TU Clausthal

Werkzeuge der Informatik

Übung 2

Entwicklung Dynamischer Webanwendungen*

Wintersemester 2009/ 2010
Institut für Informatik

*) http://php.net/
http://de.selfhtml.org/

Übersicht: Dynamische Webentwicklung

- **0: Aufgabenstellung 1**
- 1: Einführung in PHP
- 2: Variablen
- 3: Operatoren
- 4: Kontrollstrukturen
- 5: Formulare
- 6: Arrays
- 7: Datenbankanbindung von PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

0.1 Aufgabenstellung 1: PHP-Skript

- In dieser Aufgabe erstellen Sie, anhand eines vorgegebenen HTML-Formulars zur Beantragung einer Mitgliedschaft in einer Studentenvereinigung, ein Serverskript (PHP-Skript).
- Das Serverskript (register.php) erwartet genau die im HTML-Formular spezifizierten Attributnamen, wertet diese aus und erstellt eine Bestätigungsmeldung.

- **Arbeiten Sie zunächst dieses Skript durch, um die grundlegenden Prinzipien von PHP zu erlernen. Mit Hilfe dieses Skripts erhalten Sie einen Überblick über alle notwendigen PHP-Schlüsselwörter, die zur Bearbeitung der Aufgabenzettel notwendig sind!**

0.2 Aufgabenstellung 1: Details

Die zugehörige Formularoberfläche und dessen HTML-Code


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
          <title>Mitgliedsaufnahme</title>
    </head>
    <body>
          <h2>Mitgliedsaufnahme</h2>
          <form method="post" action="https://www2.in.tu-
clausthal.de/~username/register.php"">
              Name: <br />
              <input type="text" name="name" size="50" />
              <p>
            Adresse: <br />
            <textarea name="adresse" rows="5" cols=40"></textarea>
            <p>
            Alter: <br />
            <input type="text" name="alter" size="3" />
            <p>
            Fachbereich: <br />
            <input type="text" name="fachbereich" size="50" />
            <p>
            Universität : <br />
            <input type="radio" name="uni" value="yes"
checked="true" /> TU Clausthal
            <input type="radio" name="uni" value="no" /> Andere
Universitäten
            <p>
            <input type="submit" name="submit" value="Überprüfen" />
         </form>
</body>
</html>
```

0.3 Aufgabenstellung 1: Details (2)

- Erstellen Sie die HTML-Datei register.html mit dem oben gegebenen HTML-Code

- Ändern Sie in „register.html“ unter dem Attribut action="`https://www2.in.tu-clausthal.de/~username/register.php`" den username in ihren Benutzernamen.

- Erstellen Sie in Ihrem Home-Verzeichnis einen Ordner „WWW“, dort werden Sie ihr PHP-Skript register.php ablegen

- Das PHP-Skript ruft zuerst die Informationen aus dem POST-Attibut mit den Variablen:
- „name “
- „adresse“
- „alter“
- „fachbereich“ und
- „uni“ ab.

- Dann wird überprüft, ob alle Felder aus dem Formular ausgefüllt wurden; falls nicht, sollte einen Nachricht „Bitte Feld <feldname> ausfüllen“ ausgegeben werden.
- Hinweis 1: Nutzen Sie eine if-Anweisung.
- Hinweis 2: Die „empty()“ Funktion überprüft, ob eine Variable leer ist und generieren in dem Fall eine Fehlermeldung und beendet das Skript.
- Hinweis 3: Die „die()“ Funktion bietet eine bequeme Möglichkeit, die Verarbeitung des Skriptes beim Auftritt eines Fehlers abzubrechen und eine Nachricht auszugeben.

0.3 Aufgabenstellung 1: Details (3)

- Es wird für das Attribut Alter noch zusätzlich überprüft ob, der Student zwischen 18 und 60 Jahre alt. Falls nicht, sollte die Meldung „Mitgliedschaft ist nur für Studenten zwischen 18 und 60 Jahren.“ angezeigt werden. Hinweis: Nutzen Sie den logischen Operator „||“

- Die Mitgliedschaft ist auf Studenten der „uni“ TU-Clausthal beschränkt. Eine entsprechende Meldung: „Mitgliedschaft ist nur für Studenten der TU-Clausthal “ soll dementsprechend angezeigt werden.

- Nach einer erfolgreichen Registrierung soll der Text „Vielen Dank für ihren Antrag. Ihre Mitgliedsbescheinigung wird Ihnen per Post zugesendet!“ ausgegeben werden.

Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1
- **1: Einführung in PHP**
- 2: Variablen
- 3: Operatoren
- 4: Kontrollstrukturen
- 5: Formulare
- 6: Arrays
- 7: Datenbankanbindung von PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

1.1 PHP Grundlagen

- PHP (PHP Hypertext Preprocessor) ist eine Skriptsprache, mit einer an C angelehnten Syntax, die hauptsächlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird. Der Name deutet an, dass etwas vor einem Zeitpunkt verarbeitet wird. In diesem Fall werden die PHP-Befehle in einer Datei verarbeitet, bevor der Server die Datei (bzw. dessen Ausgabe) sendet.

- Die aktuelle Version von PHP ist PHP 5.3 und geht in erster ihre Linie auf den Entwickler Rasmus Lerdorf im Jahr 1995 zurück.

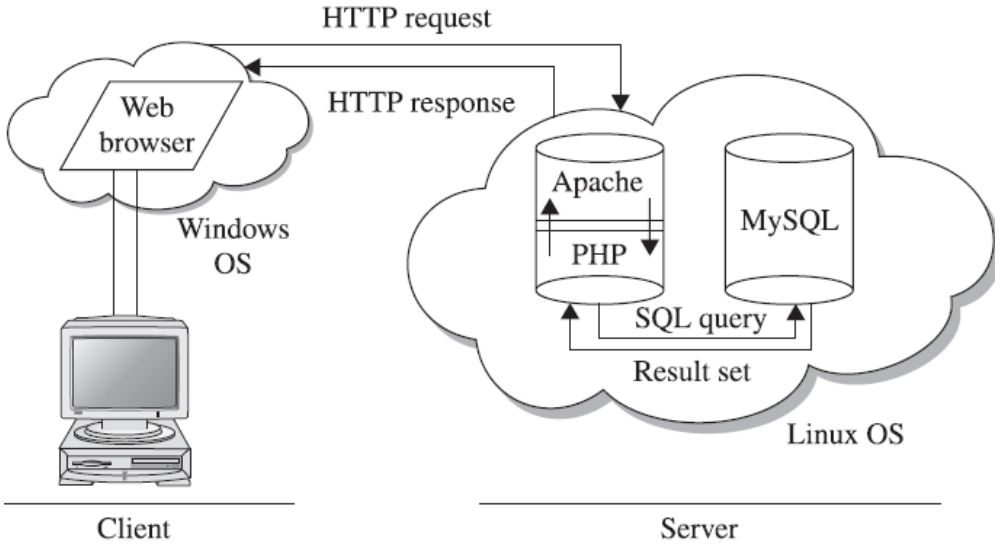
- Eine typische Vorgehensweise bei einer PHP-Anwendung für das Web ist die Einbettung von PHP-Code in eine oder mehrere Standard-HTML-Dokumente mit speziellem „Ausdruck" oder Trennzeichen. Wenn der PHP-Interpreter ein Skript liest, führt er nur den PHP-Code aus; alles was außerhalb von diesem Ausdruck ist wird ignoriert und wieder "wie es ist" zurückgesendet.

- Ein wichtiger Vorteil ist, dass PHP ein Modul bzw. ein Programm ist, welches auf einem Webserver installiert ist. So wird PHP-Code auf dem Server ausgeführt und nicht im Client-Browser.

1.2 Grundlegende Entwicklungskonzepte

- PHP Anwendungen müssen mindestens drei Komponenten enthalten:
  - Ein Basis Betriebssystem (OS) mit Server-Umgebung
  - Einen Web-Server zum "Abfangen" von HTTP Anfragen, um sie entweder direkt zu bearbeiten oder an den PHP-Interpreter weiterzuleiten
  - Ein PHP-Interpreter, um den PHP-Code zu analysieren und auszuführen, der die Ergebnisse an den Web-Server zurückschickt.

- Außerdem gibt es oft eine vierte, optionale, aber sehr nützliche Komponente:
  - Eine Datenbank (z.B MySQL, SQLite), die die Verbindung aus der PHP Schicht aufnimmt und die Daten aus einer Datenbank holt oder ändert.



1.3 Getting Started

- Um PHP zu testen kann eine PHP-Datei mit folgendem Inhalt angelegt werden.

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

- Dieser Inhalt wird in einer Datei namens phpinfo.php gespeichert und hochgeladen. Wenn dieses Skript nun aufgerufen wird, gibt es zwei mögliche Ausgaben, die erzeugt werden:

- Es wird eine leere Seite dargestellt, dies ist der Fehlerfall. Hier muss folgendes Überprüft werden:
  - Die Datei besitzt eine Dateiendung .php damit der PHP-Interpreter sie als PHP-Datei erkennt.
  - Sie enthält gültigen PHP-Code.
  - Auf dem Webserver ist PHP installiert und lauffähig.
  - Die Datei wird über eine http://-URL aufgerufen

- Im regulären Fall werden Informationen über PHP (und dessen Konfiguration) anzeigt. Mit Hilfe des phpinfo.php Skripts ist es also möglich, die Konfiguration von PHP zu betrachten.

1.4 Ihr erste PHP-Script

- Hauptziel eines PHP-Skriptes ist es, unabhängig von der dahinter steckenden Logik, eine Ausgabe zu erzeugen. Um in einem PHP-Skript Texte auszugeben wird üblicherweise das Sprachkonstrukt „echo“ verwendet.

