**Teil 2**

Formulare-Theorie

HTTP Status Messages

HTML Error Messages

|  |  |
| --- | --- |
| **1xx: Information** |  |
| 100 Continue | Server hat request header erhalten, clien soll request body senden |
| 101 Switching Protocols | Anfragende fragt server an um Protokolle zu ändern |
| 103 Checkpoint | Abgebrochene PUT- oder POST requests wieder aufnehmen |
|  |  |
| **2xx: Successful** |  |
| 200 OK | Standardantwort für erfolgreiche HTTP requests |
| 201 Created | Die Anfrage wurde erfüllt und eine neue Ressource wird erstellt |
| 202 Accepted | Request wird verarbeitet, aber ist noch nicht fertig |
| 203 Non-Authoritative Information | Request wird verarbeitet, liefert aber Informationen, die von einer anderen Quelle sein können. |
| 204 No Content | Request wurde erfolgreich verarbeitet, liefert aber keinen Inhalt |
| 205 Reset Content | Request wurde erfolgreich verarbeitet, liefert aber keinen Inhalt und erfordert neue Übermittlung des Dokuments |
| 206 Partial Conten | Server liefert nur Teile der Ressource, aufgrund eines vom Client gesendeten Range-Headers |
|  |  |
| **3xx: Redirection** |  |
| 300 Multiple Choices | Eine Linkliste. Der Benutzer kann einen Link auswählen und zu diesem Ort gehen. Maximal fünf Adressen |
| 301 Moved Permanently | Die angefragte Seite hat eine neue URL |
| 302 Found | Die angefragte Seite hat zeitweise eine neue URL |
| 303 See Other | Die angefragte Seite kann unter einer anderen URL gefunden werden |
| 304 Not Modified | Zeigt an, dass angefragte Seite seit letztem request nicht verändert wurde. |
| 307 Temporary Redirect | Die angefragte Seite hat zeitweise eine neue URL |
| 308 Resume Incomplete | Wird im Vorschlag für fortsetzbare Anforderungen verwendet, um abgebrochene PUT- oder POST-requests wieder aufzunehmen |
|  |  |
| **4xx: Client Error** |  |
| 400 Bad Request | Falsche Syntax |
| 401 Unauthorized | Anfrage war gültig, aber Server weigert sich zu antworten. Wird benutzt wenn Authentifizierung möglich ist, aber entweder fehlgeschlagen ist oder noch nicht unterstützt wird |
| 402 Payment Required | Reserviert für zukünftige Benutzung |
| 403 Forbidden | Anfrage war gültig, aber Server weigert sich zu antworten |
| 404 Not Found | Angefragte Seite konnte nicht gefunden werden, ist aber möglicherweise in der Zukunft verfügbar |
| 405 Method Not Allowed | Anfrage wird nicht von dieser Seite unterstützt |
| 406 Not Acceptable | Server kann nur eine Antwort generieren, die nicht vom Client unterstützt wird |
| 407 Proxy Authentication Required | Client muss sich zuerst beim proxy authentifizieren |
| 408 Request Timeout | Server hat TimeOut beim Warten auf die Anfrage |
| 409 Conflict | Anfrage konnte wegen eines Konflikts in der Anfrage nicht beantwortet werden |
| 410 Gone | Angefragte Seite ist nicht mehr verfügbar |
| 411 Length Required | Content-Length ist nicht definiert. Server akzeptiert Anfrage nicht ohne Content-Length |
| 412 Precondition Failed | Vorbedingung der Anfrage wird vom Server als falsch bewertet |
| 413 Request Entitiy Too Large | Anfrage-Größe ist zu groß |
| 414 Request-URI Too Long | Server akzeptiert Anfrage nicht, weil URL zu lang ist. Passiert wenn eine POST-Anfrage in eine GET-Anfrage konvertiert wird mit langen Anfrageinformationen |
| 415 Unsupported Media Type | Server akzeptiert Anfrage nicht, weil Medientyp nicht unterstützt wird |
| 416 Requested Range Not Satisfiable | Client hat einen Teil des Dokumentes angefragt, aber Server kann diesen Teil nicht liefern. |
| 417 Expectation Failed | Server kann die Anforderungen des Expect-Request-Header nicht erfüllen |
|  |  |
| **5xx: Server Error** |  |
| 500 Internal Server Error | Eine generische Fehlermeldung, die ausgegeben wird, wenn keine spezifische Meldung geeignet ist |
| 501 Not Implemented | Server erkennt Anfragemethode nicht, oder kann sie nicht erfüllen |
| 502 Bad Gateway | Server hat als Gateway oder Proxy fungiert und bekommt vom Upstream-Server eine ungültige Antwort |
| 503 Service Unavailable | Server ist derzeit nicht erreichbar |
| 504 Gateway Timeout | Server hat als Gateway oder Proxy fungiert und hat vom Upstream-Server nicht rechtzeitig eine Antwort erhalten |
| 505 HTTP Version Not Supported | Server unterstützt nicht das HTTP Protokoll das in der Anfrage verwendet wurde |
| 511 Netword Authentication Required | Client muss sich authentifizieren für den Netzwerkzugang |

