PHP Grundlagen

ID:	**TT-UU5**
Dokumenten URL:	**http://docs.tx7.de/TT-UU5**
Autor:	**Tom Gries <tom@tx7.de>**
Version:	**5.0.1 vom 13.07.2017**

>> Themen

- Was ist PHP?
- Einbettung in (X)HTML Code
- Variablen und Datentypen
- Übergabe von Parametern
- Grundbefehle
- Referenzen

>> Was ist PHP?

PHP ist die Abkürzung für **PHP Hypertext Preprocessor** (ein sogenanntes rekursives Akronym). Dabei handelt es sich um eine Scriptsprache zur **Vorverarbeitung und Erzeugung von Web-Seiten**.

PHP wird hauptsächlich zur Erstellung von dynamischen Webseiten verwendet. Eine Stärke von PHP ist die breite Datenbankunterstützung.

>> Was ist PHP?

Im Unterschied zu statischen Web-Seiten kann sich der Inhalt einer dynamischen Web-Seite aufgrund von Aktionen des Betrachters oder aufgrund neuer Basis-Informationen wie z. B. aus Datenbanken ändern.

Damit PHP-Programme von einem Webserver ausgeführt werden können, müssen sie eine bestimmte Endung (Extension) haben. Hierfür hat sich **.php** durchgesetzt.

>> Einbettung in (X)HTML

Um festzustellen, ob PHP vom eingesetzten Webserver unterstützt wird, muss lediglich eine Datei mit dem Inhalt

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

im DocumentRoot erstellt werden (z. B. als phpinfo.php).

Daraufhin sollten Tabellen mit Konfigurationsinformationen angezeigt werden. Wird nur der Inhalt der soeben erstellten Datei ausgegeben, wird PHP nicht unterstützt bzw. muss noch aktiviert werden.

>> Einbettung in (X)HTML

Anmerkung zu den Beispielen in diesem Dokument:

Bei den Codes handelt es sich meistens nur um Auszüge. Aus Platzgründen wird hier oft auf DOCTYPE und andere Auszeichnungen verzichtet. Zu einem kompletten (X)HTML Dokument gehören sie aber dazu.

(X)HTML Dokumente sollten nach der Fertigstellung immer mit einem W3C kompatiblen Validator geprüft werden, z. B. unter http://docs.tx7.de/TT-W3C.

>> Einbettung in (X)HTML

Das folgende Beispiele zeigt, wie einfach PHP in (X)HTML eingebunden werden kann.

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <title>Hallo Welt</title>
  </head>

  <body>
    <?php echo "Hallo Welt"; ?>
  </body>
</html>
```

>> Einbettung in (X)HTML

Eine PHP Datei könnte aber auch wie folgt aussehen:

Als **script-01.html** in public_html speichern

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <title>Hi all</title>
  </head>

  <body>
    <?php
        echo "<p>Der PHP Code</p>\n";
    ?>
  </body>
</html>
```

Aufgabe 1:

Das PHP Script von der vorherigen Seite erstellen, im eigenen DocumentRoot unter dem Namen **script-01.html** speichern und mit einem beliebigen Browser aufrufen.

Danach die Datei in **script-01.php** umbenennen und nochmals aufrufen.

Was ist der Unterschied?

>> Einbettung in (X)HTML

Einbettungsmöglichkeiten:

Es gibt je nach Konfiguration bis zu vier Möglichkeiten, PHP in (X)HTML Dokumente einzubetten:

```
1. <?php [PHP-Anweisungen] ?>
2. <? [PHP-Anweisungen] ?>
3. <% [PHP-Anweisungen] %>
4. <script language="php"> [PHP-Anweisungen] </script>
```

Die erste ist XML konform. Nur diese sollte verwendet werden. Die zweite und Dritte können zu etlichen Problemen führen und sollten **unbedingt** vermieden werden. Die vierte ist lediglich zu lang.

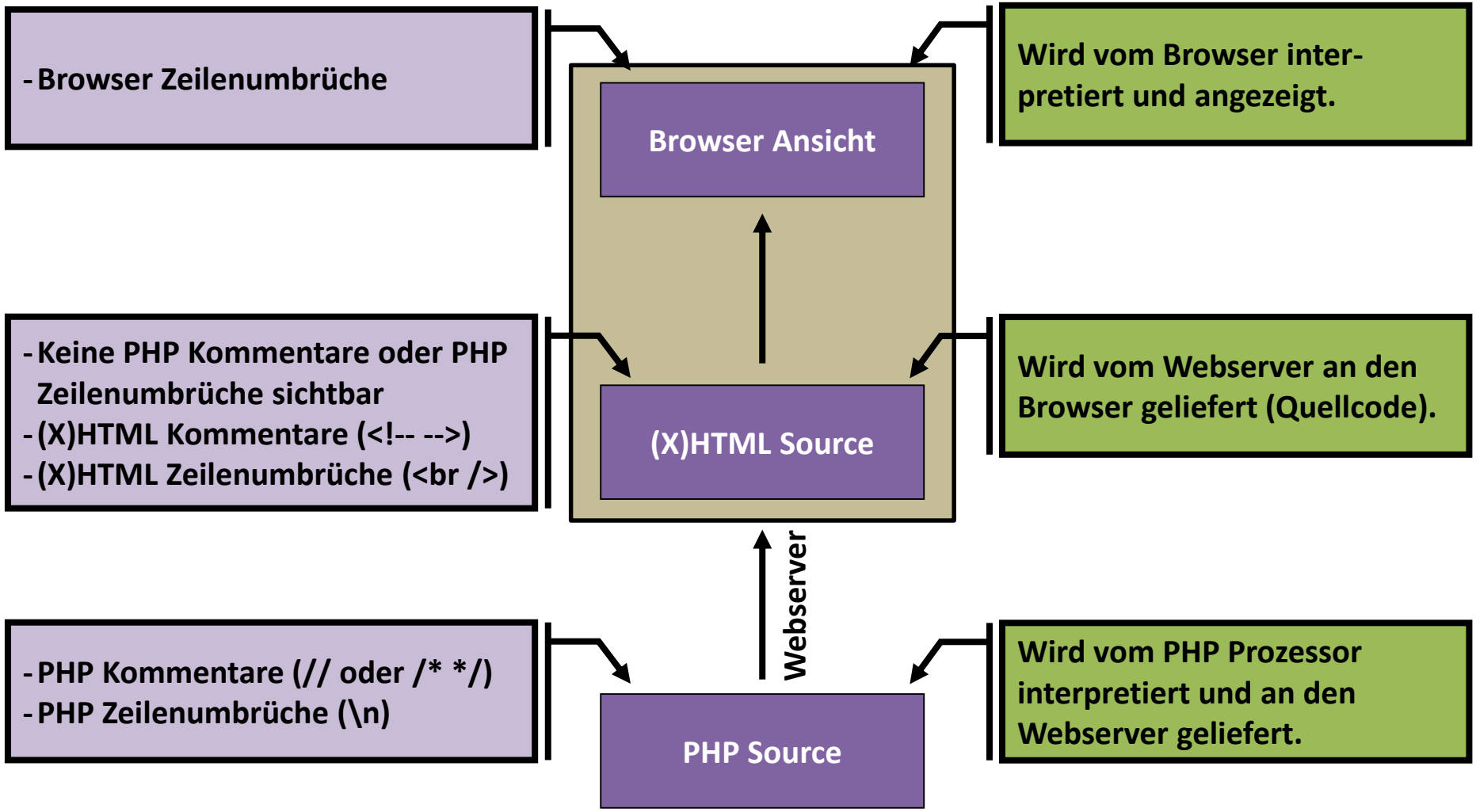
>> Kommentare und Zeilenumbrüche

Kommentare:

Mit Hilfe von Kommentaren wird ein Programm lesbarer. Kommentare werden nicht ausgeführt, sondern dienen nur zur Information für den oder die Entwickler.

```
<?php
  echo "Das ist der Anfang"; // Kommentar
                             // bis Zeilenende
  /* Ein Kommentar über
  mehrere Zeilen */

  echo " und hier das Ende des Programmes";
?>
```

>> Kommentare und Zeilenumbrüche



Variablen:

Eine Variable ist ein Platzhalter, der verschiedene Werte annehmen kann. Sie ist veränderlich, also **variabel**.

Das Gegenteil einer Variablen ist die Konstante, also ein fester, unveränderlicher Wert. Der Mehrwertsteuersatz (zum Beispiel 19%) ist beispielsweise eine Konstante.

Variablen (cont.):

Für die Namen von Variablen gelten einige Regeln:

- Sie müssen mit einem Dollar-Zeichen beginnen.
- Sie dürfen nur aus Buchstaben und Ziffern bestehen.
- Das erste Zeichen sollte ein Buchstabe sein.
- Sie dürfen keine Leerzeichen enthalten und nur den Unterstrich »_« als Sonderzeichen enthalten.
- Groß- und Kleinschreibung wird unterschieden.
- Sie dürfen nicht mit einem reservierten Wort identisch sein.

Variablen (cont.):

Es sollten selbsterklärende Namen vergeben. Das hat den Vorteil, dass jeder, der sich später mit dem Programm befasst, sofort zurechtfinden kann. Einige Beispiele:

`$Startmeldung`, `$Temperaturwert`, `$XKoordinate`

```
<?php
  $zahl_1      = 14;
  $zahl_2      = 8.3;
  $ergebnis    = $zahl_1 + $zahl_2;
  echo $ergebnis;
?>
```

Variablen (cont.):

Wie man im folgenden Beispiel sieht, kann man einer Variablen mehrmals Werte zuweisen (siehe **`$ergebnis`**).

```
<?php
  $zahl_1      = 14;
  $zahl_2      = 8.3;
  $ergebnis    = $zahl_1 + $zahl_2;
  echo "Ergebnis 1: $ergebnis<br />\n";
  $ergebnis    = $ergebnis - $zahl_2;
  echo "Ergebnis 2: $ergebnis<br />\n";
?>
```

Aufgabe 2:

Das PHP Script/Snippet von der vorherigen Seite erstellen. Im DocumentRoot unter dem Namen **script-02.php** speichern und mit einem beliebigen Browser aufrufen.

Datentypen:

Innerhalb eines Programms können Informationen zur späteren Verwendung in Variablen gespeichert werden. Diese Variablen werden nach Datentyp unterschieden. PHP unterstützt Datentypen für:

- ganze Zahlen
- Zahlen mit Nachkommastellen
- Zeichenketten (Strings)
- Arrays (ein- und mehrdimensionale Felder von Variablen)
- Objekte

>> Übergabe von Parametern

PHP kann Werte aus einem Script an ein anderes Script übergeben. Dazu stehen die Methoden POST und GET zur Verfügung. Bei der Methode POST werden die Werte im HTTP Header übergeben. Sie sind dann für den User nicht direkt sichtbar. Bei der Methode GET werden die Werte in der URL übergeben. Diese sind für den User sicht- und manipulierbar. Dies ist der sogenannte "Searchstring" Part eines URL, also

```
?var1=wert1&var2=wert2&var3=wert3
```

Detailliert behandelt wurde dies in der Vorlesung zu **URL's**.

>> Übergabe von Parametern

Im Folgendem werden wir uns nur mit der GET Methode beschäftigen. Die POST Methode findet nur bei Formularen Anwendung. Die GET Methode kann darüber hinaus auch in URL‘s angewendet werden.

Der Aufbau im "Searchstring" entspricht dem üblichen Muster

```
Variable=Wert
```

Mehrere Variablen werden mit einem "&" getrennt. Bei leeren Variablen wird nach dem Gleichheitszeichen einfach die Wertzuweisung weggelassen. Die Reihenfolge der Variablen ist beliebig.

>> Übergabe von Parametern

Die per POST oder GET übergebenen Variablen konnten in früheren Versionen von PHP direkt genutzt werden. Dies geht bei den aktuellen Versionen aus Sicherheitsgründen nicht mehr. Die hierfür zuständige Direktive (register_globals) ist defaultmäßig auf **OFF** gestellt – **und das sollte so auch bleiben**.

Um eine übergebene Variable nutzen zu können, muss diese aus dem "Global Scope" mit z. B.

```
$variable = $_GET['variable'];
```

in das Script übernommen werden.

Ausgabe mit echo erzeugen:

Wenn man mit PHP arbeitet, möchte man ja auch etwas ausgeben, und zwar üblicherweise HTML-Code. Dies funktioniert mit verschiedenen Funktionen wie zum Beispiel echo, print oder printf. Wir werden hier nur echo benutzen.

Mit echo können wir einen Text ausgeben. Dazu wird der auszugebende Text als String Parameter an den echo Befehl gehängt. Es können auch mehrere durch Komma oder Punkt getrennte Strings übergeben werden.

```
echo (string_1 [. string_n]);
```

Ausgabe mit echo erzeugen (cont.):

Eine PHP-Datei könnte beispielsweise so aussehen:

```
<?php
  echo "Dies ist eine Zeichenkette f&uuml;r HTML";
?>
```

Wie man sieht fehlen hier die Klammern. Das liegt daran, dass echo keine echte Funktion sondern ein PHP-Sprachkonstrukt ist. Sie benötigen keine Klammer. Wenn mehrere Parameter übergeben werden, darf keine Klammer verwendet werden.

>> Ausgabe mit echo

… ohne Zeilenumbruch im HTML Code:

HTML Source

```
<html>
<body>
Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML</body>
</html>
```

PHP Source

```
<html>
<body>
<?php
  echo "Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML";
?>
</body>
</html>
```

>> Ausgabe mit echo

… mit Zeilenumbruch im HTML Code:

HTML Source

```
<html>
<body>
Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML
</body>
</html>
```

PHP Source

```
<html>
<body>
<?php
  echo "Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML\n";
?>
</body>
</html>
```

>> Ausgabe mit echo

… mit einer Variablen:

HTML Source

```
<html>
<body>
Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML
</body>
</html>
```

PHP Source

```
<html>
<body>
<?php
  $var = "Zeichenkette";
  echo "Dies ist eine <b>$var</b> f&uuml;r HTML\n";
?>
</body>
</html>
```

>> Ausgabe mit echo

… mit einer Variablen und Verknüpfung:

HTML Source

```
<body>
Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML
</body>
```

PHP Source

```
<body>

<?php
  $var_1 = "Zeichen";
  $var_2 = $var_1 . "kette";
  echo "Dies ist eine <b>$var_2</b> f&uuml;r HTML\n";
?>

</body>
```

>> Ausgabe mit echo

… mehrere Strings verknüpfen:

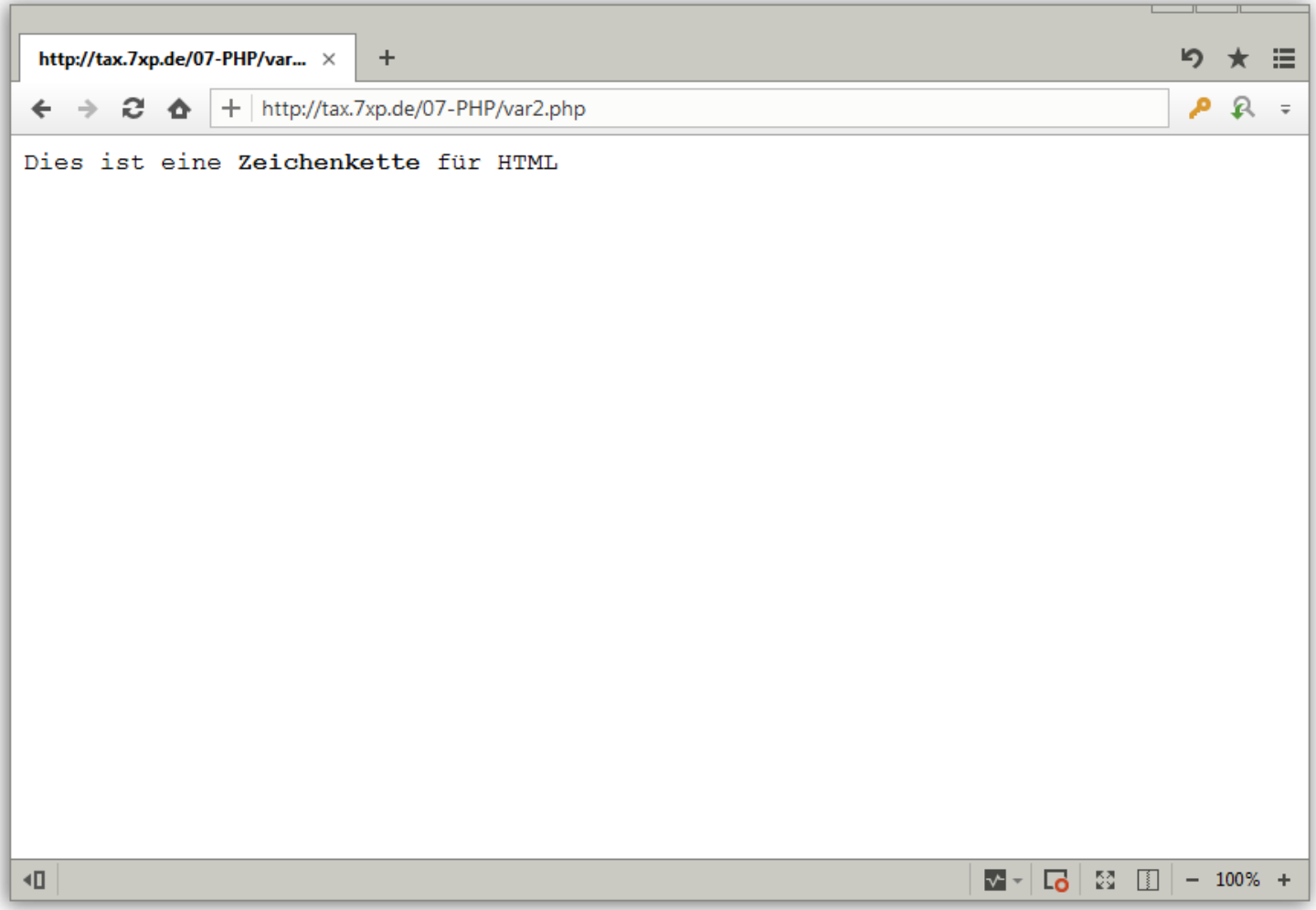
HTML Source

```
<html>
<body>
Dies ist eine <b>Zeichenkette</b> f&uuml;r HTML
</body>
</html>
```

PHP Source

```
<html>
<body>
<?php
  $var = "Zeichen" . "kette";
  echo "Dies ist eine <b>$var</b> f&uuml;r HTML\n";
?>
</body>
</html>
```

>> Und so sieht es im Browser aus



>> Ausgabe mit echo

… echo Parameter in einfachen Anführungszeichen:

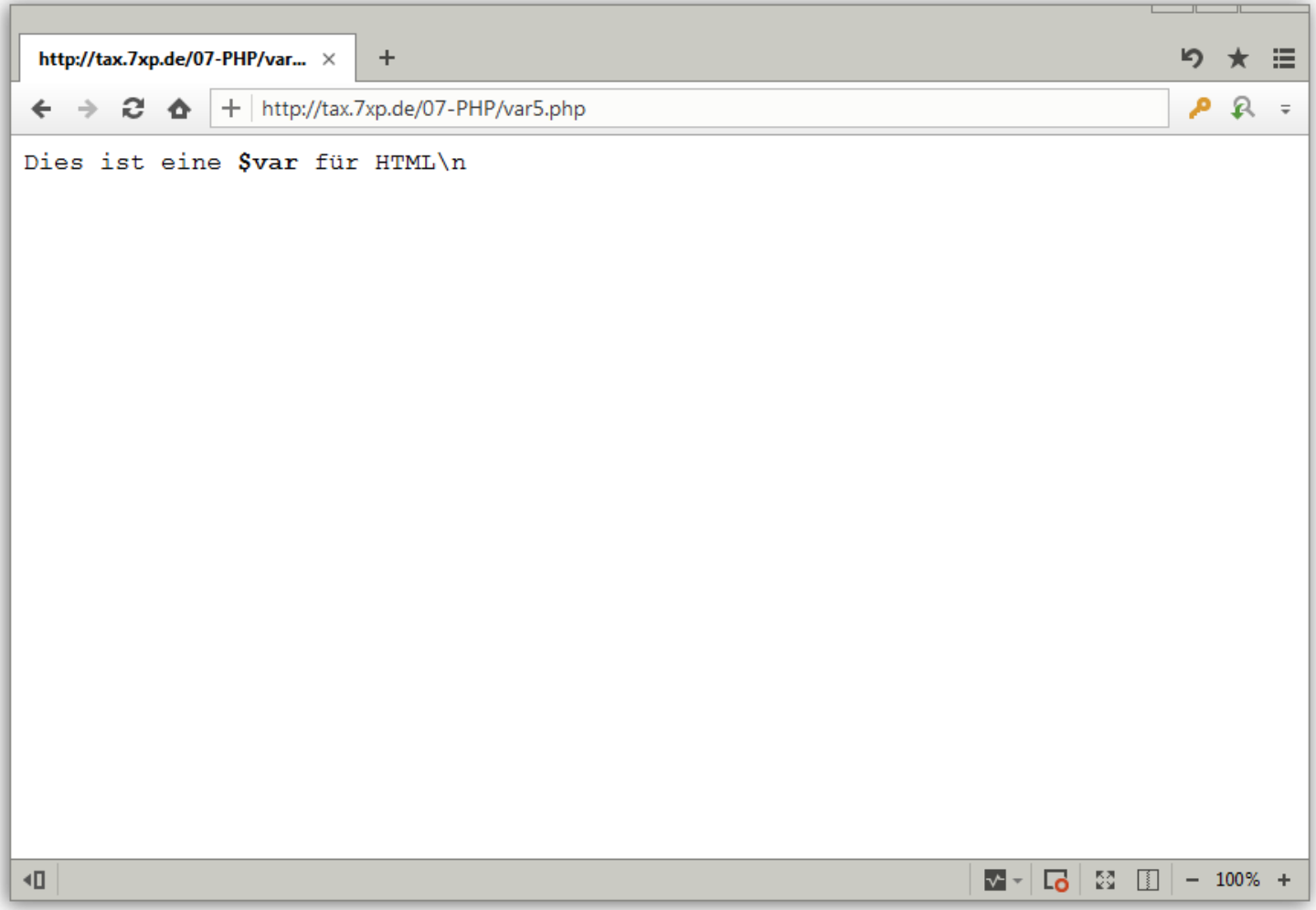
HTML Source

```
<html>
<body>
Dies ist eine <b>$var</b> f&uuml;r HTML\n</body>
</html>
```

PHP Source

```
<html>
<body>
<?php
  $var = "Zeichenkette";
  echo 'Dies ist eine <b>$var</b> f&uuml;r HTML\n';
?>
</body>
</html>
```

>> Und so sieht es im Browser aus



>> Arithmetische Operatoren

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulo (Rest bei einer ganzzahligen Division) Z. B. ergibt 7%3 den Wert 1, denn 7 durch 3 ergibt 2 Rest 1

>> Übergabe von Parametern mit GET

… zum setzen einer Farbe:

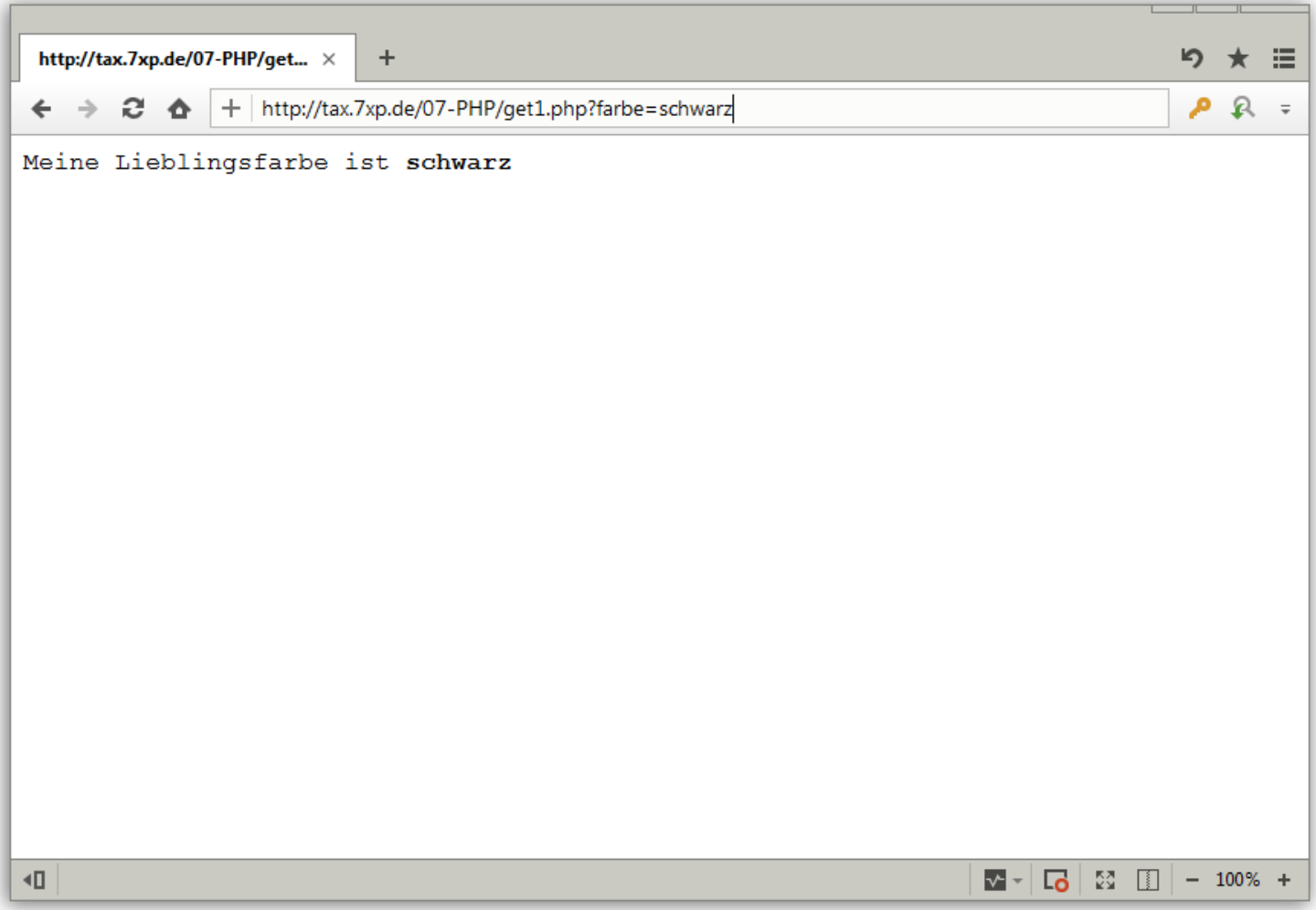
HTML Source

```
<html>
<body>
Meine Lieblingsfarbe ist <b>schwarz</b>
</body>
</html>
```

PHP Source

```
<html>
<body>
<?php
  $farbe = $_GET['farbe'];
  echo "Meine Lieblingsfarbe ist <b>$farbe</b>\n";
?>
</body>
</html>
```

>> Und so sieht es im Browser aus



>> Übergabe von Parametern mit GET

… zur Addition:

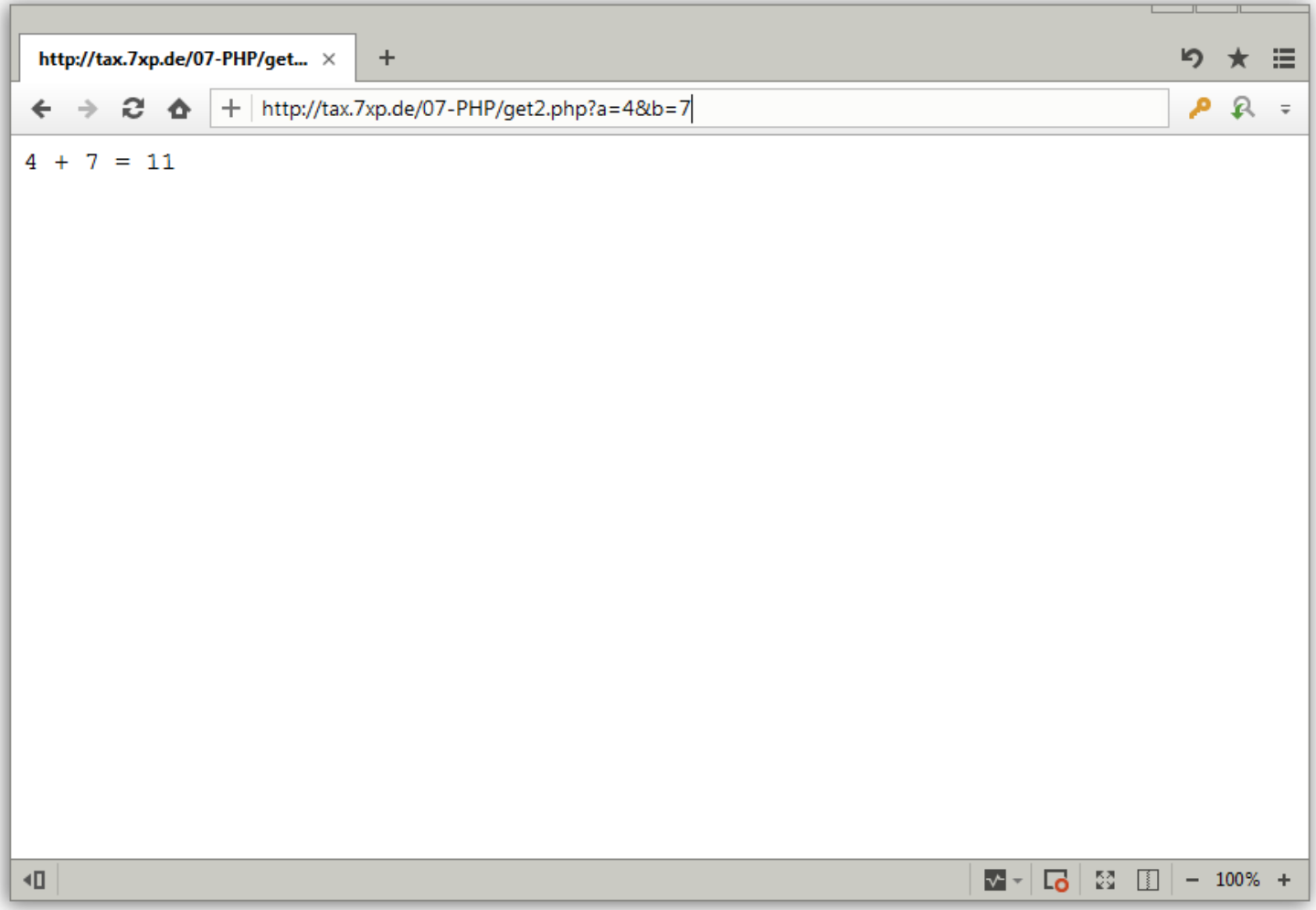
HTML Source

```
4 + 7 = 11
```

PHP Source

```
<html>
<body>
<?php
  $a = $_GET['a'];
  $b = $_GET['b'];
  echo "$a + $b = " , $a + $b;
?>
</body>
</html>
```

>> Und so sieht es im Browser aus



>> Übergabe von Parametern mit GET

… zur MwSt-Berechnung:

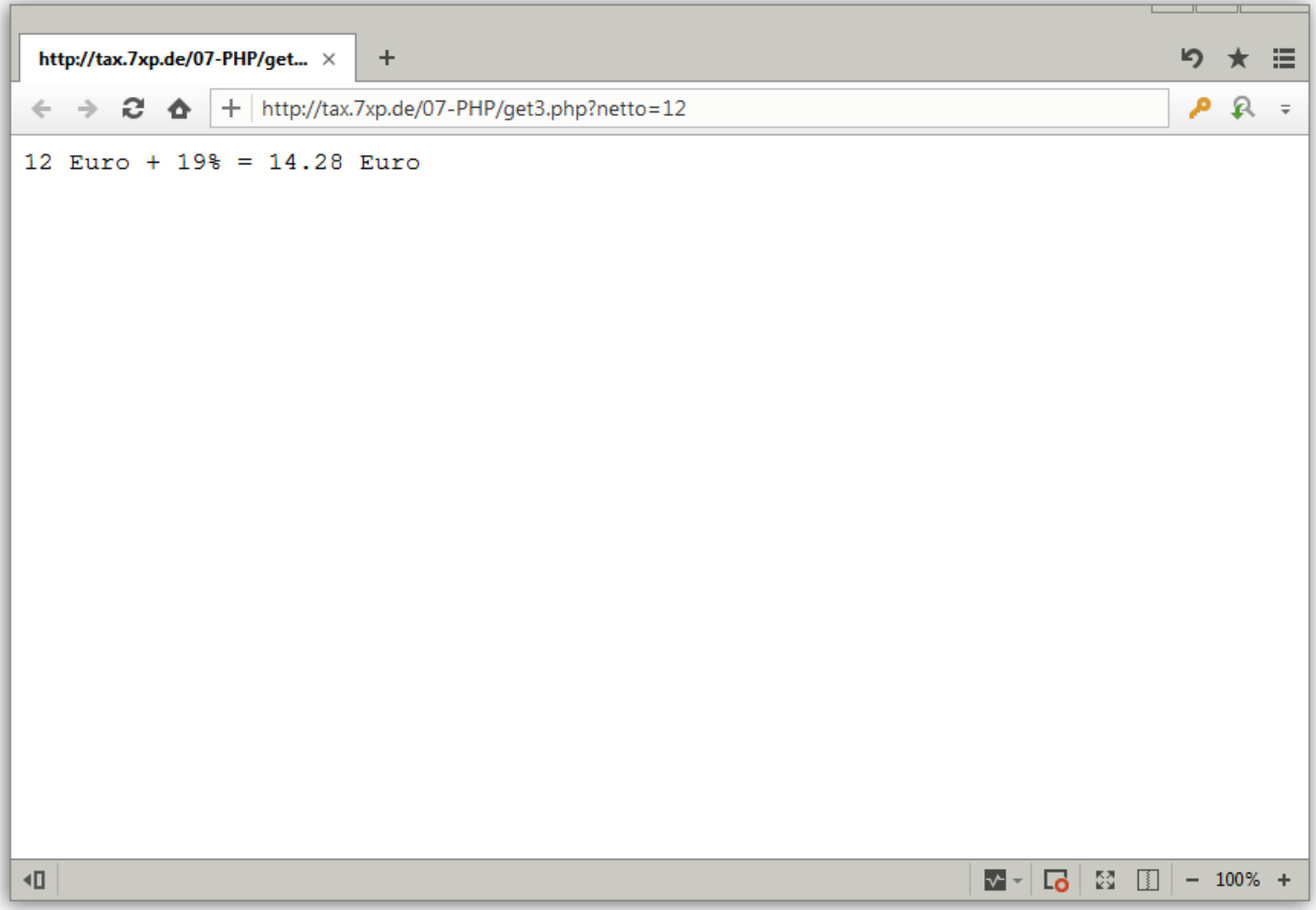
HTML Source

```
12 Euro + 19% = 14.28 Euro
```

PHP Source

```
<?php
  define('_MWST_', 0.19);          // eine Konstante
  $netto = $_GET['netto'];
  $brutto = $netto * (1 + _MWST_);

  echo "$netto Euro + " , _MWST_ * 100 , "% = $brutto
Euro";
?>
```

>> Und so sieht es im Browser aus



>> Vergleichs-Operatoren

Operator	Bedeutung	Ergebnis
$a == $b	**Gleich**	**TRUE**, wenn $a gleich $b.
$a === $b	**Identisch**	**TRUE**, wenn $a gleich $b und beide vom gleichen Typ.
$a != $b **$a <> $b**	**Ungleich**	**TRUE**, wenn $a nicht gleich $b.
$a !== $b	**Nicht identisch**	**TRUE**, wenn $a nicht gleich $b oder wenn beide nicht vom gleichen Typ.
$a > $b	**größer als**	**TRUE**, wenn $a größer als $b.
$a < $b	**kleiner als**	**TRUE**, wenn $a kleiner als $b.
$a >= $b	**größer oder gleich**	**TRUE**, wenn $a größer oder gleich $b.
$a <= $b	**kleiner oder gleich**	**TRUE**, wenn $a kleiner oder gleich $b.

>> Verzweigung mit IF

… einfache Auswahl

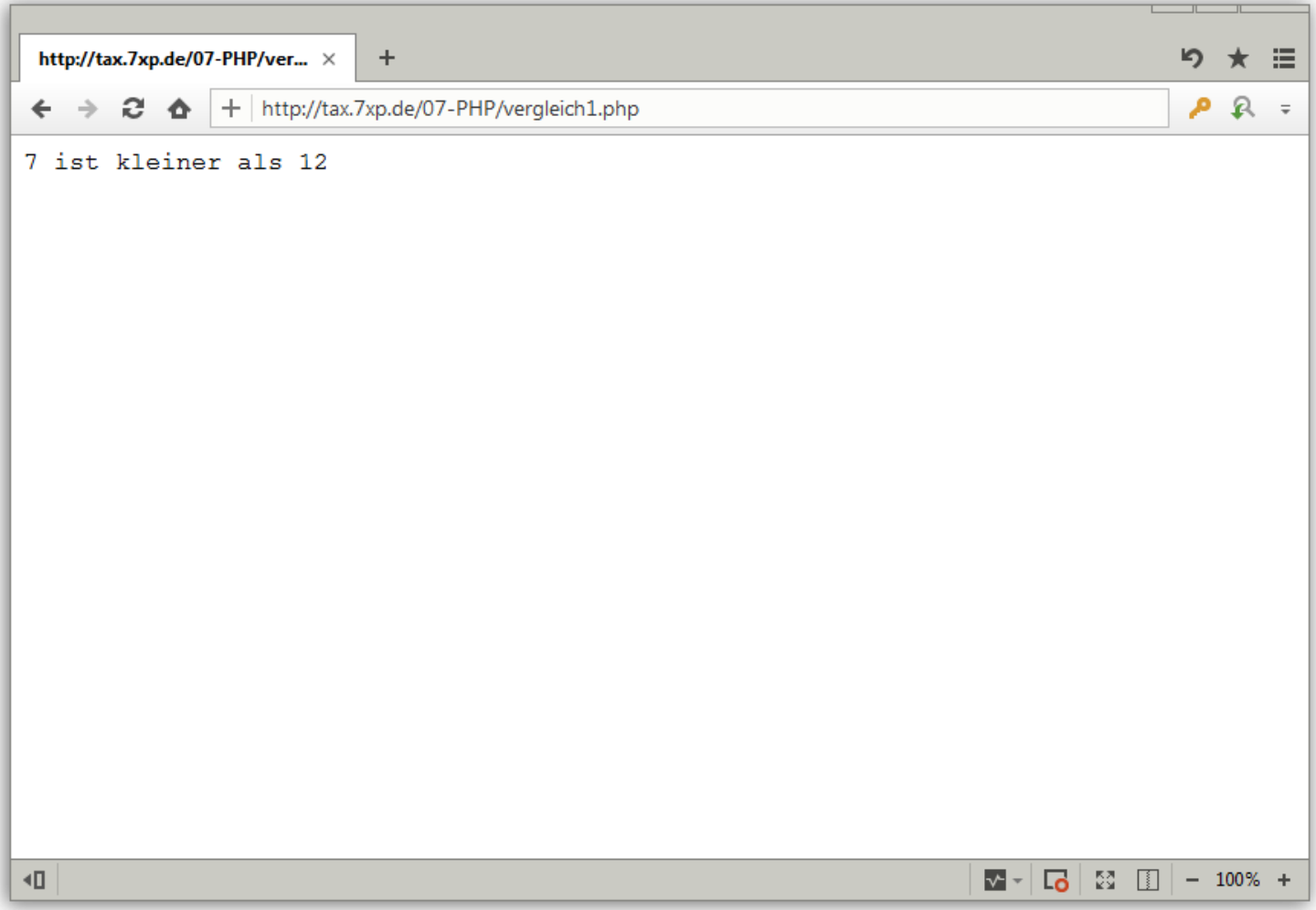
HTML Source

```
7 ist kleiner als 12
```

PHP Source

```
<?php
  $a = 7;
  $b = 12;

  if($a < $b) {
    echo "$a ist kleiner als $b";
  }
?>
```

>> Und so sieht es im Browser aus



>> Verzweigung mit IF

… kombiniert mit GET (1):

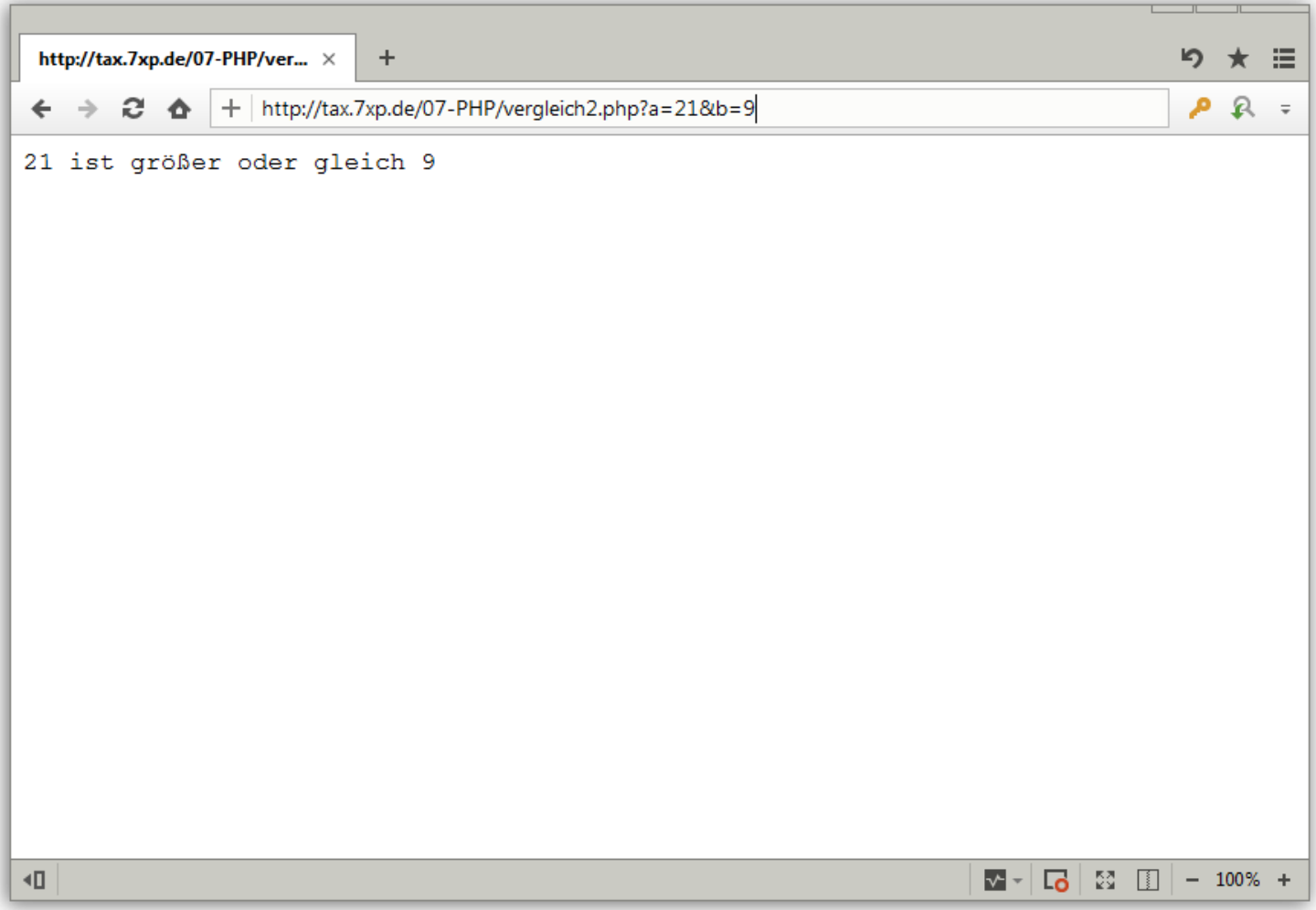
HTML Source

```
21 ist größer oder gleich 9
```

PHP Source

```
<?php
  $a = $_GET['a'];
  $b = $_GET['b'];

  if($a < $b) {
    echo "$a ist kleiner als $b";
  } else {
    echo "$a ist gr&ouml;&szlig;er oder gleich $b";
  }
?>
```

>> Und so sieht es im Browser aus



>> Verzweigung mit IF

… kombiniert mit GET (2):

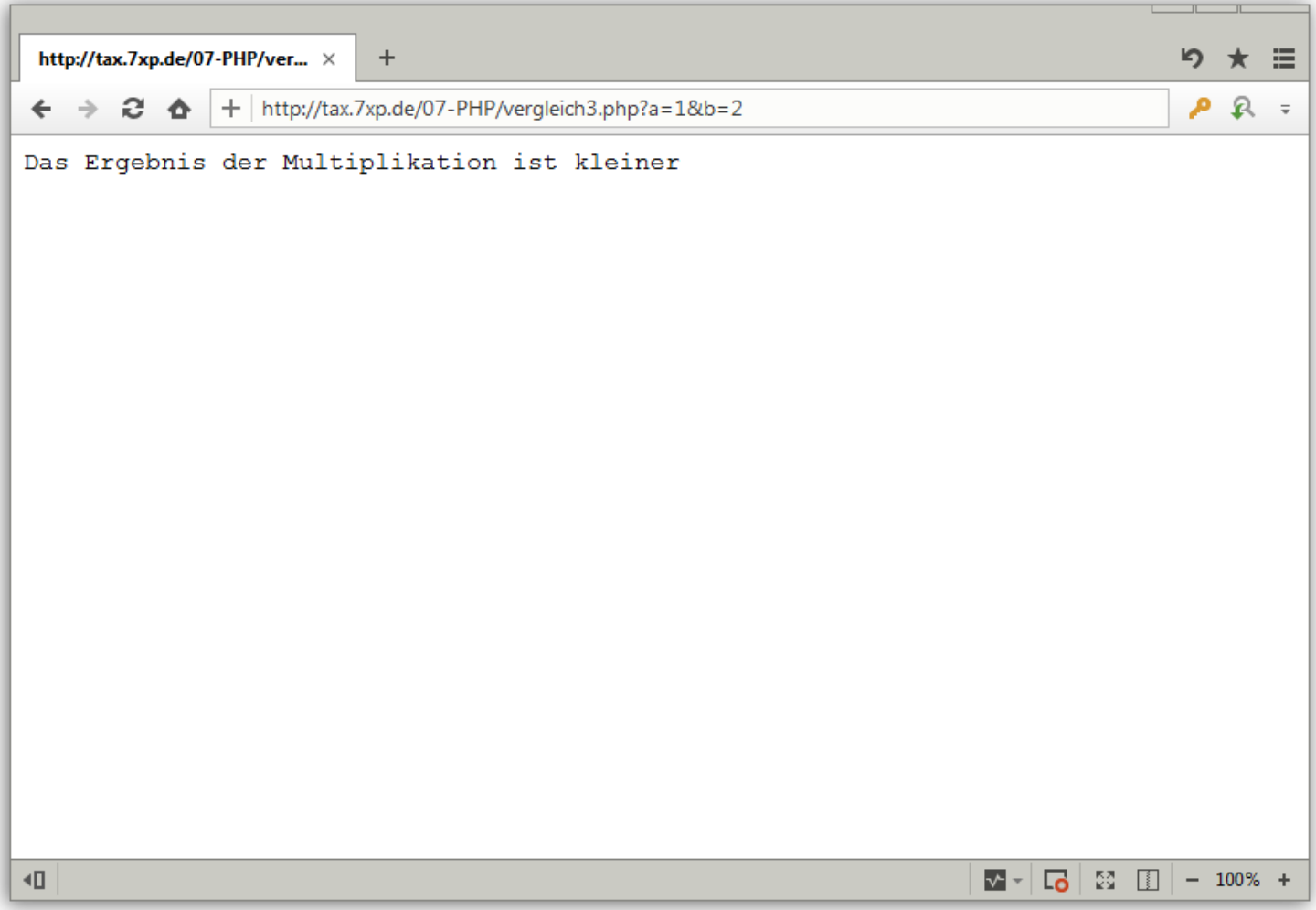
HTML Source

```
Das Ergebnis der Multiplikation ist kleiner
```

PHP Source

```
<?php
  $a = $_GET['a'];
  $b = $_GET['b'];

  if($a * $b < $a + $b) {
    echo "Das Ergebnis der Multiplikation ist kleiner";
  } else {
    echo "Das Ergebnis der Addition ist kleiner";
  }
?>
```

>> Und so sieht es im Browser aus



>> FOR-Schleifen

… Fakultätsberechnung:

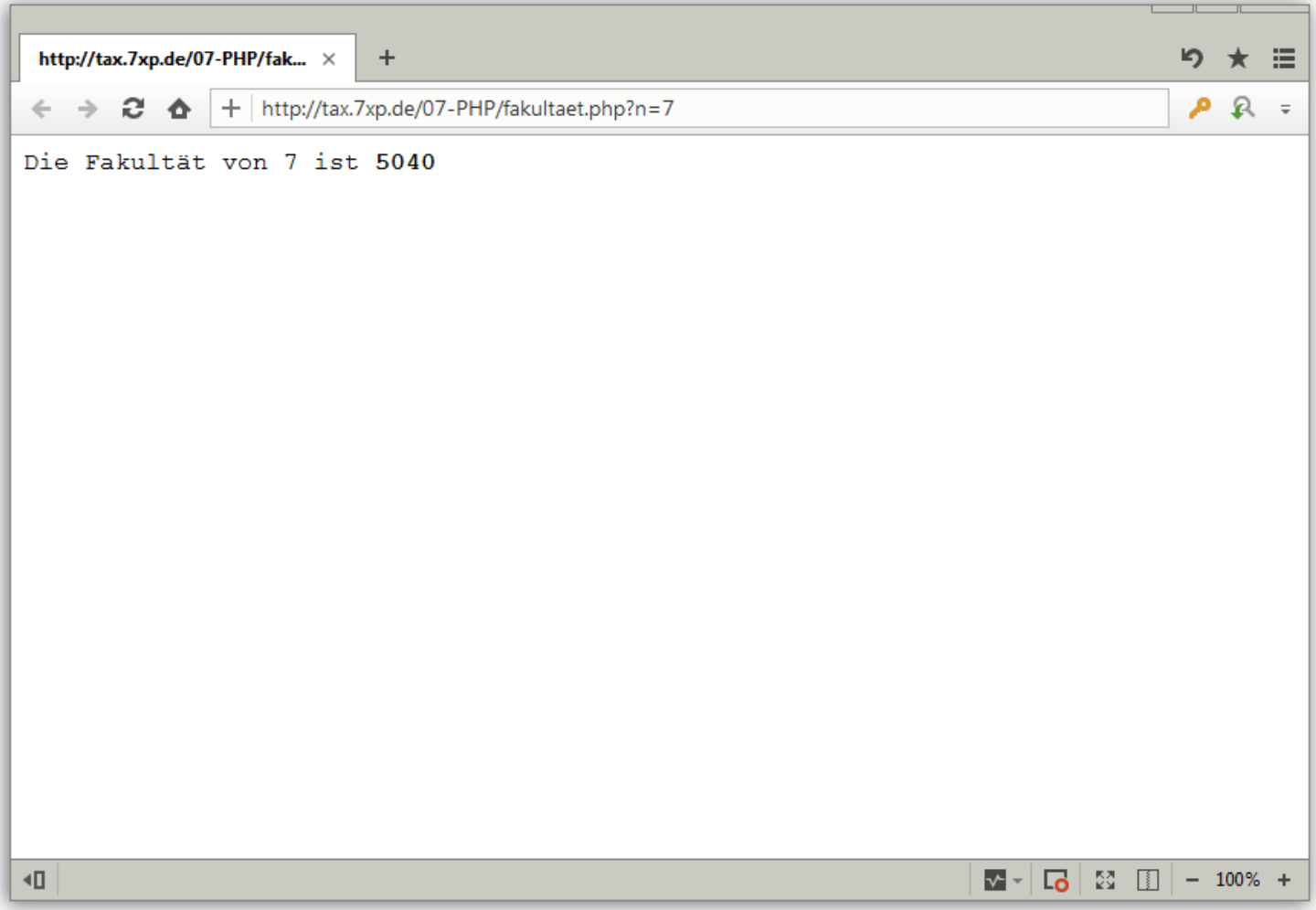
HTML Source

```
Die Fakultät von 7 ist <b>5040</b>
```

PHP Source

```
<?php
  $n = $_GET['n'];
  $fakultaet = 1;

  for($i = 1; $i <= $n; $i++) { // $i++: $i = $i +1
    $fakultaet = $fakultaet * $i;
  }

  echo "Die Fakult&auml;t von $n ist <b>$fakultaet</b>";
?>
```

>> Und so sieht es im Browser aus



>> FOR-Schleifen

… Summenberechnung - nur Ende variable:

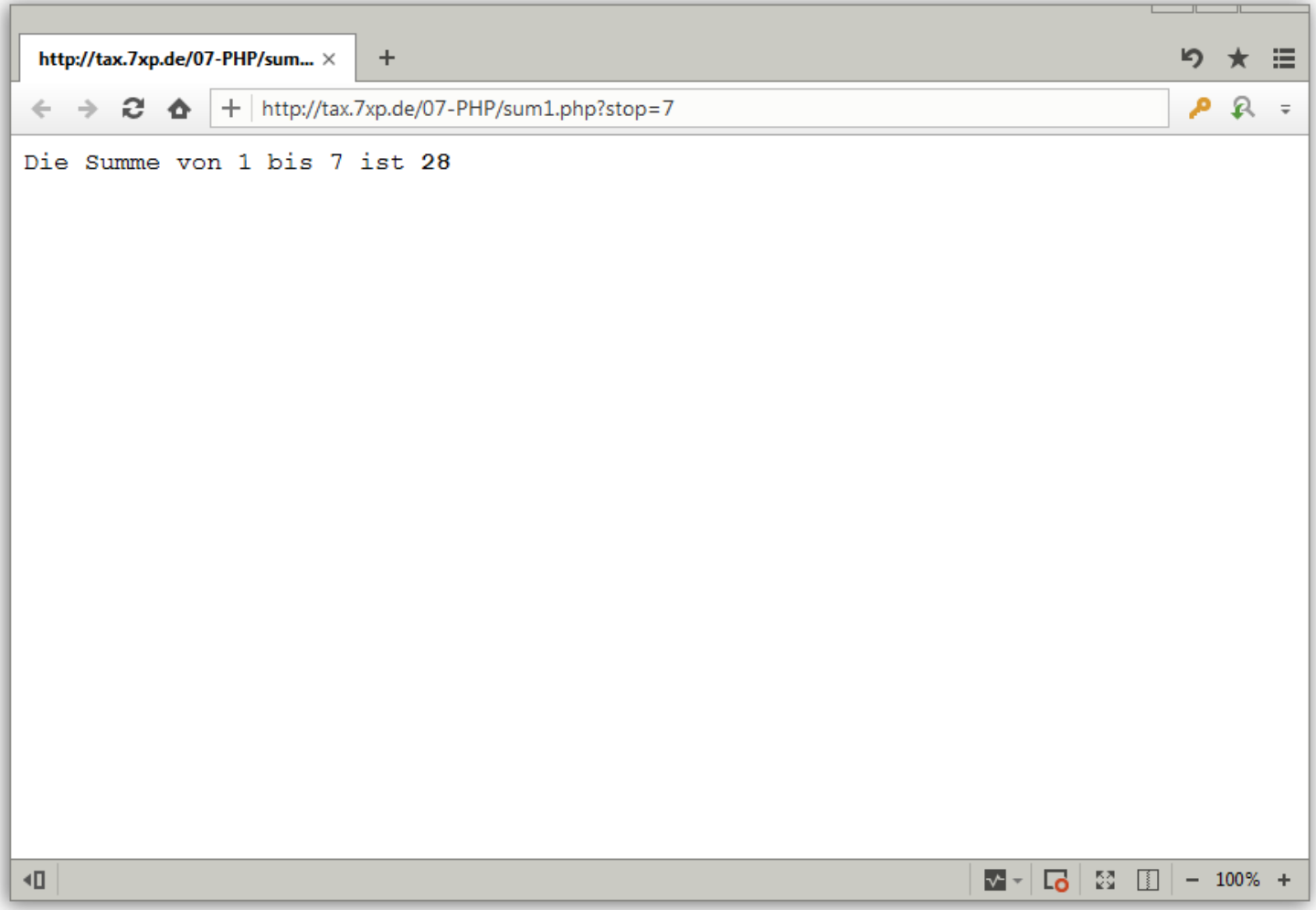
HTML Source

```
Die Summe von 1 bis 7 ist <b>28</b>
```

PHP Source

```
<?php
  $stop = $_GET['stop'];
  $summe = 0;

  for($i = 1; $i <= $stop; $i++) {
    $summe = $summe + $i;
  }

  echo "Die Summe von 1 bis $stop ist <b>$summe</b>";
?>
```

>> Und so sieht es im Browser aus



>> FOR-Schleifen

… Summenberechnung - Anfang und Ende variable:

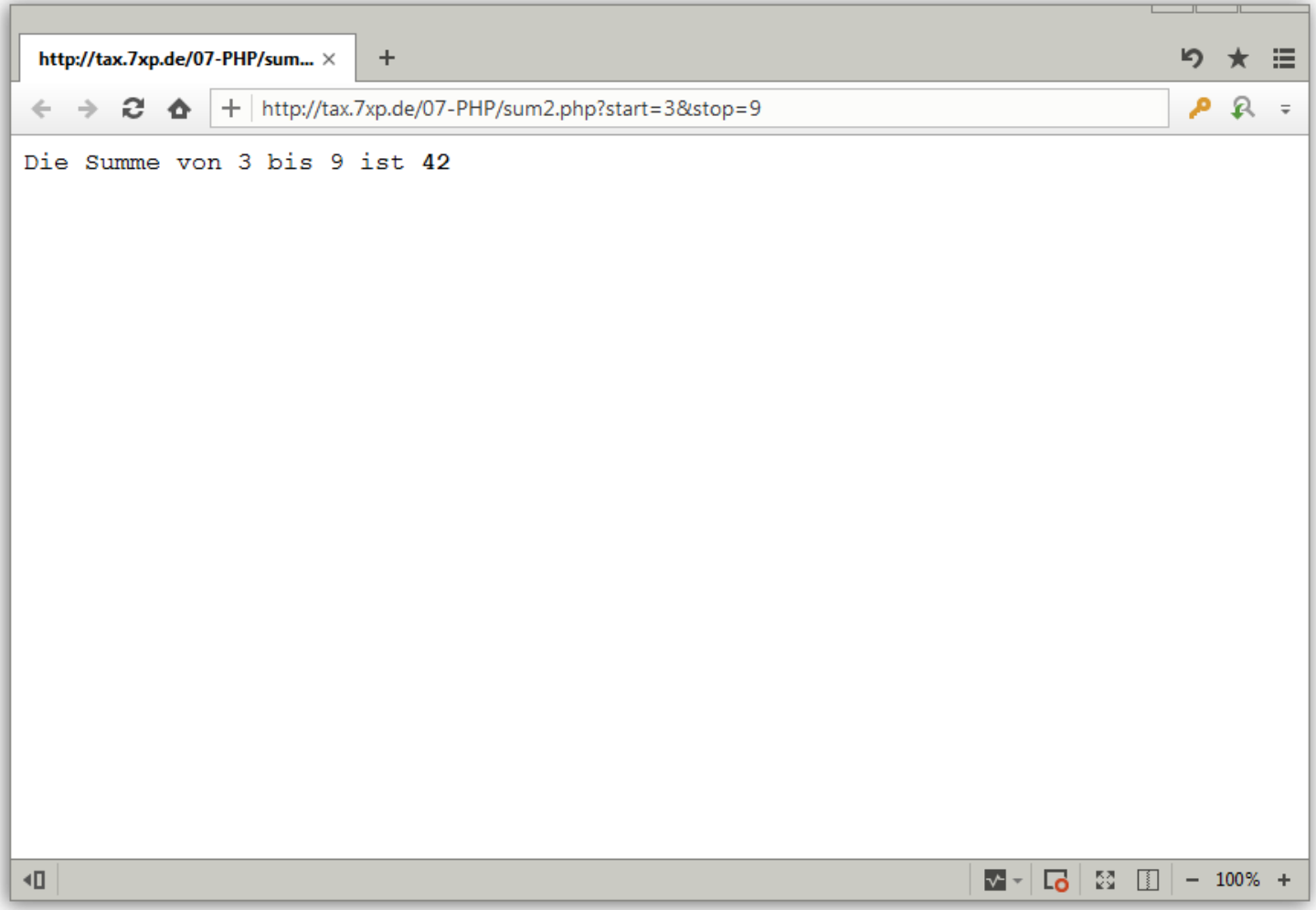
HTML Source

```
Die Summe von 3 bis 9 ist <b>42</b>
```

PHP Source

```
<?php
  $start = $_GET['start'];
  $stop  = $_GET['stop'];
  $summe = 0;

  for($i = $start; $i <= $stop; $i++) {
    $summe = $summe + $i;
  }
  echo "Die Summe von $start bis $stop ist <b>$summe</b>";
?>
```

>> Und so sieht es im Browser aus



>> FOR-Schleifen

… zweispaltige Tabelle mit GET erstellen (Variante 1):

HTML Source

```
<table border="1"><tr><td> </td><td>
 </td></tr><tr><td> </td><td> 
</td></tr></table>
```

PHP Source

```
<?php
  $rows = $_GET['rows'];
  // Zweispaltige Tabelle mit $rows Zeilen erstellen
  echo "<table border=\"1\">";
  for($i = 0; $i < $rows; $i++) {
    echo "<tr><td width=\"150\"> </td>";
    echo "<td width=\"300\"> </td></tr>\n";
  }
  echo "</table>\n";
?>
```

>> FOR-Schleifen

… zweispaltige Tabelle mit GET erstellen (Variante 2):

HTML Source

```
<table border="1"><tr><td> </td><td>
 </td></tr><tr><td> </td><td> 
</td></tr></table>
```

PHP Source

```
<table border="1">
<?php
  $rows = $_GET['rows'];
  // Zweispaltige Tabelle mit $rows Zeilen erstellen
for($i = 0; $i < $rows; $i++) {
    echo '<tr><td width="150"> </td>';
    echo "<td width=\"300\"> </td></tr>\n";
  }
?>
</table>
```

>> FOR-Schleifen

… zweispaltige Tabelle mit GET erstellen (Variante 3):

HTML Source

```
<table border="1"><tr><td> </td><td>
 </td></tr><tr><td> </td><td> 
</td></tr></table>
```

PHP Source

```
<table border="1">
<?php
  $rows = $_GET['rows'];
for($i = 0; $i < $rows; $i++) { ?>
<tr><td width="150"> </td>
<td width="300"> </td></tr>
<?php } // Ende FOR ?> <!-- Ende FOR -->
</table>
```

>> FOR-Schleifen

… zweispaltige Tabelle mit GET erstellen (Variante 4):

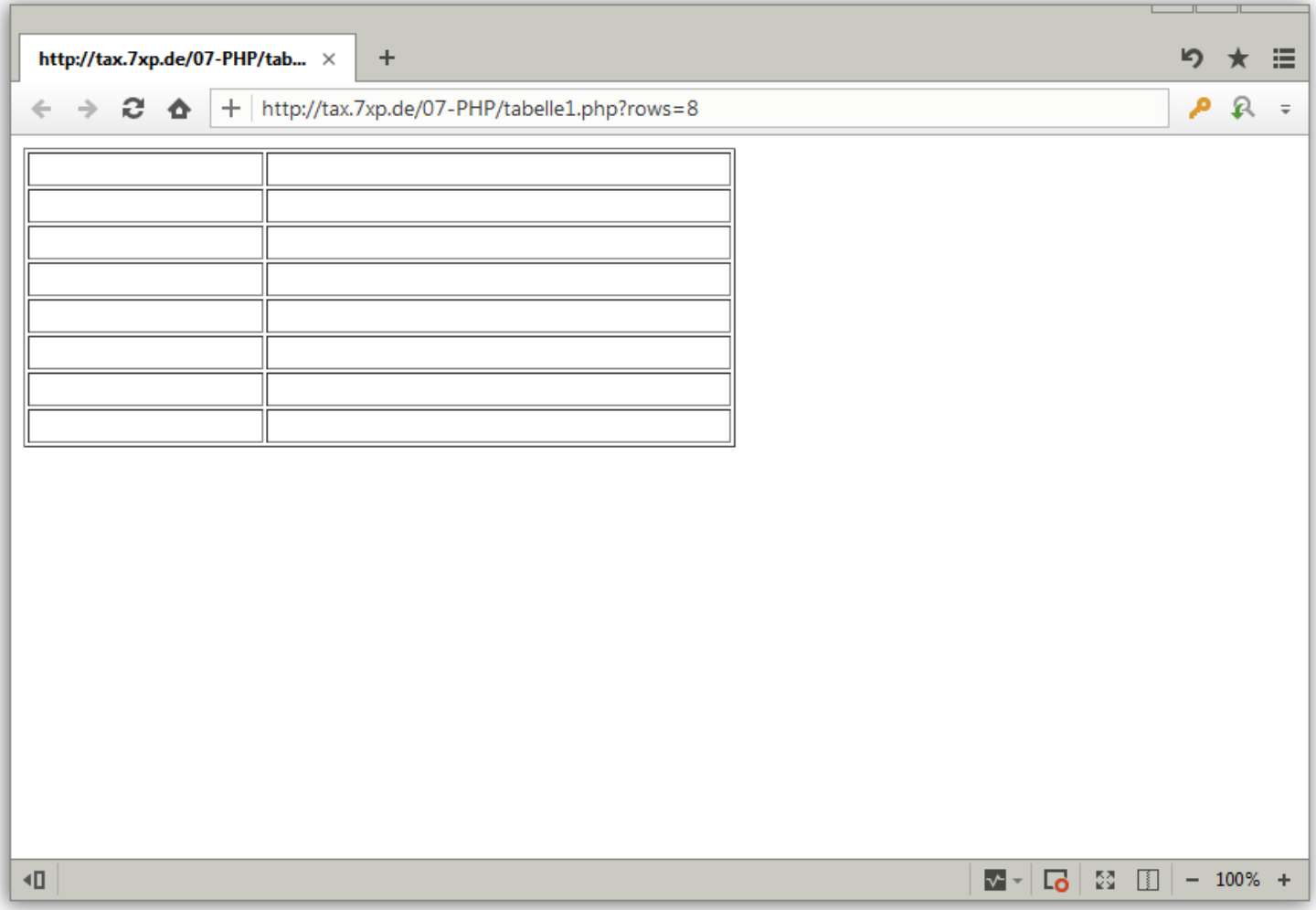
HTML Source

```
<table border="1"><tr><td> </td><td>
 </td></tr><tr><td> </td><td> 
</td></tr></table>
```

PHP Source

```
<table border="1">
<?php
  $rows = $_GET['rows'];
for($i = 1; $i <= $rows; $i++) { ?>
<tr><td width="150"><?php echo $i; ?>.</td>
<td width="300"> </td></tr>
<?php } // Ende FOR ?> <!-- Ende FOR -->
</table>
```

>> Und so sieht es im Browser aus



>> Funktionssammlung einbinden

Die **phpIniFunctions** unter http://docs.tx7.de/TT-96L downloaden und im DocumentRoot (public_html) auspacken. Im gleichen Verzeichnis die INI-Datei dhbw.ini mit folgendem Inhalt erstellen:

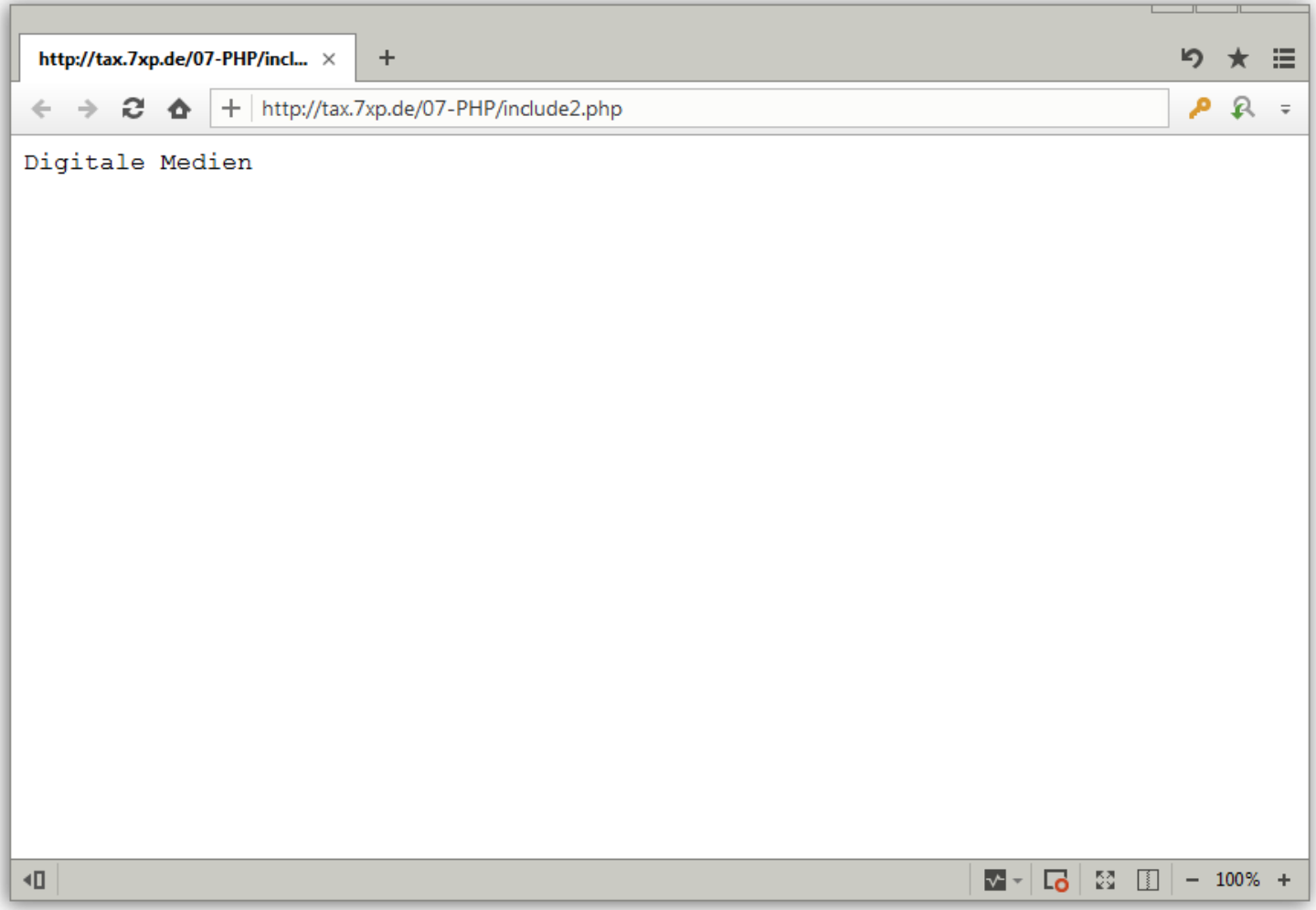
```
[DHBW]
Studiengang=Digitale Medien

[Tabelle 1]
1=Alfa
2=Bravo
3=Charly
4=Delta
5=Echo
```

>> Funktionssammlung einbinden

Die **phpIniFunctions** werden am Anfang des Scripts mit **require_once** eingebunden. Dabei auf den korrekten Pfad achten. **ReadIniValue** ist bereits eine Funktion der phpIniFunctions.

```
<?php

// Funktionssammlung einbinden.
require_once('phpIniFunctions.inc.php');


// Einen Wert aus der INI-Datei auslesen und hier ausgeben
echo ReadIniValue('dhbw.ini', 'DHBW', 'Studiengang');

?>
```

>> Und so sieht es im Browser aus



>> Funktionssammlung einbinden

Tabelle mit Werten aus einer INI-Datei erstellen.

PHP Source

```
<table border="1">

<?php
  require_once("phpIniFunctions.inc.php");

  $rows = $_GET['rows'];
  for($i = 1; $i <= $rows; $i++) : ?>

  <tr><td width="150"><?php echo $i; ?>.</td>
  <td width="300">
  <?php echo ReadIniValue("dhbw.ini", "Tabelle 1", $i); ?>
  </td></tr>

<?php endfor; ?>

</table>
```

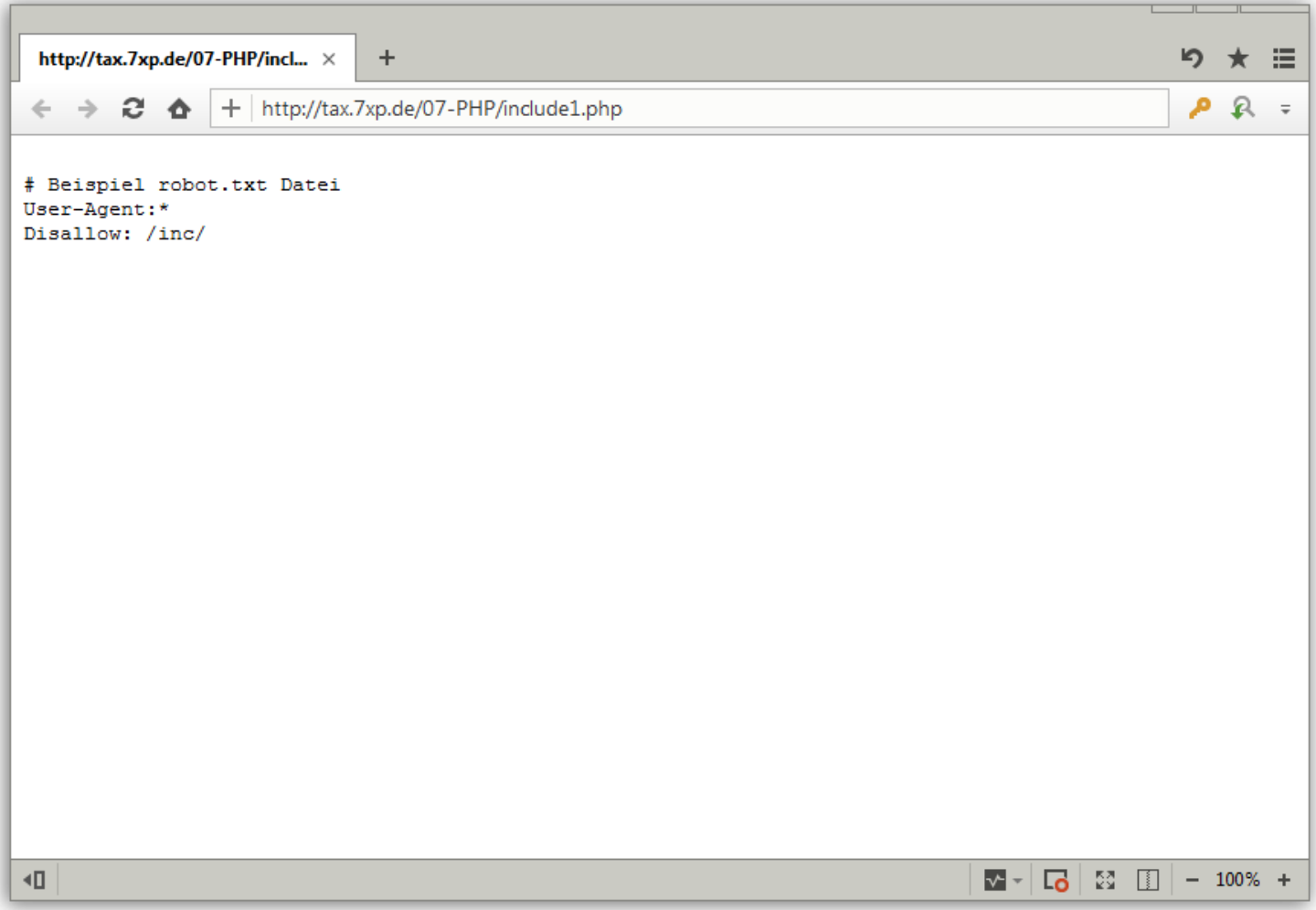
>> Dateien einbinden

Beispiel statische TXT-Datei:

PHP Source

```
<pre>

<?php

  // Eine Datei komplett einlesen und ausgeben
  readfile('robots.txt');

?>

</pre>
```

>> Und so sieht es im Browser aus



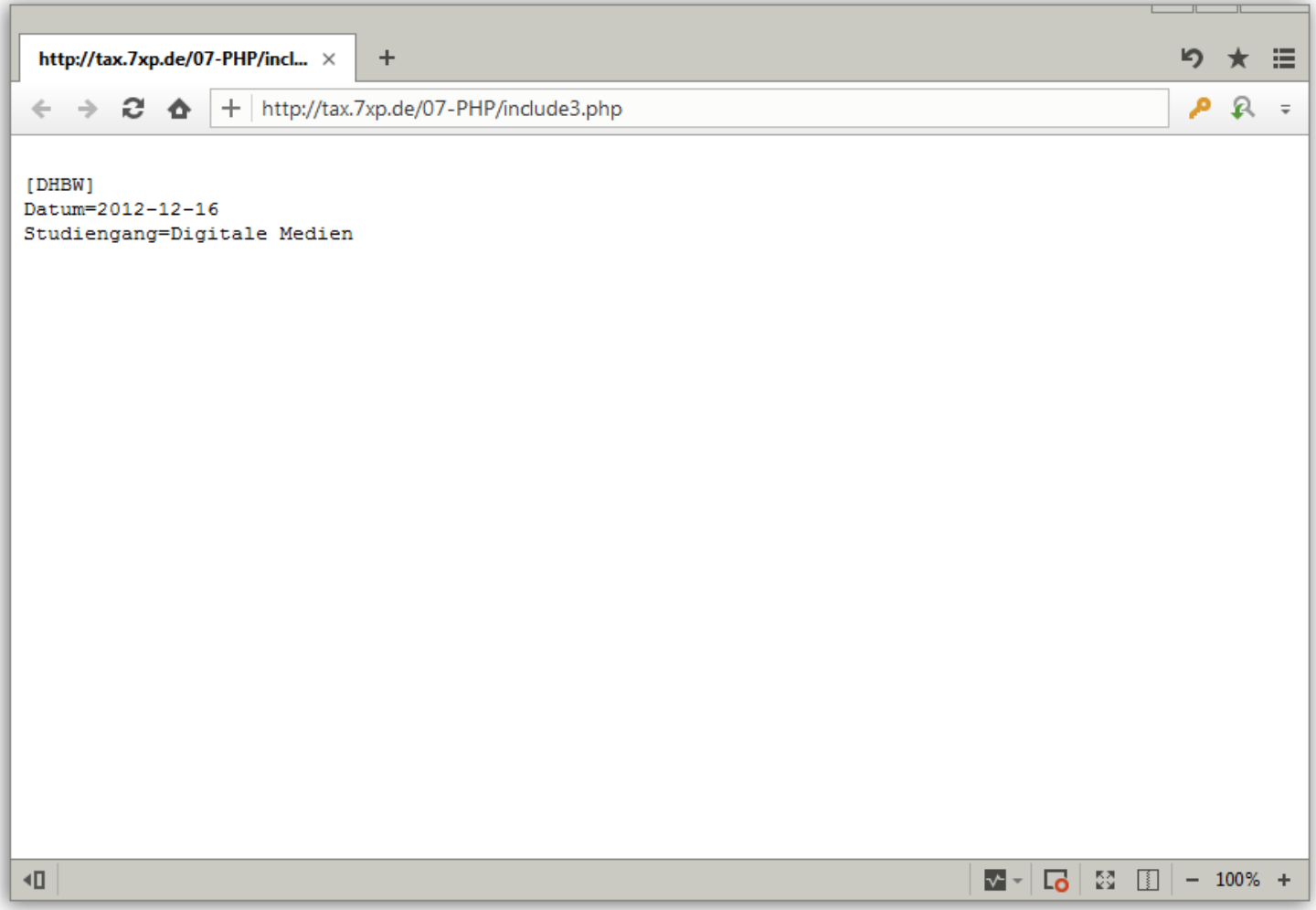
>> Dateien einbinden

… weitere PHP Funktionen:

PHP Source

```php
<?php
  date_default_timezone_set("Europe/Berlin");

  // Funktionen einbinden
  require_once('phpIniFunctions.inc.php');

  // INI Value in die Datei tux.ini schreiben
  WriteIniValue('../data/tux.ini', 'DHBW', 'Datum',
       date('Y-m-d H:i:s'));
?>

<pre>
<!-- Ausgabe der INI mit den soeben geschriebenen Werten -->
<?php readfile('../data/tux.ini'); ?>
</pre>
```

>> Und so sieht es im Browser aus



… Affenformular 1 - ohne Fehlerbehandlung:

PHP Source

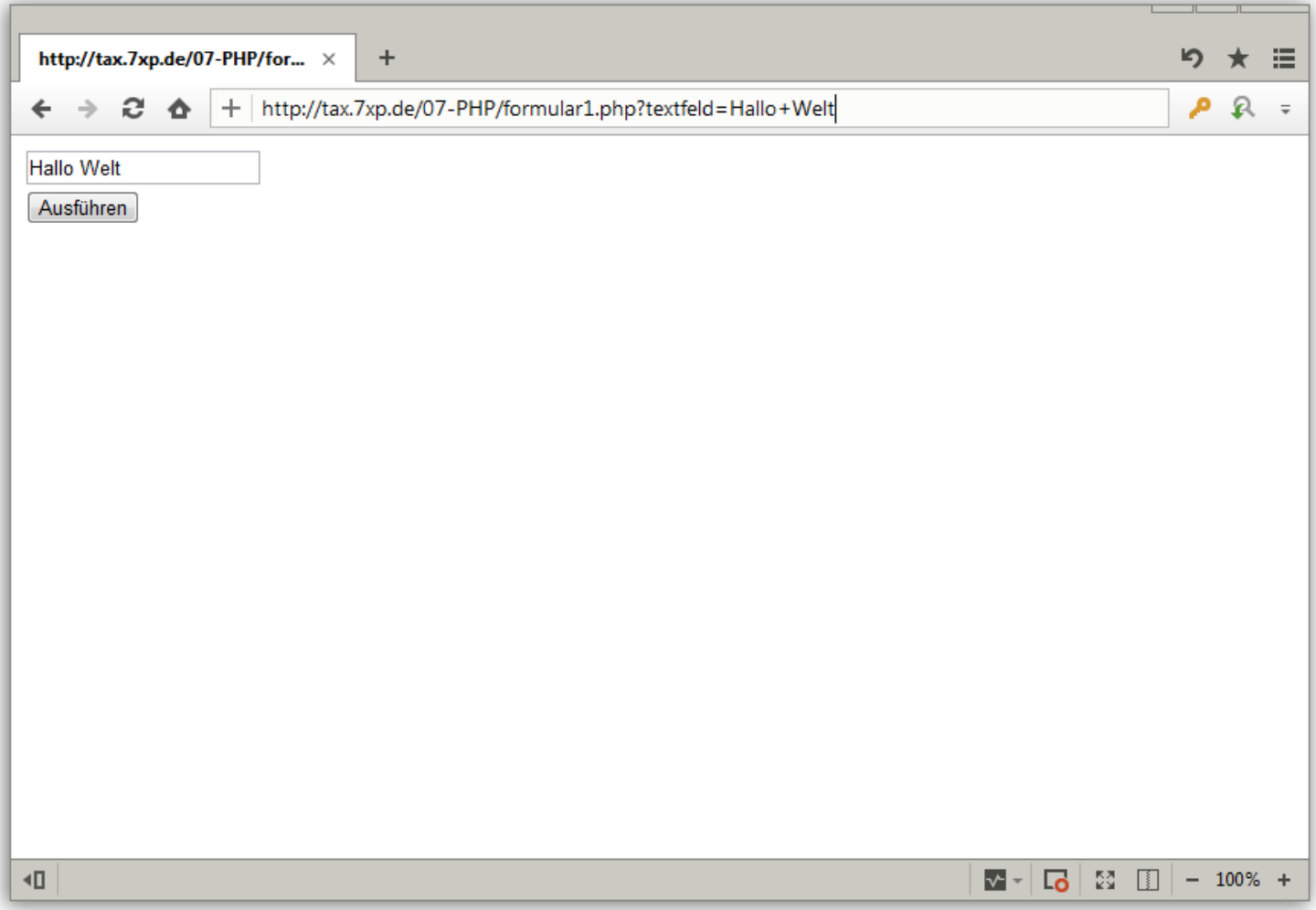
```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>">
<input type="text"
       name="textfeld"
       value="<?php echo $_REQUEST['textfeld']; ?>">
<br>
<input type="submit"
       name="Form_x"
       value="Ausf&uuml;hren">
</form>
```

… Affenformular 1 - mit Fehlerbehandlung:

PHP Source

```
<?php
 if(isset($_POST['textfeld'])) {
        $textfeld = $_POST['textfeld'];
  } else { $textfeld = ''; }
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
method="post">
<input type="text" name="textfeld"
       value="<?php echo $textfeld; ?>">
<br />
<input type="submit" name="Form_x" value="Ausf&uuml;hren">
</form>
```

>> Und so sieht es im Browser aus



… Affenformular 2 (Seite 1/4):

PHP Source

```php
<?php
  // Formular zum berechnen von zwei Werten
  if(isset($_POST['submit'])) {
     $submit = $_POST['submit'];
  } else { $submit = ''; }

  if(isset($_POST['w1'])) {
     $w1 = $_POST['w1'];
  } else { $w1 = ''; }

  if(isset($_POST['w2'])) {
     $w2 = $_POST['w2'];
  } else { $w2 = ''; }
```

… Affenformular 2 (Seite 2/4):

PHP Source

```
  if(isset($_POST['calc'])) {
     $calc = $_POST['calc'];
  } else { $calc = ''; }

$c_add = '';
$c_sub = '';

 if($submit) {
    switch ($calc) {
      case 'add':
        $ergebnis = $w1 + $w2;
        echo "<b>$w1 + $w2 = $ergebnis</b><br>";
        $c_add = 'checked';
        break;
```

… Affenformular 2 (Seite 3/4):

PHP Source

```
      case 'sub':
        $ergebnis = $w1 - $w2;
        echo "<b>$w1 - $w2 = $ergebnis</b<br>";
        $c_sub = 'checked';
        break;

      default:
        echo "Rechenoperation wird nicht unterst&uuml;tzt";
    }
  }
?>

<p>Bitte tragen Sie zwei Zahlen ein.</p>
```

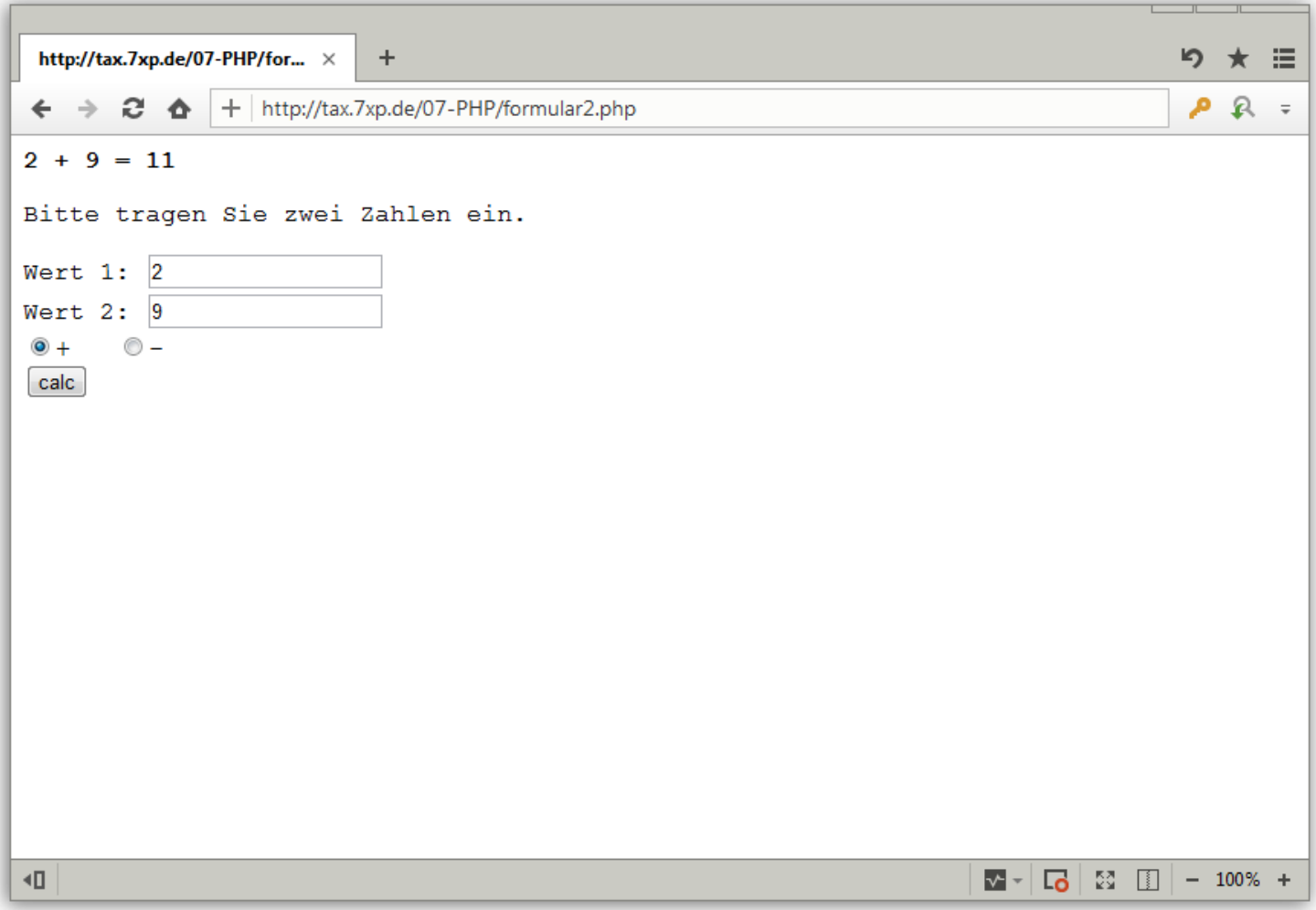
… Affenformular 2 (Seite 4/4):

PHP Source

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
      method="post">
  Wert 1: <input name="w1" value="<?php echo $w1 ?>"><br>
  Wert 2: <input name="w2" value="<?php echo $w2 ?>"><br>
  <input type="radio" name="calc" value="add"
    <?php echo $c_add ?>>+  
  <input type="radio" name="calc" value="sub"
    <?php echo $c_sub ?>>-<br>
  <input type="submit" name="submit" value="calc"><br>
</form>
```

http://docs.tx7.de/TT-F9C

>> Und so sieht es im Browser aus



>> Referenzen

Codecademy:

http://docs.tx7.de/TT-GML

W3Schools - PHP Tutorial (EN):

http://docs.tx7.de/TT-TEK

SelfPHP (DE):

http://docs.tx7.de/TT-ER1

PHP Handbuch auf php.net (DE):

http://docs.tx7.de/TT-V9H

W3C HTML Validator (EN):

http://docs.tx7.de/TT-W3C