Moderne Webtechnologien I (WT 1)
Übung: Grundlagen PHP

WiSe 23/24

Prof. Dr. C. Köhn
Daniela Böing

15. November 2023

Website:
https://www.hochschule-bochum.de/fbe/fachgebiete/institut-fuer-informatik/labor-fuer-medienkommunikation-internet-und-robotik/

Hochschule Bochum
Bochum University
of Applied Sciences

MIRO LAB

Inhaltsverzeichnis

1 Grundlagen PHP

1.1 Einführung

PHP ist immer noch die am meisten eingesetzte Sprache für die serverseitige Webentwicklung. Seit November 2020 liegt die Skriptsprache in der Version 8 vor. PHP bietet volle Objektorientiertheit, wie sie Ihnen aus Java geläufig sein sollte. In diesem Übungsteil werden Sie die grundlegenden Elemente von PHP kennenlernen und den Aufbau der Skriptdateien.

Zur Umsetzung der Übungen wird ein Webserver und später auch eine Datenbank benötigt. Sie können dies beliebig umsetzen, zum Beispiel mit XAMPP unter Windows oder Docker auf allen Betriebssystemen. Der hier vorgeschlagene Weg ist mittels **Vagrant**, worüber ein sog. LAMP[1]-Stack erzeugt und zur Verfügung gestellt wird.

[1] LAMP steht für **L**inux **A**pache **M**ySQL and **P**HP, siehe `http://www.apachefriends.org`

Übung 1:

Sie sollten einen Webserver mit PHP und ab der nächsten Übung mit MySQL und PHP-PDO zur Verfügung haben. Wenn Sie wie vorgeschlagen mit Vagrant arbeiten wollen, gehen Sie dazu wie folgt vor:

1. Installieren Sie VirtualBox von Oracle: `https://www.virtualbox.org/wiki/Downloads` (prinzipiell sind auch andere OS-spezifische Virtualisierungsumgebungen möglich) sowie Vagrant: `https://www.vagrantup.com/downloads` (**Auf den Laborrechnern sollte bereits alles installiert sein.**)
2. Downloaden Sie die Vorgabedateien in ein Verzeichnis, das dann Ihr Arbeitsverzeichnis sein wird.
3. Navigieren Sie mit einem Command Prompt in den Vagrant-Ordner und geben Sie folgendes Kommando ein:

```
1 vagrant up
```

4. Legen Sie nun im selben Verzeichnis einen Unterordner *public* an. Dies ist Ihr Webserververzeichnis, in dem alle Dateien für die „Anzeige“ liegen sollten.
5. Erstellen Sie testweise eine einfache PHP-Datei *index.php* mit folgendem Inhalt:

```
1 <?php phpinfo(); ?>
```

6. Rufen Sie nun die IP `192.168.63.10` auf. Wenn alles funktioniert hat, können Sie darunter Ihre Vagrantbox und somit das Webverzeichnis aufrufen. (Die IP finden Sie in der Vagrantfile)
7. Wenn Sie mit der Übung fertig sind, fahren Sie die Vagrantbox wieder mit folgendem Kommand herunter:

```
1 vagrant halt
```

1.2 PHP einbinden

1.2.1 In-Line-Elemente

Sie können HTML und PHP-Anweisungen beliebig mischen und auch HTML-Ausgaben durch PHP erzeugen. Üblicherweise beginnt ein PHP-Abschnitt mit `<?php` und endet mit `?>`.

```
1 <p>Dieser Teil ist nicht PHP</p>
2 <?php
```

```
3     //Anweisungen
4     echo '<p>Eine Ausgabe durch PHP!</p>';
5  ?>
6  <p>Dieser auch nicht!</p>
```

Für einfache Ausgaben kann auch eine noch simplere Syntax verwendet werden. So könnte obiges Beispiel auch so aussehen:

```
1  <p>Dieser Teil ist nicht PHP</p>
2  <?="<p>Eine Ausgabe durch PHP!</p>" ?>
3  <p>Dieser auch nicht!</p>
```

1.2.2 Einbinden externer Skripte

Die Einbindung externer Skripte ist folgendermaßen möglich:

```
1  <?php
2     include ("Datei.php"); //Einsatz von include
3
4     // Nutzen von Require...
5     require 'funktionen.php';
6     require ('funktionen.php');
7  ?>
```

Dabei fügt `include()` an der Stelle des Aufrufes den Inhalt der Datei ein, der dann zur Verfügung steht. Der inkludierte Inhalt wird als HTML interpretiert, weshalb Sie externe Skripte jeweils wieder mit `<?php ...` einleiten müssen.

`require()` funktioniert ähnlich wie `include()`, jedoch ist der wichtigste Unterschied, dass `require()` das PHP-Skript mit einem *Fatal Error* abbricht, wenn die Datei nicht vorhanden sein sollte.

1.3 Grundlegende Syntax

Es folgt ein Beispiel zur Syntax in PHP:

```
1  <?php
2     function ausgabe(){
3        return 23;
4     }
5     $myVariable = ausgabe();
6     echo $myVariable;
7  ?>
```

Zunächst wird eine Funktion `ausgabe()` definiert, die den Wert 23 zurückgibt. Die Funktion wird dann aufgerufen und der Rückgabewert in einer Variablen namens `myVariable`

gespeichert (alle Variablen in PHP beginnen mit einem Dollarzeichen). Anschließend wird der Inhalt durch die `echo`-Anweisung auf dem Bildschirm ausgegeben.

Weiterhin kennt PHP die booleschen Konstanten `TRUE` (`1`) und `FALSE` (`0`). Ein Ausdruck ist in PHP dann wahr, wenn ihm, wie im folgenden Beispiel, die boolesche Konstante `TRUE` oder ein anderer Ausdruck, dessen Wert `TRUE` ist, zugewiesen wurde.

In PHP gibt es natürlich auch die Möglichkeit von Verzweigungen und Schleifen:

```
1  if ($cond1){
2     //mache etwas
3  }
4  elseif($cond2){
5     //$cond1 ist FALSE, aber $cond2 ist TRUE
6     //mache etwas anderes
7  }
8  else{
9     //$cond1 und $cond2 sind FALSE
10 }
11
12 // Schleifen - Beispiele
13 for ($i = 1; $i <= 5; $i++){
14    echo "$i<br />";
15 }
16
17 while(TRUE){
18    echo "Endlosschleife<br />";
19 }
20
21 $a = array("Hello ", "World", "!", "<br />");
22 foreach($a as $str){
23    echo $str;
24 }
```

1.4 Einführung in objektorientiertes Programmieren in PHP

1.4.1 Klassen

Wie zu Beginn erwähnt, ist PHP objektorientiert. In PHP werden Objekte ausschließlich mit der Methode *pass-by-reference* übergeben. Wird also ein Objekt an eine Funktion übergeben, wird keine Kopie erstellt, sondern über die Referenz direkt mit dem Original gearbeitet.

Eine Klasse wird mit dem Schlüsselwort `class` eingeleitet.

```
1 class Fahrzeug {
2    \\ Klassenrumpf
3 }
```

Ein neues Objekt wird, ähnlich wie in Java, mit `new` erzeugt:

```
1 $pkw = new Fahrzeug();
```

Unter der Objektvariable `$pkw` können Sie natürlich auch auf die Attribute und Methoden zugreifen:

```
1 echo $pkw->hoechstGeschwindigkeit;
```

Das geht natürlich nur, wenn Sie dieses Attribut gesetzt hatten. Es wird beim Attributsaufruf der Pfeil genutzt, da der Punkt in PHP der Konkatenationsoperator („Stringverkettung“) ist.

1.4.2 Konstruktoren und Destruktoren

Wenn Sie das Objekt einer Klasse erzeugen, wird ein Konstruktor aufgerufen. Seit PHP 5 trägt der Konstruktor immer den Namen:

`__construct()`

Die Methode `__construct` ist eine sogenannte *Magische Methode* wie z.B. auch der Destruktor `__destruct`. Es gibt weitere, die Sie im Internet finden können.

1.4.3 Kapselung

Seit PHP 5 gibt es zudem das Prinzip der Kapselung. Dabei unterscheidet PHP drei Schlüsselwörter:

- `private`: Eigenschaften und Methoden, die mit `private` gekennzeichnet wurden, sind nur innerhalb der Klasse und nicht von außerhalb zugänglich. Sollte ein Zugriff darauf versucht werden, tritt ein Fehler auf.
- `protected`: Hiermit können Eigenschaften und Methoden nur in der Klasse selbst und in davon abgeleiteten Klassen (wie im Abschnitt Vererbung noch erklärt) auf das Element zugreifen.
- `public`: Durch `public` ist ein öffentlicher Zugriff auf die damit bezeichneten Eigenschaften oder Methoden möglich.

