1 Konzept von PHP

Um die ersten Schritte bei serverseitiger Verarbeitung durchzuführen, nutzen wir PHP, da es die wahrscheinlich einfachste serverseitige Programmiersprache ist. Wir werden im Anschluss an PHP das Ganze auch im Express Server durchführen. PHP läuft im Rahmen einer Webserver Erweiterung. Für die meisten Webserver wie Apache, Nginx oder IIS gibt es solche Erweiterungen. Für diese Übung nutzen wir zum Testen unseren USBWebserver aus der 11. Klasse – also im Rahmen des Apache Webservers. Damit wir mit Express (der auf den port 8080 hört) nicht in Konflikt geraten, stellen wir den Port auf 8081 ein.

PHP wurde als einfach zu erlernende Sprache für die serverseitige Programmierung von HTML Homepagefunkionen geschaffen. Daher rührt auch das ursprüngliche Akronym „**P**ersonal **H**ome **P**age (Tools)“. Erst danach wurde es mit „**P**HP: **H**ypertext **P**reprocessor“ übersetzt. Die Grundidee ist, dass eine HTML Seite mit PHP-Inhalten „angereichert“ wird. Um dies zu ermöglichen, muss der Webserver die Datei zuerst einmal als verarbeitbare PHP-Datei erkennen, was einfach durch die Extension „*.php“ erreicht wird. Innerhalb dieser Datei verarbeitet der PHP-Prozessor ausschließlich den Text, der zwischen den Tags „`<?php`“ und „`?>`“ liegt. (Anm.: es ist auch möglich, sogenannte „short tags“ zu verwenden, was allerdings in der zentralen PHP-Konfigurationsdatei php.ini eingestellt werden muss.). Der folgende Dateiinhalt soll nun den Textstring "`Hallo, Welt!`" eine durch den PHP Befehl „`echo`“ im HTML Body Ausgeben:

```
<!DOCTYPE html>
<html lang="de">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>PHP und HTML</title>
 </head>
 <body>

             "Hallo, Welt!";

 </body>
</html>
```

Damit wir diesen Code ausprobieren können, müssen wir ihn im `root` Ordner des USBWebservers in einer PHP-Datei abspeichern – bspw. `myFirst.php`. Unter der Adresse http://localhost:8081/myFirst.php können wir die Seite nun aufrufen.

2 PHP in a Nutshell

PHP soll an dieser Stelle „nur“ zum Ausprobieren von dynamischen Webinhalten dienen – insofern verzichten wir auf einen vollumfänglichen PHP-Kurs[1]. Einige wenige „Kniffe“ sind jedoch notwendig, damit wir sinnvoll mit PHP arbeiten können (Details siehe PHP_Grundlagen_Schüler.docx). Beginnen wir mit Variablen:

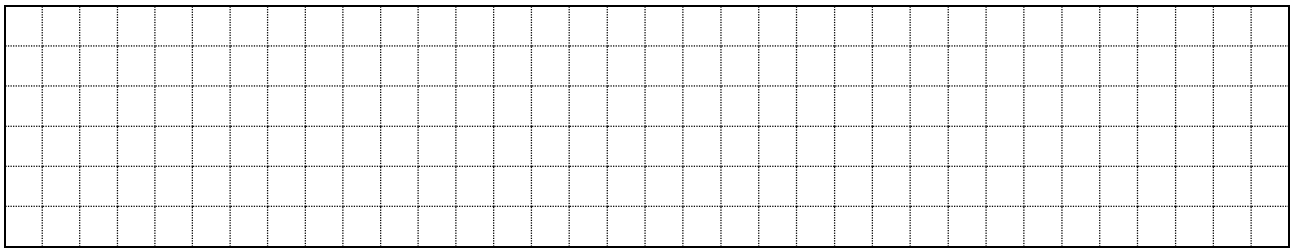
```
$vorName = "Peter"; // Deklaration und Zuweisung einer Variablen
echo "<br/>vorname: $vorName";

$anzeige = &$vorName; // Setzen der Adresse der Variablen auf $anzeige
echo "<br/>anzeige: $anzeige";

$vorName = "Paul"; // Überschreiben der Variable $vorName
echo "<br/>vorname: $vorName";
echo "<br/>anzeige: $anzeige";
```