```
<?php
  echo "Hallo Welt oder \"Hello World\"";
?>
```

- Die Anweisungen des Beispiels besteht aus einem Aufruf, der PHP echo-Anweisung, die für die Anzeige der Ausgabe am Benutzerfenster zuständig ist. Die Ausgabe ist in Anführungszeichen zu setzen.
- PHP-Code muss in die Tags `<?php ... ?>` eingeschlossen werden.
- Jede PHP-Anweisung muss mit einem Semikolon enden.
- Leerzeilen innerhalb des PHP-Ausdrucks werden ignoriert.
- Die Ausgabe von Sonderzeichen wie Anführungszeichen sind durch (\) Symbol davor möglich.

1.5 Kommentare

- In PHP werden Texte, bzw. Textstellen, die der PHP-Interpreter ignoriert, als Kommentare bezeichnet.
- Es wird allgemein zwischen zwei Arten von Kommentartypen unterschieden: *Einzeilige Kommentare* und *mehrzeilige Kommentare*.
  - Bei *einzeiligen Kommentaren* handelt es sich um Kommentare, die bis zum Zeilenende gelten. Solch ein einzeiliger Kommentar wird mit der Zeichenkette // dargestellt. Der folgende Text wird dann als Kommentar behandelt und somit vom PHP-Interpreter ignoriert:

```
<?php
    // gib einen Text aus
    echo 'Text';
 ?>
```

  - M*ehrzeilige Kommentare* können sich über mehrere Zeilen erstrecken. Statt also in jeder Zeile am Anfang ein // zu schreiben, wird ein Start- und Ende-Zeichen für einen mehrzeiligen Kommentar geschrieben. Ein mehrzeiliger Kommentar wird dabei mit /* gestartet und endet beim nächsten */:

```
<?php
     /*  Irgendein Kommentar der
         sich über mehrere Zeilen erstreckt bis zum
         entsprechenden stopzeichen
      */
   echo "Wieder was ausgeben";
?>
```

Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1
- 1: Einführung in PHP
- **2: Variablen**
- 3: Operatoren
- 4: Kontrollstrukturen
- 5: Formulare
- 6: Arrays
- 7: Datenbank mit PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

2.1 Variablen (1)

- Eine Variable ist nur ein Behälter, der sowohl mit numerischen als auch nicht-numerischen Daten gefüllt ist und an verschieden Stellen im Skript verwendet werden kann.
  - Damit Variablen in PHP als solche erkannt werden, müssen sie besonders gekennzeichnet werden, daher beginnen alle Variablen in PHP mit dem Dollarzeichen $. Danach muss sie mit einem Buchstaben oder Unterstrich _ beginnen. Nach dem ersten Zeichen können auch Ziffern im Variablennamen stehen .

```
<?php
  $gueltig
  $10_nicht_gueltig
?>
```

- Um eine Variable mit einem Wert zu füllen wird der Zuweisungsoperator „=“ verwendet. Dabei muss die Variable *links* vom Gleichheitszeichen stehen, der Wert der gespeichert werden soll *rechts*

```
<?php
   $email = ‘test@example.com';
   echo 'Meine Email-Adresse ist: ';
   echo $email;
?>
```

2.2 Variablen (2)

- Im Beispiel **(1)** wird der Variable $now der Wert „2009“ zugewiesen. Variablen können auch der Wert einer anderen Variablen oder das Ergebnis einer Berechnung zugewiesen werden.

(1)

```
<?php
// Werte zu Variable zuweisen
$now= ‘2009';

// Werte von einer Variable zu einer anderen Variablen zuweisen
$currentYear = $now;

// Berechnung durchführen
$lastYear = $currentYear - 1;

// output: '2008  ist Vorbei. Willkommen in 2009!'
echo "$lastYear  ist vorbei. Willkommen in $currentYear!";
?>
```

(2)

```
<?php
    // Werte zu Variable zuweisen
$car = 'Porsche';
 echo "Before unset(), my car is a $car"; // Ausgabe: 'Before unset(), my car is a Porsche
    // Variable löschen
unset($car);

// Oder

$car = null;
```

- Um eine Variable zu löschen, übergeben Sie sie der PHP unset()-Funktion. Alternativ kann man den Inhalt einer Variable durch den speziellen PHP Wert NULL leeren **(2)**

2.3 Konstanten

- *Konstanten* können, wie Variablen, Werte enthalten, die aber nicht mehr geändert werden
  - Konstanten können nur Skalare Werte, sowie den speziellen Datentyp NULL, enthalten. Im Gegensatz zu Variablen sind Konstanten jedoch im ganzen Skript verfügbar.
  - Konstanten werden mit Hilfe der PHP define()-Funktion definiert, die zwei Argumente akzeptiert: Der erste Parameter ist dabei der Name der neuen Konstante, der zweite Parameter der Wert für diese Konstante.
  - Konstantennamen unterliegen den gleichen Regeln wie Variablennamen, mit der Ausnahme, dass der „$“ Prefix nicht erforderlich ist. Konstanten werden üblicherweise komplett in Großbuchstaben geschrieben.

```
<?php
    define('SITE_NAME', 'Meine Tolle Homepage');

     echo SITE_NAME;

   // jedoch nicht
    echo ‘SITE_NAME!'; // wird die Konstante nicht finden
?>
```

2.3 Datentypen (1)

- PHP-Variablen können unterschiedliche Datentypen, von einfachen String- und numerischen Typen bis hin zu komplexeren Arrays und Objekten, zugewiesen werden. Die grundlegenden vier Datentypen sind:
  - Boolesche Werte: Der einfachste aller PHP-Datentypen, er bestehen aus einem einzigen Wert, der entweder 1 (true) oder 0 (false) annehmen kann.
  - PHP unterstützt auch zwei numerische Datentypen: Integer-Zahlen und Float-Zahlen. Beide können kleiner, größer oder gleich Null sein.
  - Für nicht-numerische Daten bietet PHP den String-Datentyp, der Buchstaben, Zahlen und Sonderzeichen enthalten kann. String-Werte müssen entweder in einfache oder doppelte Anführungszeichen eingeschlossen werden.
  - Der NULL Datentyp wird verwendet, um "leere" Variablen in PHP darzustellen, eine Variable des Typs NULL ist eine Variable ohne Daten.

```
<?php
    $validUser = true;      // Boolean
    $size = 15;             // integer
    $temp = 98.6;           // float-Zahl
    $cat = 'Tiger';         // string
    $hier = null;           // null
?>
```

2.4 Datentypen (2)

- Im Gegensatz zu anderen Programmiersprachen, wo der Datentyp einer Variable explizit definiert werden muss, bestimmt PHP abhängig vom Inhalt diesen automatisch. Wenn die Variable inhaltliche Änderungen im Laufe eines Skripts erfährt, übersetzt PHP die Variable auf den entsprechenden neuen Datentyp.