Hinweis:

`public` ist der Standardwert, das heißt, dass alle Eigenschaften und Methoden ohne explizite Angabe der Kapselung automatisch `public` sind.

Übung 2:

Gegeben sei die Klasse Fahrzeug, die Sie in der Vorgabe finden. Füllen Sie die Lücken sinnvoll aus.

Anschließend erzeugen Sie einen LKW und einen PKW. Starten Sie die Motoren und erzeugen Sie ein „Rennen“, indem in einer Schleife die Fahrzeuge jeweils beschleunigt und die Geschwindigkeiten ausgegeben werden.

1.5 Kommunikation zwischen HTML und PHP

Ein zentrales Element von PHP ist die Auswertung von HTML-Formularen. In diesem Kapitel werden wir uns genauer damit beschäftigen.

Übung 3:

Erstellen Sie folgendes Eingabeformular und speichern Sie den HTML-Code unter *eingabe.html* in Ihrem Web-Verzeichnis:

```
1  <html>
2  <head>
3    <title>PHP-HTML-Übung</title>
4    <meta charset="UTF-8">
5  </head>
6  <body>
7  <p>Bitte tragen Sie Ihren Vornamen und Ihren Nachnamen ein.<br />
8  Senden Sie anschließend das Formular ab.</p>
9  <form action = "eingabe.php" method = "post">
10   <p><input name = "vor" /> Vorname</p>
11   <p><input name = "nach" /> Nachname</p>
12   <p><input type = "submit" />
13   <input type = "reset" /></p>
14 </form>
15 </body>
16 </html>
```

Im `<form>`-Element werden zunächst Informationen zum Formular angegeben: `action` ist das PHP-Skript, das beim Bestätigen des Formulars (Drücken des Buttons `submit`) aufgerufen wird und das Formblatt auswertet.

Für die `method` können entweder `post` oder `get` verwendet werden. Diese weisen auf die entsprechenden *Vordefinierten Variablen* oder im speziellen *Vordefinierten Assoziativen Arrays* hin, mittels derer die Information an das PHP-Skript übergeben werden sollen.

Hier ein kurzer Überblick über die Unterschiede:

1. Bei dem assoziativen Array `$_GET` werden die Formularinformationen mittels der URL in der Form *http://www.webseite.de/eingabe.php?feld1=wert1&feld2=wert2* weitergegeben.
2. Bei `$_POST` werden die Daten mittels der HTTP Methode *POST* übergeben.

Der HTML-Tag `<input>` dient zur Eingabe von Daten in ein Formular. Dabei ist der Wert von `name` der Schlüssel, um mittels `$_GET[name]` bzw. `$_POST[name]` dann den Inhalt des Eingabefeldes abzufragen. Für das vorgenannte Beispiel sieht die zugehörige `eingabe.php` dann folgendermaßen aus:

```
1 <html>
2 <body>
3 <?php
4   echo "Guten Tag, " . $_POST["vor"] . " " . $_POST["nach"] . "!";
5 ?>
6 </body>
7 </html>
```

Übung 4:

Schreiben Sie die PHP-Datei *eingabe.php* mit vorgenanntem Inhalt und testen Sie Ihre Anwendung.

Übung 5:

Erweitern Sie die Eingaben im Formular um die Eingabe der Adresse.

Übung 6:

Erstellen Sie ein neues HTML-Formular, in dem Sie zwei Zahlen eingeben. Mittels PHP soll die Summe zurückgegeben werden. Verwenden Sie hier testweise erst die *get-*, dann die *post-*Methode und schauen Sie sich die Unterschiede an.

Übung 7:

Erstellen Sie eine HTML-Form, in der Sie den User und das Passwort abfragen (unverschlüsselt). Bei richtigen Eingabe (Sie können die Werte mit festen Werten vergleichen) wird der User bestätigt.

Versuchen Sie, nur eine PHP-Datei für Formular und Auswertung zu verwenden. Das Formular soll nicht mehr angezeigt werden, wenn der Login-Versuch erfolgreich war.

Übung 8:

Erweitern Sie die letzte Übung um einen `<input type = ''hidden'' name = ... />`.
Wie wird dieses Element dargestellt? Wobei kann ein solches Element von Vorteil sein?