FH University of Applied Sciences TECHNIKUM WIEN

Backend Basics

Serverseitiges Entwickeln mit PhP und MySql

FH University of Applied Sciences TECHNIKUM WIEN

Über mich

Helmuth Lammer

- externer Lektor
- Web Developer seit fast 20 Jahren
- Backend und Frontend Development ist mein daily business
- Unternehmer, Berater

Agenda

- Rahmenbedingungen der Lehrveranstaltung
- Ziele der Lehrveranstaltung
- Grundkonzept LAMP (**L**inux, **A**pache, **M**ySql, **P**hP)
- gemeinsames Einrichten der Arbeitsumgebung
- Planen der ersten Anwendung
- Besprechen der Hausübung

Rahmenbedingungen

- Frontend Basics
  - ECTS: 4,5
  - 7 Termine laut CiS Kalender
  - 80% Anwesenheitspflicht

- die Note setzt sich zusammen
  - 60 Punkte in 5 Hausübungen
  - 12 Punkte Theoretischer Test (Hausübungen müssen über 30 Punkte sein)

- Notenschlüssel

  100% = 72 Punkte

  <50% Nicht genügend
  >=50% und <63% Genügend
  >=63% und <75% Befriedigend
  >=75% und <88% Gut
  >=88% Sehr Gut

- **Zweitantritt**: mündliche Prüfung zu einer HÜ oder einem der Bsp. aus der LV

Rahmenbedingungen

- Modus und Hausübungen
  - theoretischer Vortrag
  - praktische Übung, gemeinsam und selbstständig
  - Hausübungen zum Wiederholen und Vertiefen (!! hier liegt der Schwerpunkt !!)
  - übergreifende Hausübung mit der LV Backend-Basics
  - Test: Theoretische Fragen

Ziele der Lehrveranstaltung

Die Studierenden sollen ohne Vorlagen eine Web-Anwendung mit PhP in objektorientierter und wartbarer Weise gegen eine definierte Schnittstelle programmieren können, der eine verständliche schriftliche Planung voraus geht.

Nach erfolgreichem Abschluss sind die Studierenden in der Lage, ...

- theoretische Kenntnisse zur Erstellung von serverseitigen Applikationen auf Basis von PHP wiederzugeben
- serverseitige Applikationen auf Basis von PHP zu implementieren
- theoretische Kenntnisse zum HTTP-Protokoll, sowie zum Zusammenspiel zwischen AJAX und PHP wiederzugeben
- Datenbanken, sowie Datenbankverbindungen zu erstellen
- RESTful Web Services zu implementieren und zu verwenden
- Datenaustausch via JSON zu implementieren

"Als es noch keine Computer gab, war das Programmieren noch relativ einfach."

Edsger W. Dijkstra

Keinem Fällt in den Schoss Programmieren zu können,
es ist für jeden sehr viel Übung.

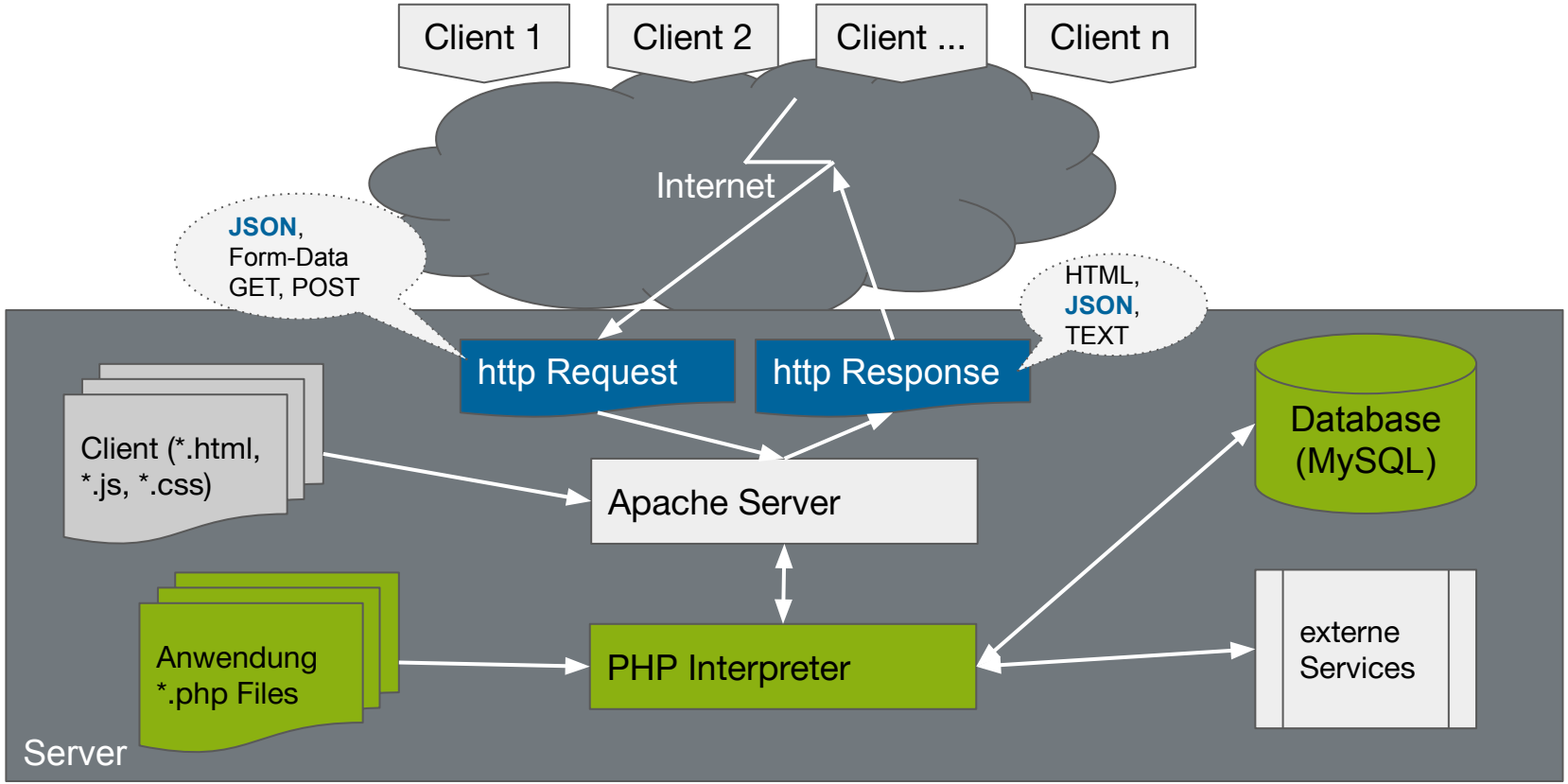
Warum serverseitige Anwendungen?

- Schnittstelle zu gemeinsam genutzter Datenbasis (Datenbank)
- Interaktion zwischen Clients ermöglichen
- Service für viele Clients verfügbar machen
- klare Trennung von Datenverarbeitung und User-Experience / GUI

Wie funktioniert eine Web-Anwendung?

- Webserver wird über IP und Domain gefunden
- ein Server wartet auf eingehende Anfragen mittels URL (Unified Request Language)
- Kommunikation im Internet über http(s) Requests mit Parametern
- eine Anfrage führt zum Auslösen einer Aktion: z.b. PHP Interpreter wird gestartet
- Der Server Antwortet mit einer Ausgabe oder einem Fehler (http Error Codes)

LAMP-Architektur: Linux, Apache, MySql, PhP



Apache Server

APACHE
HTTP SERVER PROJEC

FH University of Applied Sciences TECHNIKUM WIEN

- startete 1995 als OpenSource Projekt
- seit 1996 am meisten verbreiteter Webserver im Internet
- läuft unter Unix/Linux und Windows
- Verarbeitet und routet http und https Anfragen
- aktuelle Version: Apache httpd 2.4.29 Released 2017-10-23

PhP: hypertext pre processor

- Rasmus Lerdorf erfindet 1994/95 die Skriptsprache zur Vorverarbeitung von HTML Output (Templating Engine)
- inzwischen eine objektorientierte, vollumfängliche Programmiersprache
- wird von Apache interpretiert (nicht kompilierte Sprache)
- aktuelle Version: PHP 8.0
- berühmtes Beispiel einer PhP Anwendung ist facebook

PhP Anwendungen

- stellt Service zur Verfügung
- implementiert die datenverarbeitende Logik
- entweder via html-Formular oder als RESTFul Service
- klassische Verarbeitungsschritte eines Requests durch PhP
  - 1: Daten entgegen nehmen (GET / POST)
  - 2: Daten validieren (Inhalt, Datentyp, Sicherheit)
  - 3: Daten verarbeiten (lesen und schreiben auf z.b einer Datenbank)
  - 4: Ausgabe erzeugen (Response)

MySQL

- 1959 in Schweden als OpenSource software der Firma MySQL AB ins Leben gerufen
  - My ist der Name der Tochter des Firmengründers
  - **S**tructured **Q**uery **L**anguage
- Relationale Datenbank
  - Sammlung von Tabellen, wobei Zeilen mit Zeilen anderer Tabellen verbunden sein können
- phpmyadmin - PHP Service zum Arbeiten auf der DB

XAMPP, MAMPP

- Download Link: https://www.apachefriends.org/de/index.html

- Schritt für Schritt Anleitung
  https://goo.gl/yM4xq7 oder http://goo.gl/8JUC7p
  - xampp herunterladen
  - xampp aufsetzen
  - php testen
  - phpmyadmin aufrufen
  - mysql testen

- gemeinsames Arbeiten …

IDE Einrichten: Netbeans

- Integrated Development Environment
- vereinfacht das Entwickeln
- stellt Projekte von Beginn an auf professionelle Beine
- erleichtert das Lernen einer Programmiersprache
- https://netbeans.org/features/php/

Debugger einrichten: xdebug

- Debuggen: ist das Analysieren der Runtime z.b. bei der Fehlersuche
- wer sich mit dem Debugger nicht auskennt, vertut seine Zeit mit unsinnigen Annahmen und ohne Fakten
- XDebug Tutorial: https://gist.github.com/odan/1abe76d373a9cbb15bed
- XDebug und Netbeans: https://postimg.org/image/8tfvemy7l/

Qualität

liegt in der Software Entwicklung zum größten Teil

in der Wartbarkeit und Erweiterbarkeit von Codes

und der Verwendung von Standards

15 Minuten Pause

Programmieren mit PhP

- Grundlegende Syntax
- Datentypen, Arrays, Objekte
- Schleifen und Fallunterscheidungen
- Debugging mit PhP
- Objekt Orientiertes Programmieren

PHP Syntax

- jedes file startet mit dem Skript Tag `<?php`
- Variablen durch Dollarzeichen kennzeichnen `$variable`
- Zeilen enden mit einem Strichpunkt `;`
- Groß/Klein-Schreibung ist relevant
- Kommentare im Sourcecode
  - `// einzeiliger Kommentar`
  - `# Alternative für einzeiligen Kommentar`
  - `/* mehrzeiliger Kommentar */`

Operatoren in PhP

- Variablen Zuweisung / Allokation: `=, +=, -=, *=, /=, .=`
- Inkrementor / Dekrementor `++, --`
- Logisches UND: `&&`
- Logisches ODER: `||`
- Vergleichende Operatoren `==, ===, !=, >, >=, <, <=`
- Modulo `%`

Beispiel

```
<?php
      $variable = “Some String”;
      $maxCount = 40;

      for($i=0; $i<$maxCount; $i++){

            if($i % 4 == 0) {
                  echo $i . ”. “ . $variable . “\n”;
            }
      }
```

Datentypen

- PhP hat keinen Typezwang bei der Allokation von Variablen
- Numerische Typen
  - Integer – Ganzzahlen ohne Nachkommastellen
  - Float – Zahlen mit Nachkommastellen. Die Genauigkeit hängt von der php.ini Einstellung ab
- String – Zeichenketten
- Array – Liste
- Object – Objekte
- Boolean – true / false
- null – Eine Variable mit dem Wert null existiert, besitzt aber keinen Wert.

Arbeiten mit Strings

```
<?php
      $ourText = “Hello”;

      $ourText .= ‘ ‘; //concatenates a “ “ at the end of $ourText
      $ourLongerText = $ourText . “World!”;

      echo $ourLongText; //prints “Hello World!”

      $someMallformedText = “     I am a Text with too much white space      “;

      echo trim($someMallformedText); //will print “I am a Text with too much white space”

      //you can handle a string like an array/list
      echo $ourText[1]; //will print the “e”

      $lineBreak = “\n”;
      echo str_replace(“ “, $lineBreak, $ourLongText); //prints “Hello\nWorld”, while \n will create
                                                       //a new line
```

Arbeiten mit Arrays

```php
<?php
	$firstArray = array(); //allocating an empty array in php
	$secondArray = array("Susi", "Alex", "Peter", "Melanie"); //allocation an array with values

	$firstArray[] = 1024; //adding a value to an array - last item
	$firstArray[] = 2048; //adding another value - after the above

	array_push($secondArray, "Jenny"); //pushing a value to the end of an array
	//besides array_pop(), array_shift(), array_unshift()

	$secondArray[2] = "Petra"; //replaces the Peter with Petra

	echo $firstArray[0]; //will print out "1024", array indices start at zero

	$secondArrayLength = count( $secondArray ); //get's the number of elements of an array
	echo $secondArray[$secondArrayLength - 1]; //prints out "Jenny", the last item

	echo implode(" - ", $secondArray); //prints "Susi - Alex - Petra - Melanie - Jenny"
```

Arbeiten mit assoziativen Arrays

```php
<?php
	$assocArray = array(
		“first” => “some value”,
		“second” => “another value”
	);
	$assocArray[“third”] = “the last value so far”;

	echo $assocArray[“first”]; //prints out “some value”

	$twoDimensionalArray = array(); //allocating an empty array

	$twoDimensionalArray[] = array(“name” => “Thomas”, “birthYear” => 1990);
	$twoDimensionalArray[] = array(“name” => “Franziska”, “birthYear” => 1992);
	$twoDimensionalArray[] = array(“name” => “Martina”, “birthYear” => 1980);

	print_r($twoDimensionalArray); //prints the whole array with its structure
```

Arbeiten mit assoziativen Arrays

```
//Output

     array(
          array(
               “name” => “Thomas”
               “birthYear” => 1990
          )
          array(
               “name” => “Franziska”
               “birthYear” => 1992
          )
          array(
               “name” => “Martina”
               “birthYear” => 1980
          )
     )
```

Arbeiten mit standard Objekten

```php
<?php
	$someObject = new \stdClass;
	$someObject->name = “Lukas”;
	$someObject->birthYear = 1995;

	//much better is to use a stupid class object (stupid because it has no methods)
	class Person{
		public $name;
		public $birthYear;
	}

	$luke = new Person();
	$luke->name = “Lukas”;
	$luke->birthYear = 1995;

	echo $luke->name; // Will print “Lukas”
```

Fallunterscheidungen und Schleifen

Arbeiten mit Fallunterscheidungen

```
<?php
      $valueOne = true; $valueTwo = false;

      if( $valueOne && !$valueTwo){
            echo “Hi Students!”;
      } else {
            echo “Bye Students!”;
      }

      switch($someValue){
            case true:
                  //do something
                  break;
            case false:
                  //do something else
                  break;
            default:
                  echo “the given Value was null.”
                  break;
      }
```

Arbeiten mit der for und foreach Schleife

```php
<?php
	//if it is clear when our loop will start and end

	$vehicles = array(“Auto”, “Bus”, “Motorad”, “Fahrrad”);

	for( $i=0; $i<count($vehicles); $i++){ //we could also iterate a specific number of times
		echo $i . “. “.$vehicles[$i];
	}

	//alternatively
	foreach($vehicles as $vehicleItem){
		echo $vehicleItem; //we don’t have a loop-variable for counting
	}
```

Arbeiten mit der while Schleife

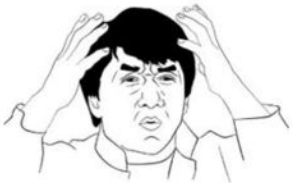
```
<?php
      //if the amount of iterations is dependent on the input

      $text = “The quick brown fox jumps over the lazy dog!”;

      //let’s count the words until the word “fox” appears
      $words = explode(“ “, $text);
      $wordCount = 0;
      while($words[$wordCount] != “fox” && $wordCount < count($words) ){
            $wordCount++;
      }

      $wordCount ++; //because we started the index with zero
```