```
<?php
      // definieren string variable
         $whoami = 'Sarah';

      // Ausgabe: 'string'
         echo gettype($whoami);

     // zuweisen neuen  Wert in Variable
        $whoami = 99.8;

    //  Ausgabe: 'double'
        echo gettype($whoami);
?>
```

- Die PHP-Methode gettype() ermöglicht den Typ einer bestimmten Variable herauszufinden. Der Output zeigt, dass die Variable $whoami als ein String mit dem Wert ‘Sarah‘ definiert ist . Das zuweisen der Zahl 99,8 konvertiert die Variable automatisch in den Datentyp float.

- Dies bedeutet nicht, dass Sie PHP völlig ausgeliefert sind, man kann auch explizit den Typ einer PHP-Variable festlegen.

Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1
- 1: Einführung in PHP
- 2: Variablen
- **3: Operatoren**
- 4: Kontrollstrukturen
- 5: Formulare
- 6: Arrays
- 7: Datenbank mit PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

3.1 Operatoren (1)

- **Rechneroperatoren**

  Um einfache Rechenoperationen auszuführen gibt es in PHP sechs Rechenoperatoren. Das sind einmal die vier Grundrechenarten +, -, *, / und dann noch der (-)-Operator der eine Zahl negiert und der (%)-Operator der den ganzzahligen Rest einer Division bestimmt.

- **Verkettungsoperatoren**

  Ein Operator von PHP ist der Verkettungsoperator, der als einzelner Punkt geschrieben wird „(.)“. Dieser wird verwendet um zwei Ausdrücke miteinander zu verbinden, dabei ist es egal ob es sich um Strings oder Variablen handelt, die man verketten möchte. Auch Zahlen können so mit Strings verkettet werden.

(1)

```
<?php
    echo   'Max'.'Mustermann'; // erzeugt den String "MaxMustermann„
 ?>
```

- Die Anzahl der Verkettungen ist unbegrenzt. PHP verkettet intern jeweils zwei Teile miteinander und nutzt das Ergebnis für die nächste Verkettung.

(2)

```
<?php
   $vorname = 'Max';
   $nachname = 'Mustermamm';

   //vorname + ein leerzeichen + nachname
   $name = $vorname.' '.$nachname;
    echo $name;
?>
```

3.2 Operatoren (2)

- **Vergleichsoperatoren**

– Um zwei Werte auf Gleichheit zu testen, verwendet man in PHP den == Operator. Auf beiden Seiten des Operators werden die Inhalte notiert, die man vergleichen möchte. Diese können Variablen sein oder auch direkte Werte. PHP wertet dann einen solchen Ausdruck zu einem boolischen Wert aus, das Ergebnis ist somit entweder false oder true.

```
<?php
      "max" == "müller";            // ergibt bool(false), das Ergebnis wurde jedoch nicht verwendet/gespeichert
      $check = "max" == $var;   // prüft den Inhalt und speichert true oder false in $check
?>
```

– Für Zahlen existieren zusätzlich die Operatoren <, <=, > und >=.

```
<?php
    $var = 5 < 7;             // ist true
    $var = 10 <= 10;     // ist true
    $var = 9 > 9;          // ist falsch
?>
```

– Dann gibt es noch den Operator != (ungleich), um zu prüfen ob zwei Werte unterschiedlich sind.

```
<?php
    $var = 10 != 10;    // false
    $var = 0 != 1;       // true, da verschieden
?>
```

3.3 Operatoren (3)

- Der Operator === ermöglicht einen strengeren Vergleich zwischen Variablen. Er gibt nur true zurück, wenn die beiden Variablen oder Werte die verglichen werden, die gleichen Informationen enthalten **und** vom gleichen Datentyp sind. Somit sind die Werte "10" (als String) und 10 (als Integer) gleich wenn man == verwendet, jedoch unterschiedlich wenn man === verwendet.

```
<?php
    $var = 5 == "5";                    // ist true
    $var = 5 === "5";                   // ist false, da int != string

    $var = 'Max' == "Max";      // ist true, obwohl single quote string und double quote string
    $var = 'Max' === "Max„;  // auch true, beide haben die gleichen Stringinhalte und beide sind vom Typ String
?>
```

Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1
- 1: Einführung in PHP
- 2: Variablen
- 3: Operatoren
- **4: Kontrollstrukturen**
- 5: Formulare
- 6: Arrays
- 7: Datenbank mit PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

4.1 If-else-Anweisung

- Kontrollstrukturen in PHP werden dazu verwendet den Programmfluss zu steuern

- Die **if-Anweisung** steuert, ob ein Programmcode ausgeführt werden soll oder nicht. Der Aufbau ist dabei wie folgt.

```
<?php
    if (ausdruck) anweisung
?>
```

- Der Ausdruck wird von PHP verarbeitet und muss einen Wert zurückliefern. Dieser wird auf einen booleschen Wert abgebildet und true geprüft. Wenn dies der Fall ist, wird die folgende Anweisung ausgeführt.

- **Die if-else-Anweisung**: Wenn man Programmteile mit einer *if*-Abfrage steuert möchte man manchmal auch einen alternativen Programmteil aufrufen, falls die if-Abfrage nicht aufgerufen wurde. In PHP wird dies mit dem Schlüsselwort else realisiert, welches zu einem vorherigen if gehören muss.

```
<?php
if (login_gueltig) {
    // zeige adminbereich
} else {
    // zeige loginformular
}
?>
```

4.2 Sonderoperator

- **Sonderoperator:** Dieser Operator ist durch ein Fragezeichen (?) dargestellt und ermöglicht eine if-else Anweisung in einer einzigen kompakten Code-Zeile zu benutzen. Die zwei unten stehenden Beispielen sind Äquivalent

(1)

```
<?php
   if ($x < 10) {
echo 'X is less than 10';
   } else {
echo 'X is more than 10';
   }
?>
```

(2)

```
<?php
  echo ($x < 10) ? 'X is less than 10' : 'X is
more than 10';
?>
```

- **(2)** Hier wählt der Sonderoperator den Code auf der linken Seite, wenn die Bedingung true ergibt, und den Code auf der rechten Seite, wenn die Bedingung nicht erfüllt ist.

4.3 Schleifen

- Schleifen dienen dazu einen Programmteil öfters hintereinander auszuführen, ohne ihn mehrfach im Code zu schreiben. Es gibt 4 Arten von Schleifen in PHP: *while*, *do-while, for* und *foreach*

- Beispiel: die *while*-Schleife. Sie ist der einfachste Schleifentyp,
  - Die *while*-Schleife ist der einfachste Schleifentyp, er besitzt einen Schleifenkopf in dem nur ein Ausdruck steht. Dieser Ausdruck wird vor jedem Schleifendurchlauf ausgewertet und auf Boolean geprüft. Wenn der Ausdruck true ergibt, so wird der Schleifenrumpf ausgeführt. Danach beginnt die nächste Überprüfung des Schleifenkopfs. Wenn der Ausdruck false ergibt, so wird die Schleife beendet bzw. übersprungen und der weitere PHP-Code wird ausgeführt.
  - Da der Ausdruck ständig für einen neuen Schleifendurchlauf überprüft wird, muss der Ausdruck irgendwann den Wert false liefern, damit die Schleife abgebrochen wird.
  - Wenn dies nicht der Fall ist/wird so hat man eine Endlosschleife, das Skript terminiert nicht sauber und der Webserver bricht das Skript nach einer bestimmten Zeit von sich aus mit einer entsprechenden Fehlermeldung ab.
  - Eine While-Schleife wird in PHP mit dem Schlüsselwort while eingeleitet. Danach folgt in runden Klammern (()) der zu prüfende Ausdruck, gefolgt von einer einzelnen oder mehreren Anweisungen in geschweiften Klammern ({}).

```
<?php
while (isAutoDreckig()) {
    Reinigen();         }
?>
```

Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1: PHP Allgemein
- 1: Einführung in PHP
- 2: Variablen
- 3: Operatoren
- 4: Kontrollstrukturen
- **5: Formulare**
- 6: Arrays
- 7: Datenbank mit PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

5.1 Formular (1)

- Viele Internetseiten bieten Bereiche an, in denen ein Text eingegeben oder eine Auswahl getroffen werden kann. Danach muss man dann üblicherweise auf einen Absenden-Button klicken. Solche Stellen werden *Formulare* genannt. Dabei werden die Eingaben und Auswahlfelder an ein PHP-Skript gesendet. Dieses PHP-Skript kann dann darauf entsprechend reagieren.

- Formulare werden mit dem HTML-Element <form> erzeugt. Das action-Attribut gibt dabei die URL an, an die die Formulardaten gesendet werden sollen, in unserem Fall also stets ein PHP-Skript. Dieses Skript wird dann mit den Daten des Formulars aufgerufen, es wird also zu dieser Seite weitergeleitet.

- Das „method“ Attribut gibt an, auf welche Weise die Formulardaten gesendet werden. Mit dem Wert get wird die URL mit der *GET-Methode* aufgerufen. Dies entspricht einem normalen Aufruf wie die Eingabe einer URL im Browser.

- Mit der POST-Methode werden die Formulardaten versteckt in dem HTTP-Request übermittelt. Die Daten selbst bleiben zwar weiterhin lesbar, sind aber für den Benutzer nicht zu sehen

5.2 Formular (2)

- Um zu illustrieren, betrachten das folgende, einfache Web-Formular (choose.html).

- Das Formular dient zur Auswahl einer Automarke (selType) mit der gewünschten Farbe (txtColor). Als Eingabeform dient eine Auswahlliste und ein Eingabefeld. Das 'action' Attribut verweist auf ein PHP-Skript namens car.php.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head><title /></head>
      <body>
         <h2>Select Your Car</h2>
         <form method="post" action="car.php">
          Type: <br />
   <select name="selType">
      <option value="Porsche 911">Porsche 911</option>
      <option value="Volkswagen Beetle">Volkswagen Beetle</option>
      <option value="Ford Taurus">Ford Taurus</option>
   </select><p />
   Color: <br />
       <input type="text" name="txtColor" /> <p />
       <input type="submit" />
   </form>
  </body>
</html>
```

5.3 Formular (3)

- Betrachten wir das entsprechende PHP-Skript „car.php“

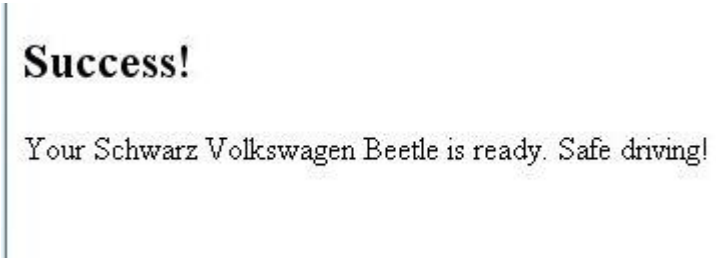
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head><title /></head>
    <body>
      <h2>Success!</h2>
        <?php

          // Formulareingabe erhalten
          $type = $_POST['selType'];
          $color = $_POST['txtColor'];

          // Verwendung Formulareingabe
         echo “Ihre  $color $type is  ready. Safe driving!";
       ?>
    </body>
</html>
```

- Die Daten aus dem Formular werden durch die POST-Methode an das PHP-Skript weitergereicht. Die eingegebenen Variablen und ihre Werte werden automatisch durch die Variable $_POST zur Verfügung stehen. Die echo-Anweisung gibt die Auswahl als Ergebnis zurück.

- Die Ausgabe sieht wie im Bild aus:



Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1: PHP Allgemein
- 1: Einführung in PHP
- 2: Variablen
- 3: Operatoren
- 4: Kontrollstrukturen
- 5: Formular
- **6: Arrays**
- 7: Datenbank mit PHP
  - Beispiel: SQLite
- 8: Aufgabenstellung 2

6.1 Arrays (1)

- Arrays sind in PHP ein wichtiger Bestandteil. Formulardaten und URL-Parameter sind z.B. in Arrays abgelegt. Werte aus einer Datenbank werden in einem Array gespeichert. Formal sind Arrays geordnete (nicht zu verwechseln mit sortiert) Paare von Schlüsseln und Werten. Ein Schlüssel darf dabei nur aus Integer-Zahlen oder Strings bestehen.

- Arrays werden in PHP mit dem Sprachkonstrukt „array“ erzeugt. Die Werte für das Array gibt man, mit Kommatas getrennt, als Parameter an.

```
<?php
    $arr = array("foo", "bar", "bla", 5.6, false, -10, "foo", "foo", "bar", "foo");
?>
```

- Dieses Array besitzt 10 Elemente. Die Schlüssel, oder auch Indizes werden automatisch bestimmt, beginnend bei 0 aufsteigend. Somit gehört zum ersten aufgelisteten Arrayelement der Index 0, das letzte aufgelistete Arrayelement besitzt den Index 9. Dies kann auch mit der var_dump- Funktion überprüft werden.

- Die Funktion var_dump() gibt strukturierte Informationen über einen oder mehrere Ausdrücke aus, darunter auch den entsprechenden Typ und Wert. Arrays und Objekte werden rekursiv durchlaufen und die jeweiligen Werte eingerückt dargestellt, um die Struktur zu verdeutlichen.

6.2 Arrays (2)

- Die Werte eines Arrayelements können beliebig verändert werden, indem der neue Wert wie bei einer Variablen zugewiesen wird.

```
<?php
$arr = array("eins", "zwei");
$arr[1] = "fünf";
?>
```

- Wenn es bereits ein Arrayelement mit dem angegebenen Index gibt, so wird der alte Wert mit dem neuen Wert überschrieben. Wenn es kein solches Arrayelement gibt, wird ein neues Arrayelement mit dem angegebenen Index und Wert an das Array angefügt. Damit ist es möglich zuerst ein leeres Array zu erzeugen und später zu füllen.

- Arrays, die Strings als Indizes enthalten, werden üblicherweise als *assoziative Arrays* bezeichnet. Im anderen Fall werden sie schlicht *Arrays* oder *nummerische Arrays* genannt.

```
<?php
$foo = array();
$foo[] = "wert";   // Index 0 wird verwendet
$foo[] = "wert";   // Index 1 wird verwendet, höchster bisheriger Index ist 0
$foo[10] = "wert"; // Index 10, wurde angegeben
$foo[] = "wert";   // Index 11, höchster bisheriger Index ist 10
var_dump($foo);

$foo = array();
$foo[-5] = "wert"; // Index -5, wurde angegeben
$foo[] = "wert";   // Index 0, höchster bisheriger Index ist -5, neuer Index ist jedoch mindestens 0
var_dump($foo);
?>
```

- Wenn bei einer Wertzuweisung der Index weggelassen wird (also nur $arr[]) so wird automatisch als Index der bisher größte Zahlenindex +1 verwendet, jedoch minimal der Wert 0, damit keine negativen Indizes erstellt werden.

6.3 Werte in Arrays zuweisen

- PHP-Regeln für die Benennung von Array-Variablen sind die gleichen wie für die regulären Variablen, Variablennamen müssen mit einem Dollarzeichen ($) beginnen

- Es gibt zwei Möglichkeiten der Zuweisung von Werten
  - Mittels der array()-Methode werden Werte, getrennt durch Kommas, auf einmal zugeordnet, diese Methode erstellt ein Standard (nummerisch indexiertes) Array (1)
  - Die zweite Möglichkeit ein solches Array zu erzeugen, ist die Werte einzeln mit Index festzulegen. (2)

- Die Indizes können auch innerhalb des array Konstrukts (assoziativen Arrays) angegeben werden. Statt den einfach den Wert zu notieren, fügt man vorher noch x => ein, wobei x der zu wählende Index ist. (3)

(1)

```
<?php
  // define array
   $cars = array(
                'Ferrari',
                 'Porsche',
                 'Jaguar',
                 'Lamborghini',
                 'Mercedes‘
                 );
?>
```

(2)

```
<?php
  // define array
    $cars[0] = 'Ferrari';
    $cars[1] = 'Porsche';
     $cars[2] = 'Jaguar';
     $cars[3] = 'Lamborghini';
     $cars[4] = 'Mercedes';
?>
```

(3)

```
<?php
$bar = array(5 => "bar", "foo");
// 2. Element bekommt den Index 6

$var = array(-10 => "abc", "xyz");
// 2. Element bekommt den Index 0, s.o.

$var = array("Name" => "Max Mustermann", "foobar");
// 2. Element bekommt den Index 0
?>
```

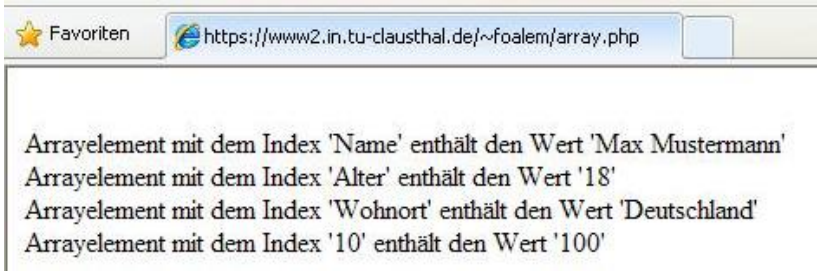
6.4 Foreach-Schleifen

- Für Arrays gibt es den speziellen Schleifentyp *foreach*. Mit diesem Schleifentyp werden die einzelnen Elemente eines Arrays durchlaufen. Eine Foreach-Schleife beginnt mit dem Schlüsselwort foreach, gefolgt von dem zu durchlaufenden Array, bzw. die Variable die das Array enthält, welches durchlaufen werden soll. Danach folgt das Schlüsselwort „as“ und eine neue Variable. In dieser Variable wird für jeden Schleifendurchlauf der neue Wert des nächsten Arrayelements gespeichert.

```
<?php
   $a = array("foo", "bar", "bla");
   foreach ($a as $value) {
      echo $value."\n";
                                  }
    // gibt nacheinander die Werte aus dem Array aus
?>
```

- Wenn man zusätzlich zu den Werten noch die entsprechenden Arrayindizes benötigt muss man vor der Schleifenvariable noch „=>“ schreiben.

```
<?php
$user = array('Name' => "Max Mustermann",
            'Alter' => 18,
            'Wohnort' => 'Deutschland',
            10 => 100);
foreach ($user as $k => $v) {
   echo "Arrayelement mit dem Index '".$k."' enthält den Wert '".$v."'\n";
}
?>
```



Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1
- 1: Einführung in PHP
- 2: Variablen
- 3: Operatoren
- 4: Kontrollstrukturen
- 5: Formular
- 6: Arrays
- **7: Datenbank mit PHP**
  - **Beispiel: SQLite**
- 8: Aufgabenstellung 2

7.1 Grundlagen der Datenbank (1)

- Was immer auch ein PHP-Skript verarbeitet, früher oder später müssen Daten für weitere Skriptaufrufe abgespeichert werden.
- Variableninhalte sterben mit dem Ende des Skriptaufrufs und verschwinden aus dem Speicher. Um Daten abzuspeichern können diese Beispielsweise in einer Datei abgelegt werden. Das PHP-Skript öffnet dabei eine Datei und schreibt die Daten in diese, dies kann jedoch aufwendig werden, da die Daten mehr oder weniger *einfach so* in der Datei liegen.
- Bei der Verwendung einer relationalen Datenbank wird ein anderer Weg eingeschlagen. Die Daten werden dabei nicht in Dateien abgelegt, sondern in einer Tabelle.
- Pro Eintrag für diese Tabelle wird eine Zeile angelegt, die auch Datensatz genannt wird.
- In einer relationalen Datenbank werden Daten in Tabellen in Form von Datensätzen abgespeichert. Damit eine solche Tabelle identifiziert werden kann, besitzen alle Tabellen einen Namen. Des Weiteren gehören zu einer Tabelle eine Anzahl von Spaltennamen und den dazugehörigen Spaltentypen.
- Um Tabellen zu erstellen wird der SQL-Befehl CREATE TABLE verwendet. Dabei wird zuerst ein Tabellenname angegeben. Danach folgt in Klammern eine Aufzählen von Spalten, die untereinander mit einem Kommata getrennt sind

7.2 Grundlagen der Datenbank (2)

- Nachdem die Tabelle mit den Spalten erstellt wurde, kann mit dieser gearbeitet werden. Um nun einen Datensatz in die Tabelle hinzuzufügen, wird der INSERT INTO Befehl verwendet. Nach den Schlüsselwörtern INSERT INTO gibt man die Tabelle an, zu der man einen neuen Datensatz hinzufügen will. Danach kommen in Klammern und untereinander, mit Kommatas getrennt, eine Liste von Spaltennamen zu denen man später die Werte angibt, die man in den neuen Datensatz speichern will. Mit REPLACE INTO werden die vorhandene Datensätze geändert, falls ein entsprechender Datensatz exisitert.

- Wenn nun die Tabellen mit Datensätzen gefüllt ist, wäre es auch wünschenswert, wenn diese Datensätze ausgelesen werden können. Dazu wird der SQL-Befehl *SELECT* verwendet. Der *SELECT*-Befehl hat eine sehr hohe Anzahl an Parametern. Unabhängig wie der SELECT-Befehl nun aussieht und was er ermittelt, ist das Ergebnis immer eine Tabelle.

- Da wir nun einzelne Daten auslesen können versuchen wir nun Daten aus Tabellen auszulesen. Dazu wird der SELECT-Befehl um ein FROM ...-Teil erweitert. Der ...-Teil gibt dabei die Tabelle an aus der wir die Daten auslesen wollen. Wenn wir auf diese Weise eine Tabelle auslesen, können wir im SELECT-Teil die Spaltennamen aus der Tabelle verwenden. Die Syntax ist also SELECT spalten FROM tabelle.

- Um wieder zurück auf PHP zu kommen: wir wollen natürlich mit PHP auf die Datenbank zugreifen. Es gibt eine Reihe von relationalen Datenbank (MySQL,…). Wir werden wir uns auf SQLite beschränken.

7.3 Beispiel: SQlite

- SQLite ist ein schnelles und effizientes Datenbank-System, vor allem für kleine bis mittlere Anwendungen. Im Gegensatz zu MySQL, die eine große Zahl von miteinander verbundenen Komponenten enthält, ist SQLite selbst vollständig in einer einzigen Library-Datei enthalten.

- SQLite unterstützt vier Spaltentypen.
  - INTEGER für Integer-Werte
  - REAL für Float-Werte
  - TEXT für String-Werte und
  - BLOB für große Datenmengen

- SQLite unterstützt alle Standard-SQL-Anweisungen: SELECT, INSERT, DELETE, UPDATE und CREATE TABLE.

7.4 Abrufen von Daten aus einer SQLite -Datenbank

- Bevor man auf die Datenbank zugreifen kann, muss diese geöffnet werden.
- Beispiel:

```
$sqlite = new SQLiteDatabase('Datenbankname.db') or die (could not open Database)
```

- **SQLite** öffnet die Datenbank, bzw. erzeugt sie, sollte sie nicht vorhanden sein. Falls die Anweisung scheitert, wird eine Fehlermeldung ausgegeben.

- Um aus einem Skript eine Datenbank aufzurufen, muss eine SQL Abfrage ausgeführt werden. Dies geschieht mit Hilfe der SQLite query() -Methode.

- Die sqlite_escape_string()-Funktion bereitet einen String für die Verwendung als SQL-Parameter auf.

- Sobald die gesamte Ergebnismenge verarbeitet worden ist, und keine weitere Datenbank-Operationen stattfinden, sollte die Datenbank geschlossen werden: „unset($sqlite)“ .

Übersicht: Dynamische Webentwicklung

- 0: Aufgabenstellung 1
- 1: Einführung in PHP
- 2: Variablen
- 3: Operatoren
- 4: Kontrollstrukturen
- 5: Formular
- 6: Arrays
- 7: Datenbank mit PHP
  - Beispiel: SQLite
- **8: Aufgabenstellung 2**

8 Aufgabenstellung 2

- In dieser Aufgaben erstellen Sie einen Skript „feedback.php“, in dem Sie eingegebene Werte aus einem Formular in einer Datenbank „formular.db“ speichern.
- Benutzen Sie das Formular (feedback.html) aus der 1. Übung.
- Ändern Sie das action Attribut in feedback.html durch „https://www2.in.tu-clausthal.de/~username/feedback.php“.
- Nachdem Sie die Eingabewerte gesendet haben, soll der Text: Eingabe in Datenbank erfolgreich hinzugefügt ausgegeben werden.

- Die gespeicherten Werte werden mithilfe eines anderen Skripts, „suche.php“, gesucht und angezeigt.
- Hinweis: Folgen Sie den Hinweise chronologisch.

8.1 Aufgabenstellung 2: Details (1)

1. Erstellen Sie ein PHP-Skript „feedback.php“ in ihrem WWW-Ordner.

2. Definieren Sie zunächst das CSS Format „message“:
   - Aussenabstand nach rechts und links: auto
   - Width: 40%
   - Textausrichtung: zentriert
   - Rahmen: „solid 2px green“

3. Erstellen Sie eine neue SQLite Datenbank mit dem Namen „formular.db“

```
$sqlite = new SQLiteDatabase('formular.db') or die ("Datenbank kann nicht geöffnet werden");
```

4. Erstellen Sie eine Tabelle „Daten“, in der die Datensätze gespeichert werden. Dabei sollen Vorname und Nachname als Schlüssel (verwendet für die Suche der gespeicherten Datensätzen) definiert werden.
   - Mit einem Query wird auf die Tabelle zugegriffen.
   - Mit einer if-Anweisung soll verhindert werden, dass die Tabelle „Daten“ bei jedem Aufruf erneut erstellt wird.

```
if ($sqlite->query("SELECT COUNT(*) FROM Daten") === false)
                        {
$sql = "CREATE TABLE Daten (Vorname TEXT KEY, Nachname TEXT KEY , Strasse TEXT, Postleitzahl INTEGER, Ort TEXT, EMail TEXT,
Nachricht TEXT, Auswahl TEXT)";

$sqlite->query($sql);
                        }
```

8.2 Aufgabenstellung 2: Details (2)

5. Jede POST Variable wird gefiltert, um SQL-Injections vorzubeugen.

```
foreach ($_POST as $key=>$value)
    {
        $_POST[$key] = sqlite_escape_string($value);
    }
```

6. Alle Eingabewerte werden als Wertkette in der Variable $values zusammengeführt.

```
$values = '"' . $_POST['vorname'] . '","' . $_POST['nachname'] . '","' . $_POST['strasse'] . '","' . $_POST['plz'] . '","' . $_POST['ort'] . '","' . $_POST['email'] . '","' . $_POST['nachricht'] . '","' . $_POST['auswahl'] . '"';
```

7. Eingabewerte in der Tabelle hinzufügen oder aktualisieren.

```
$sqlite->query("INSERT OR REPLACE INTO Daten (Vorname, Nachname, Strasse, Postleitzahl, Ort, EMail, Nachricht, Auswahl) VALUES (" . $values . ")");
```

8. Nachricht ausgeben

9. Schließen der Datenverbindung

```
unset($sqlite);
```

8.3 Aufgabenstellung 2: Suche

- Jetzt sollen die gespeicherten Werte gesucht und in einer Tabellen angezeigt werden.
- Erstellen Sie einen Skript „suche.php“ in ihrem WWW-Ordner.
- Es wird nur nach dem Vor- und Nachname gesucht. Ändern Sie ihr Feedback Formular dementsprechend und speichern Sie es in einer neuen Datei (suche.html).

- Der Skript kann in suche.html über „https://www2.in.tu-clausthal.de/~username/suche.php“ aufgerufen werden
- Prüfen Sie, ob alle Daten richtig angezeigt werden

8.4 Aufgabenstellung 2 : suche.php

```
<html><head> <style type="text/css">
TABLE {
border-collapse: collapse; }
TABLE TD {
border: 1px solid #808080; }
TABLE TH {
font-weight: bold;   border: 1px solid #808080;  background-color: #EEEEEE; text-align: left; }

</style></head><body><table>

<?php
   $sqlite = new SQLiteDatabase('formular.db') or die ("Could not open database");
    // Abfrage der Datenbank
   $daten = $sqlite->query("SELECT * FROM Daten WHERE Vorname = '" . $_POST['vorname'] . "' AND Nachname = '" . $_POST['nachname'] . "'");
               if ($daten !== false)
              {
                    // einen einzelnen Datensatz holen
                   if ($array = $daten->fetch(SQLITE_ASSOC))
                  {
// Pruefen, ob der Datensatz leer ist (fetch(SQLITE_ASSOC) gibt false bei leeren Datensaetzen zurueck)
if ($array !== false)
                        {
      // Falls nicht, Daten ausgeben (Spaltenname: Wert)
       foreach ($array as $key=>$value) {
       echo "<tr><th>" . $key . ": </th><td>" . $value . "</td></tr>";
                                                }
}
                          else {
                                // Dem Nutzer anzeigen, dass kein Datensatz gefunden wurde
       echo "Es wurde kein Datensatz gefunden, der dem Vor- und Nachnamen entspricht!";
                                }
                    }
               }
  // Datenbankverbindung schliessen
   unset($sqlite);

?>
</table></body></html>
```