HTTP Request Methods

HTTP wurde designed um die Kommunikation zwischen Servern und Clients zu ermöglichen.  
Arbeitet als Anfrage-Antwort Protokoll zwischen Server und Client.

HTTP Methods

* GET
* POST
* PUT
* HEAD
* DELETE
* PATCH
* OPTIONS

**GET-Methode**: Anfrage von Daten von Quelle

* können zwischengespeichert werden
* bleiben im Browserverlauf
* können mit Lesezeichen versehen werden
* sollten niemals verwendet werden, wenn es um sensible Daten geht
* haben Längenbeschränkungen
* werden nur verwendet, um Daten anzufordern, nicht um zu ändern

**POST-Methode**: Senden von Daten an Server, um Ressource zu kreieren oder zu aktualisieren

* werden nie zwischengespeichert
* verbleiben nicht im Browserverlauf
* können nicht mit einem Lesezeichen versehen werden
* haben keine Beschränkung der Datenlänge

**PUT-Methode:** Senden von Daten an Server, um Ressource zu kreieren oder zu aktualisieren

Unterschied zu POST: PUT-Anfragen führen immer zum gleichen Ergebnis, auch bei mehrfacher Ausführung. Bei POST wird dieselbe Ressource bei mehrfacher Ausführung mehrmals erstellt

**HEAD-Methode**: wie GET-Methode nur ohne Anfrage-Körper

z.B.: GET-Methode liefert eine Liste mit Usern, HEAD macht selbe Anfrage, liefert die Liste jedoch nicht zurück.

HEAD-Anfragen sind nützlich, um herauszufinden, ob die GET-Anfrage funktioniert

**DELETE-Methode**: löscht Ressource

**OPTIONS-Methode**: beschreibt die Kommunikations-Optionen für die Ziel-Ressource.

HTML-Formulare

Werden für Benutzer-Eingaben verwendet

<form>

<form>-Element ist ein Container für verschiedene Typen von input-Elementen (Text, Felder, Checkboxes, Submit-Buttons,…)

<input>

|  |  |
| --- | --- |
| **Typ** | **Beschreibung** |
| <input type=“text“> | Zeigt ein einzeiliges Input-Feld an |
| <input type=“radio“> | Button für Einfachauswahl |
| <input type=“checkbox“> | Keine- oder Mehrfachauswahl |
| <input type=“submit“> | Formular abschicken |
| <input type=“button“> | Klickbarer Button |

Textfelder

<input type=“text“>

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

<label> -Element

definiert ein Label für viele Formularelemente.

Vergrößert die klickbare Fläche von Buttons.

<label> -Tags sollte dem id-Attribut des <input>-Elements entsprechen, um sie miteinander zu verbinden.

Radio Buttons

<input type=“radio”> EINFACHAUSWAHL

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Checkboxes

<input type=“checkbox“> MEHRFACHAUSWAHL

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Submit Button

<input type=”submit”> gibt Formular an form-handler weiter.

Der form-handler ist typischerweise eine Datei auf dem Server mit einem Script um den Input zu verwerten.

Der form-handler wird im action-Attribut des Formulars angegeben.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Name-Attribut für <input>

Jedes Input-Feld muss einen Namen haben, um es zu benutzen.

Wenn der Name nicht vorhanden ist, wird das Input-Feld nicht geschickt.

Bootstrap

3\_4AKIF/TINF/NEMI/

PHP Tutorial

<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
echo "My first PHP script!";  
?>  
  
</body>  
</html>

PHP = “PHP: Hypertext Preprocessor”

Was kann PHP?

* kann dynamischen Seiteninhalt erstellen
* kann Dateien auf einem Server erstellen, öffnen, lesen, schreiben, löschen und schließen
* kann Datenformulare lesen
* kann Cookies senden und empfangen
* kann Daten in einer Datenbank hinzufügen, löschen und ändern
* kann User-Access kontrollieren
* kann Daten verschlüsseln

Warum PHP?

