Surfen im Web, technisch betrachtet

V 4.2, 12.1.2022

DI Herwig Diernegger

Inhaltsverzeichnis

1 Client-Server-Kommunikation im WWW 3
2 Dateien am Webserver zur Auslieferung hinterlegen 4
2.1 Konfiguration des Webserver-Dienstes Apache 4
2.2 Apache-HTML-Übung 4
2.3 Übungen 4
2.3.1 Auf welchen Wert ist auf meinem XAMPP-Webserver die Variable DocumentRoot gesetzt? 4
2.3.2 Beispiel 1 5
2.3.3 Beispiel 2 5
2.3.4 Beispiel 3 5
2.3.5 Beispiel 4 6
2.3.6 Beispiel 5 6
2.3.7 Beispiel 6 6
2.3.8 Beispiel 7 6
3 Ablauf der HTTP-Kommunikation anhand einer HTML-Datei 8
3.1 Dateiendungen sind wichtig 8
3.2 Spezialthema index.html und index.php 8
4 HTML versus PHP 10
5 Ablauf der HTTP-Kommunikation bei einer PHP-Datei 11
6 PHP 12
7 Das Datenbank-System 14
7.1 mysql-Client oder mysql-Server? 14
7.2 Übung: msql.exe starten 15
8 HTTP-Kommunikation mit Apache, PHP und MySQL 16
8.1 Zugriff auf die Datenbank mit der mysqli-PHP-Bibliothek 17

1 Client-Server-Kommunikation im WWW

- Jede Anfrage eines Clients fragt nach einer konkreten Datei, die am Webserver hinterlegt sein muss.

Bsp.: http://localhost/index.php

Wo am Webserver muss die Datei index.php abgelegt werden, damit obiger Request funktioniert? Dieser Frage gehen wir auf den folgenden Seiten nach.

2 Dateien am Webserver zur Auslieferung hinterlegen

2.1 Konfiguration des Webserver-Dienstes Apache

Einen Dienst konfigurieren, damit meinen wir, dass wir Einstellungen vornehmen, die das Verhalten des Dienstes ändern. Im Fall von XAMPP für Windows sind alle Einstellungen in der Datei httpd.conf im Verzeichnis C:\xampp\apache\conf\ vorzunehmen.

DocumentRoot heißt die Variable im Konfig-File httpd.conf des Apache-Webservers, die festlegt, wo die Files gespeichert werden müssen, damit der Webserver diese ausliefern kann.

```
DocumentRoot "C:\xampp\htdocs"
```

2.2 Apache-HTML-Übung

1. Erstelle mit Notepad++ das File hallo.html, und speichere es in deinem DocumentRoot ab!(also meist hier: C:\xampp\htdocs)

2. Inhalt des Files hallo.html:

```
<h1>Hallo, Welt!</h1>
```

3. Speichere die Datei!

4. Wie arbeitet der Webserver? Es sucht die angeforderte Datei in seinem DocumentRoot!

Daraus folgt: Wenn du im Browser

```
http://localhost/hallo.html
```

eingibst, so liefert der Webserver den Inhalt von hallo.html an den Browser aus, und der Browser zeigt den Inhalt an!!!!!

Somit hast du deine erste Webseite am eigenen Webserver erstellt und abgerufen!!!!

2.3 Übungen

2.3.1 Auf welchen Wert ist auf meinem XAMPP-Webserver die Variable DocumentRoot gesetzt?

Die Variable DocumentRoot ist Teil der Konfiguration des Webservers Apache.

Bei XAMPP/Windows findest du diese Konfiguration hier: C:\xampp\apache\conf.

1. Erstelle eine Sicherungskopie der Datei (Strg+C, Strg+V)
2. Öffne httpd.conf mit einem Texteditor
3. Suche nach 'DocumentRoot' (case insensitive):

Du findest die Lösung:

```
DocumentRoot C:/xampp/htdocs
```

Anmerkungen:

1. Am macOS: heißt die Datei auch httpd.conf, und liegt unter Programme/xamppfiles/apache2/conf

2. Das 'd' bei 'httpd' steht für 'daemon' (anderes Wort für 'Server')

3. Konfigurationsfile: Ein Konfigurationsfile ist eine Datei, in der festgeschrieben ist, wie sich der zugehörige Server-Prozess verhalten soll.

2.3.2 Beispiel 1

Gehen wir nach c:/xampp/htdocs und erzeugen dort eine Datei first.html mit dem Inhalt:

```
<h1>meine erste HTML-Datei, mit Apache ausgeliefert!</h1>
```

Wie lautet der korrekte HTTP-Request?

Antwort: http://localhost/first.html

2.3.3 Beispiel 2

Legen wir eine Datei 'second.html' im Unter-Verzeichnis "help" ab; Wie lautet der korrekte HTTP-Request?

A: http://localhost/help/second.html

2.3.4 Beispiel 3

Der Aufruf

```
http://localhost/aaa/bbb/test.html
```

soll funktionieren - wo muss die Datei test.html abgelegt werden - gib den absoluten Pfad

an: Lösung für XAMPP und Windows: C:/xaamp/htdocs/aaa/bbb/test.html

2.3.5 Beispiel 4

Im Browser wird eingegeben:

```
http://localhost/schule/3/imcm/aufgabe1.html
```

Wo im Filesystem des Webservers muss die Datei gespeichert sein, damit der Aufruf klappt? Warum?
Teste auf deinem Gerät!!!

2.3.6 Beispiel 5

Im Browser wird eingegeben:

```
http://localhost/schule/3/aufgabe2.html
```

Wo im Filesystem des Webservers muss die Datei gespeichert sein,

damit der Aufruf klappt? Warum? Teste auf deinem Gerät!!!

2.3.7 Beispiel 6

Im Browser wird eingegeben:

```
http://localhost/info.php
```

Inhalt der Datei info.php:

Surfen im Web, technisch betrachtet

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

Wo im Filesystem des Webservers muss die Datei gespeichert sein, damit der Aufruf klappt? Warum? Teste auf deinem Gerät!!! Screenshot!!

2.3.8 Beispiel 7

Kopiere die Datei info.php nach info.html, rufe sie auf und dokumentiere die Ergebnisse mit einem Screenshot!

Warum kommt eine weiße Seite? Antwort liefert eventuell:

Rechte Maustaste – Seitenquelltext anzeigen.

Antwort: PHP kommt am Browser an, weil falsche Datei-Endung!

PHP-Dateien müssen immer die Endung .php haben.

3 Ablauf der HTTP-Kommunikation anhand einer HTML-Datei

Ablauf der HTTP-Kommunikation anhand des Requests nach der Datei hallo.html:

1. Im Browser wird eingegeben: "http://localhost/hallo.html" und "Enter" gedrückt. Der Browser erzeugt dann einen HTTP-Request, also eine Anfrage an den Web-Server 'localhost'.

2. Der Web-Server empfängt den Request. Er liest die angeforderte Datei von seinem Filesystem.

3. Da es sich um eine Datei mit der Endung ".html" handelt, übernimmt er den Inhalt der Datei sofort in den HTTP-Body der HTTP-Antwort.

4. Er ergänzt den HTTP-Header der HTTP-Antwort um eine Anzahl von Variablen. (Welche Variablen das sind, wird auch im HTTP-Protokoll definiert.) Sichtbar gemacht werden können diese Variablen mit dem Netzwerkanalyse-Tool des Browsers.

5. Er sendet die Antwort, also den HTTP-Response.

6. Der Browser empfängt die Daten, verarbeitet die Daten und stellt die HTML-Seite dar.

3.1 Dateiendungen sind wichtig

Im WWW muss jede Datei die richtige Endung haben.

HTML-Dateien:	.html
PHP-Dateien:	.php
Javascript-Dateien:	.js
PDF-Dateien:	.pdf
JPEG-Dateien:	.jpg

usw.

3.2 Spezialthema index.html und index.php

Wenn der Benutzer im HTTP-Request keinen Dateinamen angibt (http://www.sz-ybbs.ac.at), dann ist jeder Webserver meist so konfiguriert, dass er nach den Dateien index.php,index.html sucht, in dieser Reihenfolge, und die erste, die er findet, verwendet er. (wenn der Webserver PHP kann; anderenfalls sucht er nur nach index.html)

Übung: Prüfe, ob dein System auch so arbeitet.

4 HTML versus PHP

Hyper Text Markup Language – Hyper Text Auszeichnungs Sprache!!!

Mit HTML kann Text ausgezeichnet (auch: formatiert) werden: p, h1, h2, ….

Das Wort "Hyper" bezieht sich auf die Möglichkeit, im HTML auch Links zu setzen.

HTML ist keine Programmiersprache! HTML kann zum Beispiel nicht:

a) rechnen: Bsp.: Der Benutzer soll sein Geburtsdatum eingeben (das geht mit einem HTML-Formular), es soll dann berechnet werden, wie viele Tage er auf der Welt ist. (das geht nur mit einer Programmiersprache wie zB PHP)

b) Datenbanken abfragen: Zur Beantwortung eines HTTP-Requests muss eine Datenbank abgefragt werden. Dazu wird ebenfalls eine Programmiersprache benötigt.

c) Einkaufen im Web. Die vielen Aufgaben, die beim Einkaufen im Web notwendig sind, können nur mit einer Programmiersprache umgesetzt werden, HTML alleine reicht nicht.

d) Web-Dienste wie Google Suche, Google Docs, Facebook, Instagramm usw. brauchen eine Programmiersprache am Webserver, HTML reicht nicht aus.

5 Ablauf der HTTP-Kommunikation bei einer PHP-Datei

Ablauf der HTTP-Kommunikation anhand des Abrufes der Datei info.php:

1. Im Browser wird eingegeben: "http://localhost/info.php" und "Enter" gedrückt. Der Browser erzeugt dann einen HTTP-Request, also eine Anfrage an den Web-Server 'localhost'.

2. Der Web-Server empfängt den Request. Er liest die angeforderte Datei von seinem Filesystem.

3. Da es sich um eine Datei mit der Endung ".php" handelt, übergibt der Web-Server den Inhalt der Datei an an den PHP-Interpreter. Der PHP-Interpreter arbeitet die Datei ab, er "interpretiert" also alle PHP-Statements. Das Ergebnis der Interpretation (meist ist es HTML) wird zurück an den Web-Server gegeben. Nun geht es weiter wie bei einer HTML-Datei auch:

4. Der Web-Server ergänzt den HTTP-Header der HTTP-Antwort um eine Anzahl von Variablen. (Welche Variablen das sind, wird auch im HTTP-Protokoll definiert.) Sichtbar gemacht werden können diese Variablen mit dem Netzwerkanalyse-Tool des Browsers.

5. Er sendet die Antwort, also den HTTP-Response.

6. Der Browser empfängt die Daten, verarbeitet die Daten und stellt die HTML-Seite dar.

6 PHP

PHP ist eine Programmiersprache, die meist in Verbindung mit dem Apache-Webserver in Erscheinung tritt.

1. Ein erstes Skript: info.php

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

Speichere die Datei im DocumentRoot deines Webservers; Wie lautet der Aufruf im Browser?

http://localhost/info.php

2. Jedes PHP-Programm muss mit <?php eingeleitet werden

und mit ?> beendet werden. Siehe info.php!

Davor und danach kann (soll aber nicht:-) HTML stehen!

3. phpinfo(); ist der Aufruf der vorgefertigten gleichnamigen PHP-Funktion.

Diese Funktion erzeugt die HTML-Zeilen, die im Browser dann die vielen grau-blauen Tabellen erzeugen.

4. Das HTML, das vom Browser verarbeitet wird, kann ich mir ansehen mit:

rechte-Maus→ Seitenquelltext anzeigen

Bsp2:

Schreibe eine Datei info2.php und speichere sie ebenfalls im DocumentRoot:

```
<h1>TEST1</h1>
<?php
 echo "<h1>TEST2</h1>";
?>
```

Außerhalb von <?php und ?> kann HTML geschrieben werden; innerhalb von <?php und ?> muss PHP geschrieben werden; ein HTML-String muss in PHP mit echo ausgegeben werden, auch die Anführungszeichen sind notwendig…….

Bsp. 3:

Nenne die Datei aus Beispiel 2 um nach info2.html, tausche also die Dateiendung aus – schau mal, was passiert, hehe :-)

Weil die Datei-Endung nicht .php ist, wird am Web-Server das PHP auch nicht interpretiert, also nicht abgearbeitet. Das PHP "taucht" somit am Browser auf, dieser weiß aber nicht, was er damit anfangen soll und zeigt an, so gut es geht…:-)

7 Das Datenbank-System

Das Datenbank-System arbeitet auch nach dem Client-Server-Prinzip. Speziell ist, dass jeder Client sich mit User-Name und Passwort authentifizieren muss.

7.1 mysql-Client oder mysql-Server?

Frage: Wie startet man (unter Windows) den Command-Line-Client mysql?

Antwort:

1. Windows+R → CMD

2. cd c:\xampp\mysql\bin

3. .\mysql.exe

Anmerkung: Der Client heißt sehr ähnlich wie der mySQL-Datenbank-Server.

Nicht verwechseln!!!!

7.2 Übung: msql.exe starten

1. Schwarze CMD starten am Windows:

2. cd c:\xampp\

3. cd mysql

3.1 cd bin

4. dir

Scheint eine Datei "mysql.exe" auf? Wenn ja, rufen wir diese auf:

5. .\mysql.exe

Der Punkt steht für "aktuelles Verzeichnis", das brauchen wir, weil wir dem System sagen müssen: Hier in diesem Verzeichnis nimm die Datei "mysql.exe" und für sie aus.

6. Jetzt kommt vom DB-Server die Meldung: "Access denied", weil Username und Passwort nicht richtig sind: Soweit, so gut:-)

8 HTTP-Kommunikation mit Apache, PHP und MySQL

Der Webserver ist über das Internet (genauer WWW) erreichbar, der Datenbankserver (aus Sicherheitsgründen) nicht. Ebenfalls aus Sicherheitsgründen ist der Zugriff auf den Datenbankserver mit Username und Passwort geschützt.

XAMPP-MySQL-Server: Username: "root", leeres Passwort: ""

Somit greifen wir (in der Praxis oft über das Internet) auf den Webserver zu, der Webserver wiederum mit PHP auf den DB-Server. Meist liegen Webserver und DB-Server am selben Rechner, wie bei XAMPP auch.

Beschreibung zur Skizze oben:

1. Der Benutzer setzt im Browser einen HTTP-Request ab:

```
http://localhost/database/getData.php
```

2. Der Webserver empfängt diesen Request. Sei seine Variable DocumentRoot auf den Wert

C:\htdocs gesetzt. Somit versucht der Webserver, die Datei C:\htdocs\database\getData.php

von seiner Festplatte zu laden.

3. Weil die Dateiendung .php ist, übergibt der Webserver den Inhalt der Datei an den PHP-Interpreter. Dieser Interpreter arbeitet den PHP-Code ab und gibt das Ergebnis (meist wird HTML erzeugt) an den Webserver zurück.

3.1. Wenn im PHP-File eine Datenbank-Abfrage programmiert wurde (mit zB. mysqli), dann wird mit diesem mysqli-Datenbank-Client eine Abfrage gegen den mysql-Server durchgeführt, das Abfrage-Ergebnis entgegen genommen und durch PHP weiterverarbeitet. (Siehe dazu die zugehörige Skizze).

4. Der Webserver baut die HTTP-Antwort zusammen: In den HTTP-Body der Antwort trägt er die vom PHP-Interpreter erhaltenen Daten ein.