[1] API: https://www.php.net/manual/de/ bzw. Kurs: https://www.w3schools.com/php/

PHP ist wie JavaScript dynamisch typisiert. Variablen werden mit einem $ Zeichen markiert und deren Adresse kann mit & abgerufen werden. PHP ist in der Lage, Variablen in Strings zu erkennen und mit dem Inhalt zu ersetzen. Stringverkettungen werden mit einem Punkt (nicht mit + wie in JavaScript) durchgeführt. Also folgende beiden Zeilen würden das gleiche Ergebnis liefern:

```
echo "<br/>vorname: $vorName";
echo "<br/>vorname: " .$vorName;
```

Kontrollstrukturen sind vergleichbar mit Java/JavaScript. Unterprogramme werden mit dem Schlüsselwort „`function`“ eingeleitet. Realisieren Sie nun eine Funktion, welche als Parameter `$vorName` und `$nachName` aufweist und bei Aufruf `buildNameSentence("Peter", "Lustig")` den String „`Mein Name ist Peter Lustig`“ zurückgibt:



Weiterhin erlaubt PHP, Code in mehrere Dateien aufzuteilen. Dadurch können wir eigene Bibliotheken schreiben oder auch existierende Bibliotheken in unser Projekt einbinden. Für den Anfang ist „`require_once`“ der einfachste Weg diese dann einzubinden. Hierzu verschieben wir die Funktion buildNameSentence in ein neues PHP File „`mySentenceBuilder.php`“ und erzeugen ein File `mySentenceUser.php`:

```
<!DOCTYPE html>
<html lang="de">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>PHP und HTML</title>
 </head>
 <body>

 <?php
  require_once("./mySentenceBuilder.php");
  echo buildNameSentence("Peter", "Lustig");
 ?>

 </body>
</html>
```

Mit dem Aufruf http://localhost:8081/mySentenceUser.php sehen wir die entsprechende Ausgabe.

3 Datenbankverbindung

Es existieren verschiedene Optionen, Datenbankverbindungen mit PHP aufzubauen. Passwörter werden hierbei in produktiven Umgebungen außerhalb des root (bzw. www oder http) – Ordner ausgelagert. Wichtig ist dann auch, dass wenn wir mit einem Versionsmanagement-Tool wie bspw. GIT arbeiten, wir diese Files entweder anonymisieren oder komplett aus dem Versionsmanagement ausklammern (bspw. mit `.gitignore`).

Wir nutzen in unseren Übungen die am häufigsten gebrauchte Form über PDO (**P**HP **D**ata **O**bjects), welche auf dem USBWebserver standardmäßig existieren. Hierbei verwenden wir „prepared Statements“, welche im ersten Schritt eine parametrierbare SQL Abfrage vorbereiten und anschließend die Parameter mit Werten besetzen.

Folgender Code erzeugt eine Datenbankverbindung, lädt die Daten in ein Array und gibt sie aus:

```
<!DOCTYPE html>
…
 <?php
  $dbh = new PDO('mysql:host=localhost:3306;dbname=warpshop',
                 'root', 'usbw');
  $stmt = $dbh->prepare("SELECT Name, ModelNumber, gewicht, groesse
          FROM Product WHERE gewicht >= ? ORDER BY gewicht LIMIT 10");
  if ($stmt->execute(array(100))) {
    while ($row = $stmt->fetch()) {
      echo $row["Name"] ."|" .$row["ModelNumber"]."|" .$row["gewicht"]
      ."|" .$row["groesse"] ."<br/>";
    }
  }
  $stmt = null;
  $dbh = null;
 ?>
 </body>
</html>
```