* läuft auf verschiedenen Plattformen (Windows, Linux, Unix, Mac OS X, etc.)
* ist mit fast allen Servern kompatibel (Apache, IIS, etc.)
* unterstützt viele Datenbanken
* ist gratis
* ist leicht zu lernen und läuft effizient auf der Serverseite

Syntax

<?php

//PHP-Code

?>

PHP ist nicht Case-sensitive

Kommentare wie in Java // /\* \*/

Variablen

mit $-Zeichen

$x = 5;

Arten von Variablen

* local
* global
* static

<?php

$x = 5; // globale Variable

locale Variablen in Funktionen

mit “global”-Keyword können globale Variablen überall verwendet werden

global $x;

bei „static“-Variablen können die Informationen der Variablen auch außerhalb einer Funktion abgerufen werden

Datentypen

* String
* Integer
* Float (inkludiert auch double)
* Boolean
* Array
* Object
* NULL
* Resource

String kann mit „“ oder ‚‘ deklariert werden

Array

$cars = array(„Volvo“,“BMW“,“Toyota”);

Object

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

\_\_construct() ist Konstruktor und wird bei jedem Neuen Objekt aufgerufen

Konstante

Keyword: define

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Array-Konstante:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

PHP String Operators

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Foreach-Schleife

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Funktionen

Keyword: function

Ein Bild, das Text enthält.

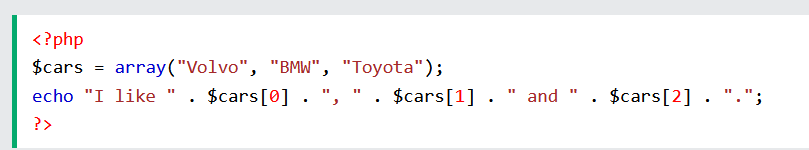
Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

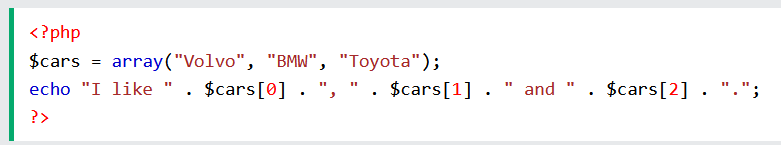
Automatisch generierte Beschreibung

//strict\_types=1 -> für strikte Datentyp Einhaltung (String kann nicht zu int zugeordnet werden)

Arrays



Indexierte Arrays



Ein Bild, das Text enthält.

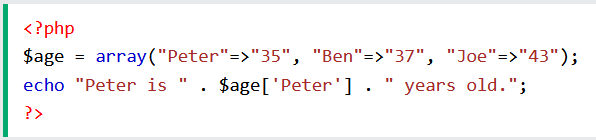
Automatisch generierte Beschreibung

Assoziative Arrays

wie HashMap

Ein Bild, das Text enthält.

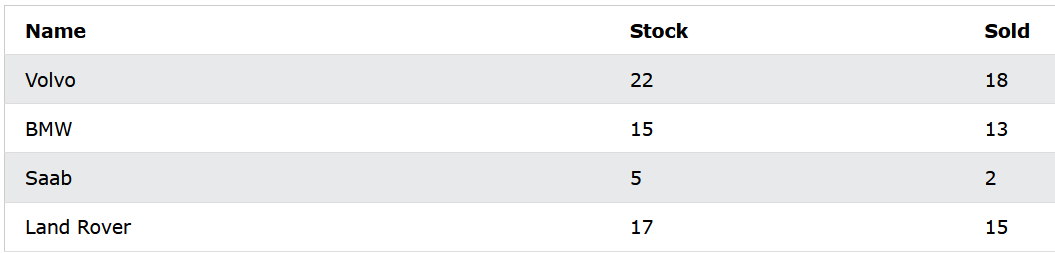
Automatisch generierte Beschreibung



Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Multidimensionale Arrays



Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Superglobals

sind immer und überall verfügbar

* $GLOBALS
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

$GLOBALS

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

$\_SERVER

hält Informationen über Header, Paths und Script Locations

einige $\_SERVER-Elemente:

Syntax: $\_SERVER[‚*Element*‘]

|  |  |
| --- | --- |
| Element | Beschreibung |
| PHP\_SELF | Name des aktuell ausgeführten Scripts |
| GATEWAY\_INTERFACE | Version des Common Gateway Interface (CGI) das der Server benutzt |
| SERVER\_ADDR | IP-Adresse des Host-Servers |
| SERVER\_NAME | Name des Host-Servers (z.B. [www.w3schools.com](http://www.w3schools.com)) |
| SERVER\_SOFTWARE | Server Identifications-String (z.B. Apache/2.2.24) |
| SERVER\_PROTOCOL | Name und Revision des Informations-Protokolls (z.B. HTTP/1.1) |
| REQUEST\_METHOD | Request Methode um die Seite zu öffnen (z.B. POST) |
| REQUEST\_TIME | Zeitstempel der Request Startzeit (z.B. 1377687496) |
| QUERY\_STRING | Abfragestring, wenn auf die Seite über Abfragestring zugegriffen wird |
| HTTP\_ACCEPT | Accept Header der aktuellen Anfrage |
| HTTP\_ACCEPT\_CHARSET | z.B. utf-8,ISO8859-1 |
| HTTP\_HOST | Host-Header für aktuelle Anfrage |
| HTTP\_REFERER | komplette URL der aktuellen Seite |
| HTTPS | wird Anfrage über HTTPS übertragen? |
| REMOTE\_ADDR | IP-Adresse von wo aus der Benutzer die Seite anschaut |
| REMONTE\_HOST | Host-Name des Benutzers |
| REMOTE\_PORT | Port des Benutzers |
| SCRIPT\_FILENAME | absoluter Pfad des aktuell ausgeführten Scripts |
| SERVER\_ADMIN | Pfad zur SERVER\_ADMIN-Direktive im Web-Server configuration file (z.B. [someone@w3school.com](mailto:someone@w3school.com)) |
| SERVER\_PORT | Kommunikations-Port-Nummer des Servers (z.B .80) |
| SERVER\_SIGNATURE | Server Version und host name |
| PATH\_TRANSLATED | Datei-System basierter Pfad |
| SCRIPT\_NAME | Pfad des aktuellen Scripts |
| SCRIPT\_URI | URI der aktuellen Seite |

$\_REQUEST

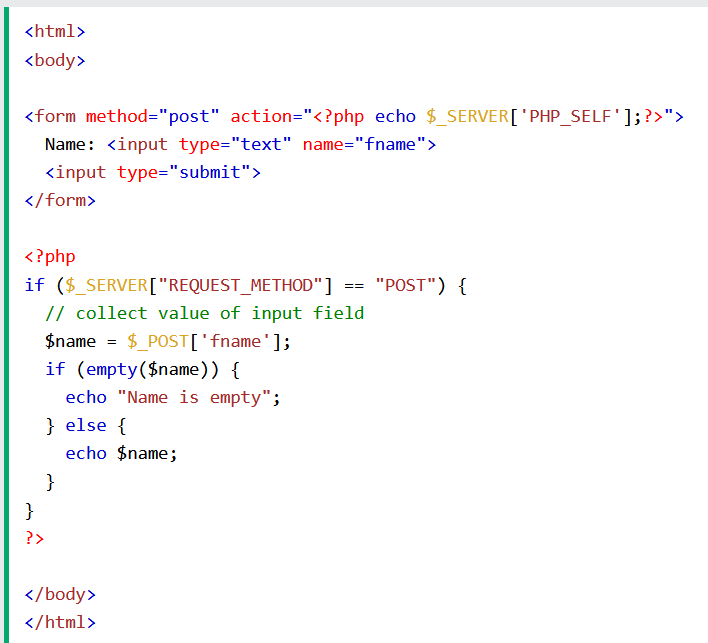
ist eine superglobale Variable, um Daten zu sammeln, nachdem das HTML-Formular submitted wurde

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

$\_POST

superglobale Variable, um ein Datenformular zu sammeln, nachdem das HTML-Formular submitted wurde.