FH University of Applied Sciences TECHNIKUM WIEN

Objektorientiertes Programmieren

Programmierparadigma OOP

- um ein Modell der Wirklichkeit in Software abbilden zu können, entwirft man Software als Zusammenspiel von Objekten
- Objekte haben Eigenschaften und Fähigkeiten (Methoden)
- eine Klasse beschreibt die Eigenschaften und Methoden von Objekten
- Objekte können andere Objekte aggregieren oder assoziieren, wodurch die Orchestrierung von Objekten möglich wird.



Assoziation vs. Aggregation

- **Assoziation** ist ein loses Verhältnis zwischen zwei Objekten, die auf der gleichen (Abstraktions-) Ebene stehen.

- **Aggregation,** wenn ein Objekt das zweite als Eigenschaft besitzt

Andere wichtige Begriffe

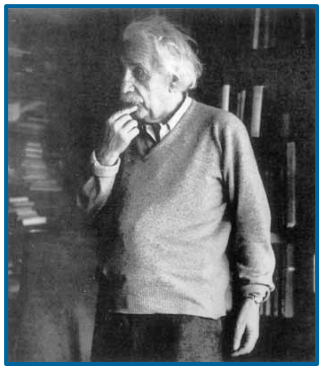
- Interface (interface - implements)
  - Methodensignaturen werden vereinbart und müssen von der implementierenden Klasse mit Funktionalität gefüllt werden
- Accesslevel
  - private, public, protected
  - final, static
- Vererbung und Abstraktion (abstract - extends)
  - abstrakte Klassen können einige Methoden implementieren und andere durch die erbende Klasse verlangen befüllt zu werden - als Charakteristikum (Bsp. Obst)
  - Polymorphie (Vielgestaltigkeit) bedeutet, dass eine Methode immer die gleiche Signatur aber nicht die gleiche Funktionalität bereitstellt (mehr oder anders)

Syntax in PhP

```
<?php
	class Department{
		private $employeesList;
		private $phoneExtension;

		public function __construct(){
			$this->initEmployeesList();
		}

		private function initEmployeesList(){
			//load employes and allocate private property $employeesLlist
		}

		public function callIn(){
			//handle call by any available employee
		}

		public function getPhoneExtension(){
			return $this->phoneExtension();
		}
	}
```

Das Problem zu erkennen ist wichtiger als die Lösung zu erkennen, denn die genaue Darstellung des Problems führt zur Lösung.

Albert Einstein



Software Entwurf

- den Use case von Beginn bis Ende durchdenken
- mit welchen Daten werden wir es zu tun haben (input / output)
- Technologien und Konzepte definieren
- Objekte mit Eigenschaften und Methoden definieren
- aussagekräftige Namen auswählen

```
1.)   Daten entgegennehmen
2.)   Daten validieren
3.)   Daten verarbeiten
4.)   Ausgabe erzeugen
```

Erstes Beispiel

Konzipieren Sie die Implementierung einer Telefonzentrale als Service, das Anrufe für mehrere Abteilungen entgegen nimmt, verteilt und der Gegenstelle die Leitung, sowie den Namen der Abteilung und eines zuständigen Mitarbeiters zurück gibt.

Folgende **Schnittstelle** soll das Service bereitstellen:

- `GET Parameter String: “phoneNumber”`
- `GET Parameter String: “callerName”`

Der **Output** soll wie in folgender Form sein:

JSON Object:`{department: <string>, employee: <string>, line: <int>}`

Abgabe

Abgabe Files und **Planung**

- im Moodle unter Uebung 1
- NACHNAME_BE_Uebung1.zip (kein *.rar)
  - mit allen Files des Projektes und ein Bild der Planung
  - ohne diverse Metadaten von Netbeans oder PhPStorm
  - bitte keine versteckten Files die sonst in Ihrem Filesystem vorkommen *.ts, *.ps, usw.

- Fragen bitte im Diskussionsforum (Keine Scheu! Wir sind hier um zu Lernen!)
- Arbeiten Sie alleine!