Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Dr. Jürg M. Stettbacher
Margrit Rainer Strasse 12a
CH-8050 Zürich

Telefon: +41 43 299 57 23
Fax: +41 43 299 57 25
E-Mail: dsp@stettbacher.ch

Kurze Einführung in PHP

Vorbereitung für Übungen in INCO

Version 2.01
2013-06-12

Zusammenfassung: PHP ist eine komfortable, weit verbreitete Skript-Sprache. Sie wird heute oft im Zusammenhang mit Web-Applikationen verwendet, ist aber nicht darauf beschränkt. Ihre Syntax ist an die Programmiersprache C angelehnt, jedoch verfügt sie über einige mächtige Erweiterungen. Eine solche Erweiterung sind assoziative Arrays, welche bei einigen unserer INCO-Praktikas besonders nützlich sind. Dieses Dokument gibt eine kurze Einleitung in PHP.

Inhaltsverzeichnis

1 Einleitung

PHP [1] ist eine Skript-Sprache, die heute oft für Webseiten verwendet wird. Die Skripte können aber auch direkt in einer Shell[1] ausgeführt werden. Die Sprache ist sehr mächtig und besitzt einige Eigenschaften, die für die Anwendung in der Informationstheorie sehr nützlich sind. Falls Sie mit PHP noch nicht vertraut sind, so spielen Sie die Beispiele in diesem Dokument durch.

Im Folgenden setzen wir die PHP Version 5 oder höher voraus. Die Referenz ist unter [1] zu finden. Eine Einleitung in grundsätzliche Konzepte, z. B. der Umgang mit Arrays oder mit Strings, ist zu finden unter [1] *Documentation* (Menü-Zeile oben) → *English* (blaue Tabelle) → *Types* (Inhaltsverzeichnis). Erklärungen zu bestimmten Funktionen findet man, indem man direkt den Funktionsnamen, z. B. *strlen*, oben rechts in [1] eintippt und danach suchen lässt. Anschliessend findet man Verweise auf verwandte Funktionen und Themen links auf der Seite.

PHP schliesst ab Version 5 auch objektorientierte Konzepte ein. Diese werden für die Praktikas in INCO aber nicht benötigt.

[1] Eine Shell ist typischerweise ein Programm, das einen textbasierten Zugang zum System bietet. Oft ist es ein Fenster mit Prompt, wo Befehle eingetippt werden können. Unter Linux und Mac OS X werden häufig *bash* (Bourne-Again-Shell), *ksh* (Korn Shell) oder *csh* (C-Shell) verwendet. Unter Windows steht *cmd* (Windows Kommandozeile) oder *powershell* (Windows PowerShell) zur Verfügung.

2 Installation

Falls Sie auf Ihrem Rechner Linux als Betriebsystem verwenden, so ist PHP meist bereits vorhanden oder kann als Paket nachinstalliert werden. Unter OpenSuSE beispielsweise heisst das Paket *php5*. Den Source Code, sowie Binaries für Windows finden Sie unter [2] resp. [3]. Alternativ ist PHP auch Teil des XAMPP Projekts [4] und kann auf diesem Weg auf Linux, Mac und Windows installiert werden. Beachten Sie, dass in diesem Fall (mindestens auf Windows) von Hand der Pfad korrekt ergänzt werden muss, damit PHP auch auf der Shell genutzt werden kann.

Zum Testen der Installation öffnen Sie eine Shell und tippen Sie:

```
> php --version
```

Die Installation ist in Ordnung, wenn Sie eine Ausgabe ähnlich dieser erhalten:

```
PHP 5.3.3 (cli)
Copyright (c) 1997-2010 The PHP Group Zend Engine v2.3.0,
Copyright (c) 1998-2010 Zend Technologies
```

3 Erstes Skript

Verwenden Sie zum Schreiben Ihrer Skripte einen ASCII Text-Editor[2]. Falls Sie einen Unicode Text-Editor benutzen möchten, so achten Sie darauf, dass Sie keine Umlaute und keine sonstigen Sonderzeichen gebrauchen. Öffnen Sie also eine Datei *first.php* und geben Sie den folgenden Text ein.

```
<?php
    echo "\n";
    echo "Hello World.\n";
    echo "\n";
?>
```

Speichern Sie die Datei ab und führen Sie in einer Shell im betreffenden Verzeichnis den folgenden Befehl aus:

[2] In einem Text-Editor schreibt man unformatierten Text. Die einzigen Gestaltungsmöglichkeiten sind Leerschläge, Zeilenumbrüche, Leerzeilen und Tabulatoren. Es werden nur die tatsächlich getippten Zeichen gespeichert. Editoren gibt es haufenweise für alle Systeme. Beispiele für Linux sind *kate* (für KDE) oder *gedit* (für Gnome), für Mac OS X gibt es *textedit* und für Windows *notepad*. Beachten Sie, dass Office-Pakete nicht geeignet sind, da sie im Normalfall Formatierungen mit dem getippten Text speichern.

```
> php first.php
```

Das Skript wird nun ausgeführt und sagt *Hello World* in der Shell. Alles andere wäre ein Fehler.

4 Wichtige Sprachelemente

Wir wollen hier nur einen kurzen Überblick über die wichtigsten Sprachelemente geben. Eine ausführliche Referenz gibt es unter [1].

Kommentare Die Kommentare sehen gleich aus wie in C/C++.

```
// Das ist ein einzeiliger Kommentar.
/* Das ist auch ein Kommentar, aber
   er ist mehrzeilig und geht bis hierher. */
```

Variablen Datentypen müssen nicht deklariert werden. Neue Variablen lassen sich an jeder Stelle im Programm einführen.

```
$nr  = 1;
$pi  = 3.14;
$str = "PI = ";
echo $nr." --> ".$str.$pi."\n";
```

Der Befehl *echo* schreibt auf den Bildschirm. Variablen jeden Typs werden dabei automatisch in Strings umgewandelt. Mehrere Strings können über Punkte miteinander verbunden werden. Der abschliessende String "\n" bewirkt einen Zeilenumbruch. Das vorherige Beispiel gibt dies auf dem Bildschirm aus:

```
1 --> PI = 3.14
```

Es gibt in PHP einige spezielle Variablen, die immer vorkommen:

- *$argc* - die Anzahl Argumente, die an das Skript übergeben wurden. Die Variable hat immer mindestens den Wert 1, da der Name des Skripts das erste Argument ist.
- *$argv* - ein Array (siehe unten) das alle Argumente enthält, die an das Skript übergeben wurden. $argc[0] ist der Name des Skripts.

Arrays In unseren Praktikas spielen Arrays eine ganz wichtige Rolle. Wir verwenden sie beispielsweise um ein Histogramm aufzubauen. PHP kennt sog. assoziative Arrays. Das heisst, dass der Index zu einem Arrayeintrag nicht einfach nur ein Integer sein muss, sondern es können beliebige Strings als Index, resp. Schlüssel dienen. Hier sind einige Beispiele.

```
// Einfaches Array:
$fruits = array("apfel", "birne", "banane");
$fruits[2] = "ananas";  // Ersetzt 'banane'.

// Array durchlaufen:
$N = count($fruits);  // Laenge des Arrays.
for ($n=0; $n<$N; $n++) {
    echo $n." --> ".fruits[$n]."\n";
}

// Elemente manipulieren:
$fruits[] = "zitrone"; // Hinzufuegen zum Array.
unset($fruits[1]);     // Loeschen aus Array.

// Noch eine Methode um Arrays zu durchlaufen:
foreach ($fruits as $fruit) {
    echo $fruit." ";
}


// Assoziatives Array:
$deutsch = array( "apfel"  => "Apfel",
                  "birne"  => "Birne",
                  "banane" => "Banane" );
$english = array( "apfel"  => "Apple",
                  "birne"  => "Pear",
                  "banane" => "Banana" );

// Assoziative Array durchlaufen:
foreach ($deutsch as $key => $value) {
    echo "Deutsch: ".$value."  -->  English: ".$englisch[$key]."\n";
}
```

Hier sind einige wichtige Funktionen im Zusammenhang mit Arrays genannt:

- *array()* - ein Array leeren oder anlegen.
- *unset()* - ein Array oder ein Element eines Arrays löschen.
- *isset()* - prüft ob eine Variable oder ein Arrayelement existiert.
- *array_push()* - ein Element am Ende das Arrays hinzu fügen.
- *array_pop()* - das letzte Element eines Arrays lesen.

- *array_reverse()* - Elemente eines Arrays in umgekehrter Reihenfolge anordnen.
- *sort()* - ein Array nach Werten sortieren.
- *ksort()* - ein Array nach Schlüsseln sortieren.
- *array_keys()* - alle Schlüssel eines Arrays extrahieren.
- *array_key_exists()* - prüfen, ob ein Index existiert.
- *array_values()* - alle Werte eines Arrays extrahieren.
- *array_search()* - ein Array nach einem Wert durchsuchen, liefert den Schlüssel.
- *array_sum()* - liefert die Summe aller Werte im Array.
- *count()* - gibt die Anzahl Elemente eines Arrays an.

Strukturen PHP bietet alle bekannten Strukturelemente. Die *for* Schleife wurde oben schon gezeigt. Hier folgen einige weitere.

```
if ($a > 0) { ... }
elseif ($a < 0) { ... }
else { ... }

while ($a < 10) { ... }

do { ... } while ($a < 10);

switch ($a) {
case 1: ...
        break;
case 2: ...
        break;
default: ...
}
```

Funktionen In PHP gibt es hunderte von Funktionen. Eine ausführliche Referenz gibt [1], wo oben rechts im Suchfenster direkt nach einer Funktion gesucht werden kann. Suchen Sie etwa nach Funktionen für das Handling von Dateien, so geben Sie zum Beispiel *fopen* ein. Beachten Sie, dass viele Funktionen gleich oder ähnlich heissen wie in C/C++. Als Resultat erhalten Sie eine genaue Beschreibung der Funktion. Auf der linken Seite finden Sie Verweise auf andere Funktionen, die mit *fopen* irgendwie verwandt sind. Am unteren Ende der Beschreibung gibt es ausserdem Verweise auf näher verwandte Funktionen.

Im folgenden Beispiel wird gezeigt, wie aus einer Datei zeichenweise gelesen wird.

```
// Datei zum Lesen oeffnen:
$handle = fopen("filename", "r");

// Zeichenweise bis zum Dateiende lesen:
while (!feof($handle)) {
    $ch = fgetc($handle);
    ...
}

// Datei schliessen:
fclose($handle);
```

Literatur

[1] http://www.php.net

[2] http://www.php.net/downloads.php

[3] http://windows.php.net/download/

[4] http://www.apachefriends.org/