5. Die HTTP-Antwort wird an den Browser gesendet.

6. Der Browser zeigt die Webseite an.

8.1 Zugriff auf die Datenbank mit der mysqli-PHP-Bibliothek

Siehe: https://www.php.net/manual/en/mysqli.quickstart.dual-interface.php

Ein erster Zugriff auf den MySQL-Server:

Im nachfolgenden Code müssen die Größen "user", "password" und "database" angepasst werden an unsere Verhältnisse.

Die Datenbank "database" muss voher mit zB phpmyadmin erstellt werden.

a) Prozeduraler Zugriff:

```
<?php
 // Verbindung zum Datenbankserver herstellen:
 $mysqli = mysqli_connect("localhost","user","password","database");
 // Abfrage auf die virtuelle Tabelle DUAL ausführen:
 $result=mysqli_query($mysqli,"SELECT 'Great job!!!' AS _msg FROM DUAL");
 // Ergebnis der Abfrage auslesen:
```

```
 $row = mysqli_fetch_assoc($result);
 // Ergebnis ausgeben:
 echo $row['_msg'];
?>
```

b) Objektorientierter Ansatz:

```
<?php
 // Verbindung zum Datenbankserver herstellen:
 $mysqli = new mysqli("localhost", "user", "password", "database");
 // Abfrage auf die virtuelle Tabelle DUAL ausführen:
 $result = $mysqli->query("SELECT 'Great job!!!' AS _msg FROM DUAL");
 // Ergebnis der Abfrage auslesen:
 $row = $result→fetch_assoc();4
 // Ergebnis ausgeben:
 echo $row['_msg'];
?>
```

Achtung:

Obiger Code fragt die immer vorhandene virtuelle Tabelle "DUAL" ab; somit brauchen wir an dieser Stelle noch keine anderen Tabellen in der Datenbank.

Mit dem Werkzeug phpmyadmin (Teil von XAMPP) können nun in der Datenbank "database" Tabellen erstellt werden.

Nennen wir die Tabelle "friends". Wir wollen in dieser Tabelle den Namen unserer Freunde und deren E-Mail-Adresse speichern. Somit erzeugen wir mit phpmyadmin (oder mit MS-Workbench) die Tabelle "friends" mit den Spalten "name" und "email".

Daten-Typ für name und email sei varchar[30]

Ebenfalls mit diesem Tool können dann in diese Tabelle Datensätze eingetragen werden.

"valentin", "vali@gmail.com"

"sara", "sara@sz-ybbs.ac.at"

8.1.1 Source-Code

```
<?php
	// den in PHP vorhandenen mySQL-Client verwenden:
	// Verbindung zum mysql-Server herstellen; notwendige Angaben:
	// Name des DB-Servers: localhost
	// Username zum Anmelden: root
	// Passwort des Users: kein Passwort
	// Name der Datenbank am DB-Server, die verwendet werden soll: first
	$mysqli = new mysqli("localhost", "root", "", "first");

	// Eine Datenabfrage wird formuliert:
	// Gib mir den Namen und die E-Mail-Adresse von allen Datensätzen,
	// die in der Tabelle friends vorhanden sind:
	$sql = "SELECT `name`,`mail` FROM `friends`";

	// Die Abfrage nun durchführen:
	$results = $mysqli->query($sql);

	// die HTML-Tabelle für die Ausgabe der Daten aufbauen:
	// Ausgabe des Tabellenkopfes:
	echo "<table border=1 cellpadding=12>";

	echo "<tr>";
	echo "<td>Name</td>";
	echo "<td>E-Mail</td>";
	echo "</tr>";

	// Schleife über alle Datensätze der Abfrage:
	// solange (while) ein Datensatz vorhanden ist im
	// Abfrage-Ergebnis ($results):
	while($rowitem = mysqli_fetch_array($results))
	// jeder Datensatz ist durch $rowitem (Zeilenwert) erreichbar:
	{
		echo "<tr>";
		// Die verwendeten Namen in den eckigen Klammern müssen
		// den Namen in der Datenbank entsprechen:
		echo "<td>" . $rowitem['name'] . "</td>";
		echo "<td>" . $rowitem['mail'] . "</td>";
		echo "</tr>";

	}
	echo "</table>";
?>
```