$\_GET

superglobale Variable, um Daten aus einem Formular nach dem „submit“ auszulesen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

wenn ein User auf den Link klickt, werden die Parameter „subject“ und „web“ an „test\_get.php“ gesendet. mit $\_GET kann auf diese Parameter zugegriffen werden

PHP RegEx

Syntax

$exp = „/w3schools/i“;

i macht suche case-insensitive.

m multiline search

u aktiviert korrekte UTF-8 Patterns

|  |  |
| --- | --- |
| **Funktion** | **Beschreibung** |
| preg\_match() | gibt 1 zurück, wenn pattern gefunden wurde, 0 wenn nicht |
| preg\_match\_all() | gibt Anzahl der gefundenen Ergebnisse zurück, 0 wenn nichts gefunden wurde. |
| preg\_replace() | gibt String zurück, wo passendes Pattern mit anderem String ersetzt wurde. |
|  |  |
| **Patterns** |  |
| [abc] | findet einen char aus den Optionen in den Klammern |
| [^abc] | findet alles außer Optionen in den Klammern |
| [0-9] | findet character zwische 0 und 9 |
|  |  |
| **Metacharacters** |  |
| | | ODER (cat|dog|fish) |
| . | findet nur eine Instanz von irgendeinem character |
| ^ | sucht am Beginn des Strings (^Hello) |
| $ | sucht am Ende des Strings (World$) |
| \d | findet eine Zahl |
| \s | findet ein Leerzeichen |
| \b | sucht am Beginn (\bHELLO) oder am Ende des Wortes (WORLD\b) |
| \uxxxx | findet Unicode character mit hexadezimalem Wert xxxx |
|  |  |
| **Quantifiers** |  |
| n+ | mindestens 1 mal |
| n\* | 0 oder mehr |
| n? | 0 oder 1 mal |
| n{x} | X mal n |
| n{x,y} | X bis Y mal n |
| n{x,} | mindestens X mal n |

PHP Forms

superglobale $\_GET und $\_POST werden verwendet, um Daten zu sammeln

Form Handling

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

wenn User Formular ausgefüllt hat und auf „Submit“ geklickt hat wird Daten an „welcome.php“ mit HTTP POST gesendet.

welcome\_get.php:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Validierung fehlt noch

GET vs. POST

beide erzeugen einen array(key1 => value1, key2 => value2, …)

Keys sind die Namen der Form-Control, Values sind die eingegebenen Werte.

$\_GET ist ein Array von Variablen, das an das Script via URL Parametern

$\_POST ist ein Array von Variablen, das an das Script via HTTP POST Methode.

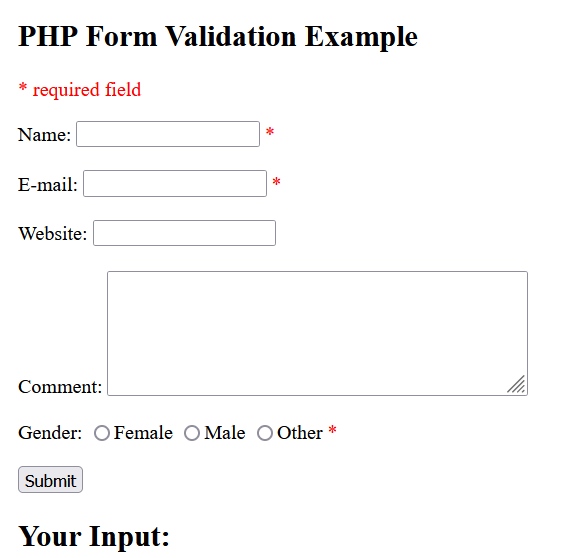
When to use GET?

alles sichtbar für jeden (alle Variabelnamen und Werte in der URL).

When to use POST?

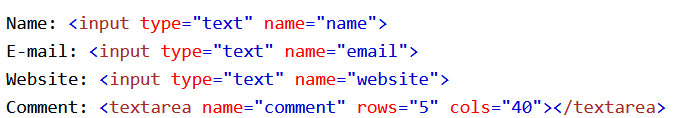
unsichtbar für andere (alle Namen und Werte sind im body des HTTP requests) und hat kein Längenlimit.

PHP Form Validation



|  |  |
| --- | --- |
| **Feld** | **Validation Rules** |
| Name | Required. + muss nur Buchstaben und Leerzeichen enthalten. |
| E-Mail | Required. + muss eine gültige E-Mail-Adresse enthalten. |
| Website | Optional. Wenn angegeben, muss eine gültige URL enthalten. |
| Comment | Optional. Multi-line input Feld |
| Gender | Required. Eines muss ausgewählt werden |

Text Fields



Radio Buttons

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

The Form Element

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

htmlspecialchars() ist eine Funktion die special Characters in HTML entities konvertiert -> das heißt dass es HTML characters wie < und > mit &lt; und &gt; ersetzt. Das verhindert, dass Andere HTML- oder Javascript code übergeben geben.

Komplettes Beispiel

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung