenn A und B nicht identisch sind.

Boolesche Operatoren

Mit booleschen (logischen) Operatoren können Sie z. B. die Ergebnisse einzelner Vergleichsoperationen verknüpfen.

Beispiel: Die Bedingung soll erfüllt sein, wenn A gleich B ist oder C gleich D ist. Beide Vergleichsoperatoren werden verknüpft mit der ODER-Verknüpfung:

```
(A==B) || (C == B)
```

Eine Übersicht der booleschen Funktionen bietet Ihnen die folgende Tabelle:

Boolesche Funktionen

A and B	UND-Verknüpfung: Bedingung ist erfüllt, wenn sowohl A als auch B wahr sind.
A or B	ODER-Verknüpfung: Bedingung ist erfüllt, wenn A oder B wahr sind.
A xor B	ENTWEDER oder // EXKLUSIV-ODER-Verknüpfung: Bedingung ist erfüllt, wenn A oder B wahr sind, aber nicht beide.
! A	NICHT-Verknüpfung: Bedingung ist erfüllt, wenn A nicht wahr ist.
A && B	UND-Verknüpfung: Bedingung ist erfüllt, wenn sowohl A als auch B wahr sind.
A || B	ODER-Verknüpfung: Bedingung ist erfüllt, wenn A oder B wahr sind.

Schleifen programmieren

Schleifendurchläufe sind die mit am häufigsten genutzten Sprachelemente. In nahezu jedem PHP-Script sind Schleifendurchläufe enthalten, etwa zum Ausgeben mehrerer Datensätze einer Tabelle. Auch wenn Sie später mit Dreamweaver MX 2004 Bereiche wiederholen, werden Schleifen verwendet. Schleifen laufen immer so lange durch, bis die Schleifenbedingung erfüllt ist.

for-Schleifen

Wir beginnen mit einer for-Schleife, die relativ komplex aufgebaut wird:

```
for ($x = 1; $x <= 10; $x++)
{
    echo $x;
}
```

Listing 3 for-Schleife

Diese Schleife führt die Befehle in den geschweiften Klammern so lange aus, bis `$x` den Wert 10 erreicht hat und damit die Schleifenbedingung erfüllt ist. Geschweifte Klammern umschließen immer einen Codeblock, der bei einer erfüllten Bedingung abgearbeitet wird.

Mit der Schleife aus Listing 3 kann man zum Beispiel zehn Datensätze ausgeben lassen. Über `$x` steht die Anzahl der durchlaufenen Schleifen als Variable zur Verfügung, mit der man zusätzlich arbeiten kann.

Die Schreibweise am Ende der Schleifenbedingung `$x++` ist eine vereinfachte Schreibweise von `$x = $x + 1`.

++ bedeutet, dass der Wert um 1 inkrementiert, also erhöht wird. Würden wir ein – einsetzen, würde der Wert um 1 verringert.

Schleifen und später auch Bedingungen bauen sich nach folgendem Schema auf:

```
Schleifen- oder Bedingungstyp ( Bedingung )
   {
   Auszuführender Codeblock bei erfüllter Bedingung
   }
```

Listing 4 Prinzip einer Schleife

while-Schleifen

Eine while-Schleife ist einfacher aufgebaut als eine for-Schleife:

```
$x = 1;
while ($x <= 10)
{
    echo $x++;
}
```

Listing 5 Einfache while-Schleife

`$x` wird hier so lange ausgegeben, bis der Wert 10 erreicht ist. Der Wert der Variablen `$x` wird in der Schleife bei jedem Durchlauf um 1 erhöht.

do...while-Schleifen

do...while-Schleifen sind den while-Schleifen sehr ähnlich:

```
$x = 0;
do
```

```
{
    echo $x;
}
while ($x>0);
```
Listing 6 do…while-Schleife

Der Unterschied liegt darin, dass bei diesen Schleifen das Erfüllen der Bedingungen nicht am Anfang der Schleife, sondern erst am Ende eines Durchlaufes überprüft wird. Somit kann man sicherstellen, dass die Schleife in jedem Fall mindestens einmal durchlaufen wird. Bei der while-Schleife kann es vorkommen, dass sie nie durchlaufen wird.

foreach-Schleifen

foreach-Schleifen ermöglichen die einfache Ausgabe von Arrays:

```
foreach ($array as $ausgabe)
{
    echo "aktueller Inhalt: $ausgabe";
}
```
Listing 7 foreach-Schleife

Diese Schleife funktioniert ausschließlich mit Arrays. Inhalte des Arrays werden einer neuen Variablen, in unserem Fall `$ausgabe`, zugewiesen und ausgegeben. Beim nächsten Schleifendurchlauf erhöht sich der Index des Arrays um 1, und der nächste Wert des Arrays wird zugewiesen und ausgegeben. Das geschieht, solange es Inhalte im Array gibt.

Alle hier beschriebenen Schleifen können in nahezu beliebiger Tiefe verschachtelt werden.

Bedingungen mit PHP

Eine der wichtigsten Anweisungen in der Programmierung überhaupt, PHP eingeschlossen, ist if. Mit dieser einfachen Anweisung und einer nachfolgenden Bedingung können Programmabläufe kontrolliert und beeinflusst werden. Der prinzipielle Aufbau ist denkbar einfach (Listing 8):

if

```
if ($A Vergleichsoperator $B)
{
Führe Folgendes aus
}
```

Listing 8 if-Bedingung

Wenn (`if`) die Bedingung erfüllt ist, wird der Inhalt zwischen den geschweiften Klammern ausgeführt.

if else

Eine Abwandlung davon ist:

```
if ($A Vergleichsoperator $B)
{
Führe Folgendes aus
}
else
{
Ansonsten mache das
}
```

Listing 9 if else-Bedingung

Mit dem Zusatz `else` wird sichergestellt, dass im Falle der Nichterfüllung einer Bedingung in der if-Anweisung das ausgeführt wird, was sich in der else-Anweisung befindet.

switch

Mit switch-Anweisungen kann man sehr elegant und komfortabel, je nach Inhalt einer Variablen, zwischen mehreren Möglichkeiten auswählen (umschalten, also switch).

```
switch ($x)
{
       case 0:
           echo "Inhalt 0";
           break;
```