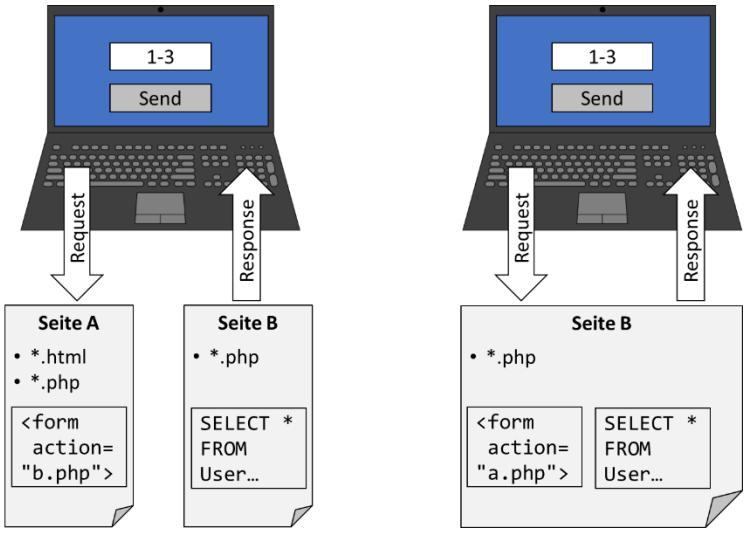
Die eigentliche Abfrage ist die folgende:

```
SELECT Name, ModelNumber, gewicht, groesse
FROM Product
WHERE gewicht >= ?
ORDER BY gewicht
LIMIT 10
```

Das ? in der Abfrage ist ein Platzhalter im Parameter (welcher nur für die „WHERE – Klausel“ gilt). Dieser wird beim Aufruf mit „execute“ durch die Inhalte des Arrays (in unserem Fall nur die „100“) ersetzt. Dabei gilt, dass die Fragezeichen der Reihe nach mit den Arraypositionen ersetzt werden – weshalb die Anzahl der Fragezeichen natürlich gleich der Arraylänge sein muss.

4 Forms

In unserer Query ist das Minimalgewicht ein Parameter. Diesen können wir nun auch vom Web-User einfordern, indem wir ihm ein Eingabefeld vorgeben. Eine Möglichkeit hierfür bieten uns HTML-Forms. Dies sind zusammenhängende Elemente, welche meist über POST an den Server geschickt werden können. Hierbei gibt es nun zwei Möglichkeiten. Entweder die Form wird über eine Seite A bereitgestellt und von Seite B verarbeitet, oder die Bereitstellung und Verarbeitung geschieht von ein und derselben Seite:



Wir sehen uns hier beide Möglichkeiten an und beginnen mit der Seite A als HTML File:

```
<!DOCTYPE html>
<html lang="de">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>PHP und HTML</title>
 </head>
 <body>
  <form action="SeiteB.php" method="post">
  Minimalgewicht: <input type="text" name="minweight">
  <input type="submit" value="send">
  </form>
 </body>
</html>
```

Die „Form“ benötigt die Sendemethode (bei uns „POST“) und das serverseitige Skript, welches die Verarbeitung übernimmt. Dieses sieht dann wie folgt aus:

```
<?php
function readDb($postData) {
  if (isset($postData["minweight"])) {
    $dbh = new PDO('mysql:host=localhost:3306;dbname=warpshop',
                'root', 'usbw');
    $stmt = $dbh->prepare("SELECT Name, ModelNumber, gewicht, groesse
            FROM Product WHERE gewicht >= ? ORDER BY gewicht LIMIT 10");
    if ($stmt->execute(array($postData["minweight"]))) {
      while ($row = $stmt->fetch()) {
        echo $row["Name"] ."|" .$row["ModelNumber"]."|" .$row["gewicht"]
        ."|" .$row["groesse"] ."<br/>";
      }
    }
    $stmt = null;
    $dbh = null;
  }
} ?>

<!DOCTYPE html>
<html lang="de">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>PHP und HTML</title>
 </head>
 <body>
     <form action="SeiteB.php" method="post">
     Minimalgewicht: <input type="text" name="minweight">
     <input type="submit" value="send">
     </form>
     <br/>
    <?php readDb($_POST); ?>
 </body>
</html>
```

Hier finden wir die (zugegebenermaßen schlecht formatierte) Ausgabe als PHP echo und im unteren Bereich den HTML Code. Dieser ist bis auf den PHP Aufruf 1:1 wie in der HTML Seite. Somit können wir die Eingabe in SeiteA.html und in SeiteB.php machen. Die Form-Daten finden wir in PHP im $_POST – Array unter dem jeweiligen Namen des HTML Feldes.