Ajax Backend Erstellung mittels PHP

Agenda

- Ziel
- Kommunikation
- Aufteilung des Backends
- Testen des Backends

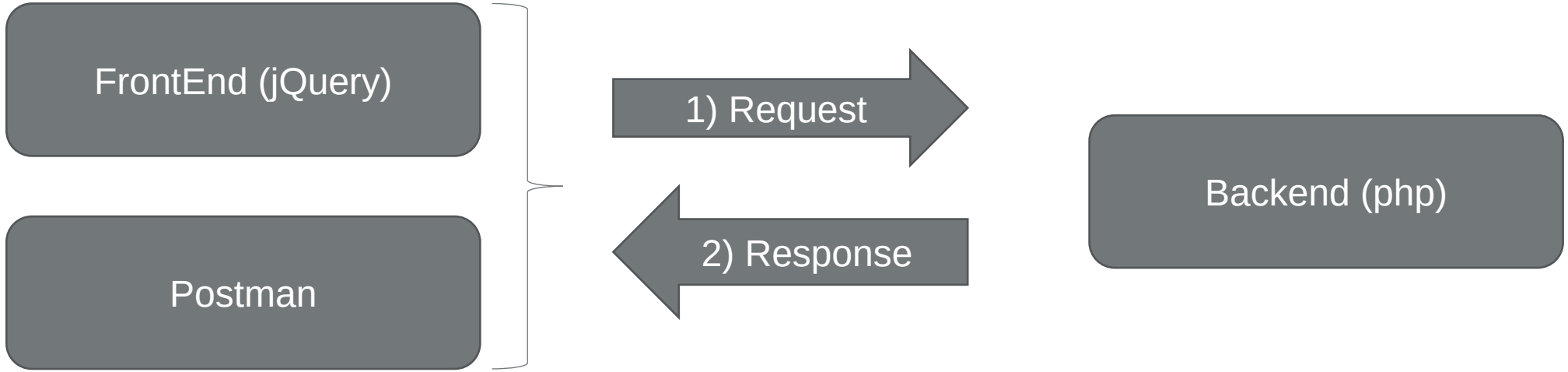
Ziel

Nach durcharbeiten des Foliensatzes sind sie in der Lage:

- **mittels PHP einfache REST Services zu implementieren**
- **JSON Daten zum Client zu senden bzw. vom Client zu empfangen**
- **ihr Backend zu strukturieren um wartbaren Code zu erzeugen**

Kommunikation

Kommunikationsablauf zwischen Client und Server (Frontend und Backend)



Kommunikation

Folgende Protokolle sind beim Datenaustausch zu betrachten

- **Transportprotokoll**

  bei RESTfull Services wird http/https verwendet (da wenig overhead)

  HTTP stellt verschiedene Methoden zur Verfügung (get, post, put, delete,…)

- **Messageprotokoll**

  Die Nachrichten werden im JSON Format übermittelt (weniger overhead als z.B. xml)

Kommunikation

Dementsprechend muss der http header so gesetzt werden, dass die Gegenstelle erkennen kann wie die Daten übermittelt werden:

- **http Status Code (z.B: 200 für ok, 404 für not found,…)**
- **Content Type (application/json)**

Umsetzung in PHP:

```php
header('Content-Type: application/json');
http_response_code($httpStatus);
```

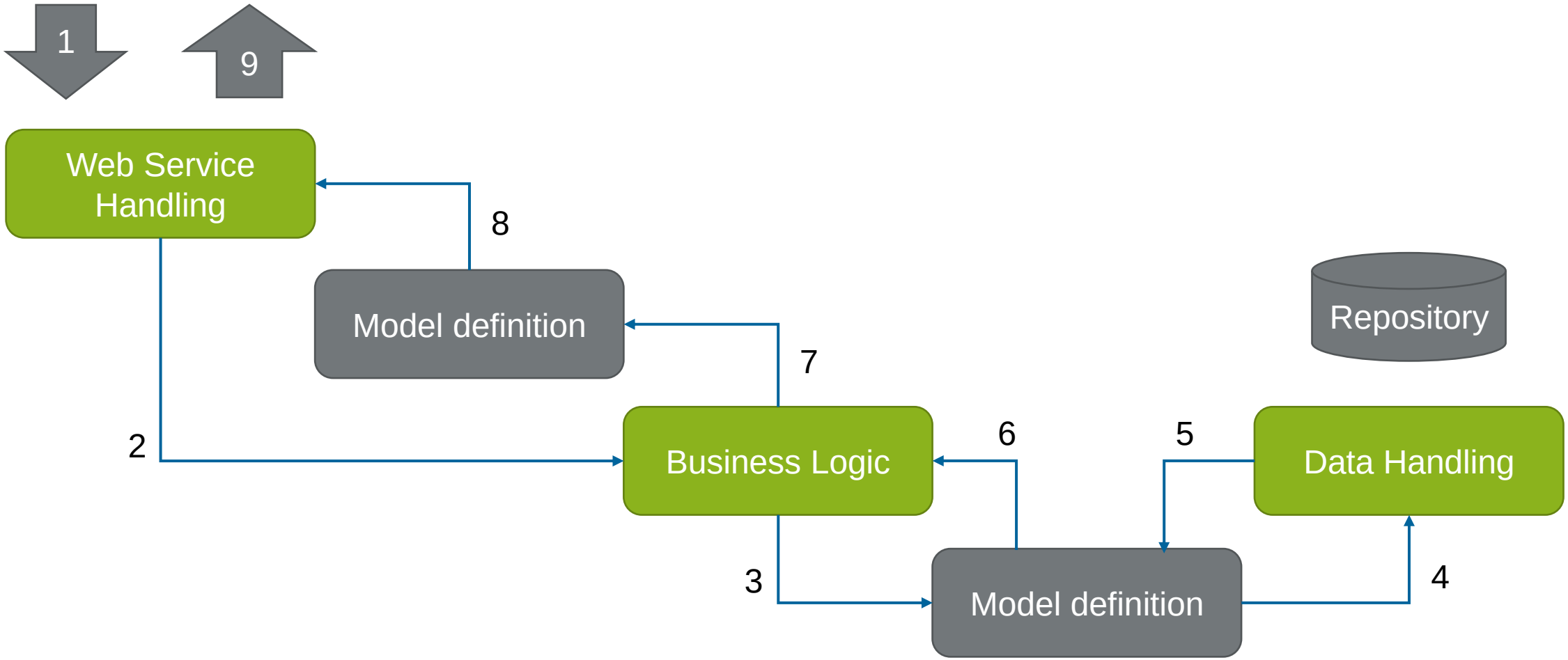
Aufteilung des Backends

- **Im Sinne der Wartbarkeit ist es sehr zu empfehlen unterschiedliche Zuständigkeiten im Code voneinander zu trennen.**

- **Beispiel für Zuständigkeiten in diesem Zusammenhang:**
  - **Request annehmen, Response verschicken**
  - **(Business) Logik**
  - **Definition der Datenstruktur (Model)**
  - **CRUD Operationen auf das Repository (Datenbank, File, …)**

Aufteilung des Backends

Daraus würde sich folgende Zusammespiel der einzelnen Teile ergeben:



Aufteilung des Backends

1 Request des Clients an Server

2 Die Business Logik entscheidet (anhand der Parameter, Methode,…) wie die Applikation auf diese Anfrage reagiert

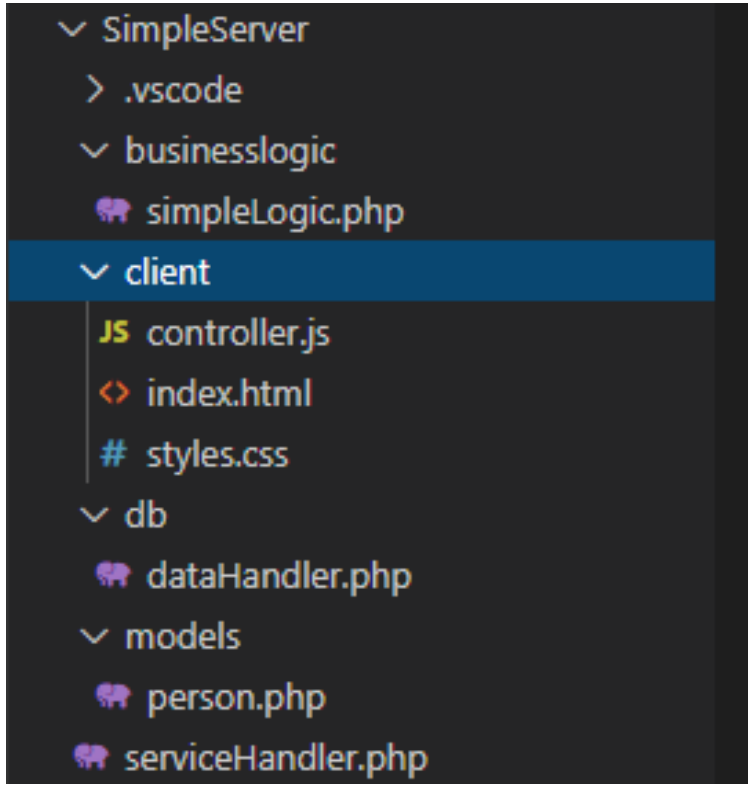
3-6 Durchführen der notwendigen CRUD Operationen gegen ein Repository. Zwischen Repository und Logik kann (muss aber nicht) eine Abstraktion des Modelles durchgeführt werden.

7-8 Ebenfalls kann hier eine Abstraktion des Modells durchgeführt werden

9 Übermitteln der Daten an das Service Handling welches den Response durchführt.

Aufteilung des Backends

Möglicher Projektaufbau:



Aufteilung des Backends

Möglicher Projektaufbau – Service Handling:

Include business Logic

Request Parameter prüfen

Aufruf der Logik

Response versenden

SimpleServer > serviceHandler.php > ...

```php
<?php
include("businesslogic/simpleLogic.php");

$param = "";
$method = "";

isset($_GET["method"]) ? $method = $_GET["method"] : false;
isset($_GET["param"]) ? $param = $_GET["param"] : false;

$logic = new SimpleLogic();
$result = $logic->handleRequest($method, $param);
if ($result == null) {
    response("GET", 400, null);
} else {
    response("GET", 200, $result);
}

function response($method, $httpStatus, $data)
{
    header('Content-Type: application/json');
    switch ($method) {
        case "GET":
            http_response_code($httpStatus);
            echo (json_encode($data));
            break;
        default:
            http_response_code(405);
            echo ("Method not supported yet!");
    }
}
```

Aufteilung des Backends

Möglicher Projektaufbau – Business Logik:

Include Repository Logik

Instanz der Repository Logik erstellen

Anhand der Logik wird response **Inhalt** vorbereitet

SimpleServer > businesslogic > simpleLogic.php > ...

```php
<?php
include("db/dataHandler.php");

class SimpleLogic
{
    private $dh;
    function __construct()
    {
        $this->dh = new DataHandler();
    }

    function handleRequest($method, $param)
    {
        switch ($method) {
            case "queryPersons":
                $res = $this->dh->queryPersons();
                break;
            case "queryPersonById":
                $res = $this->dh->queryPersonById($param);
                break;
            case "queryPersonByName":
                $res = $this->dh->queryPersonByName($param);
                break;
            default:
                $res = null;
                break;
        }
        return $res;
    }
}
```

Aufteilung des Backends

Möglicher Projektaufbau – Repository Handling:

Include des Models

In diesen Fall wird nicht auf eine DB zugegriffen sondern lediglich Demodaten generiert. Für die Demodaten werden unterschiedliche Read Funktionen zur Verfügung gestellt.

SimpleServer > db > dataHandler.php > ...

```php
<?php
include("./models/person.php");
class DataHandler
{
    public function queryPersons()
    {
        $res =  $this->getDemoData();
        return $res;
    }

    public function queryPersonById($id)
    {
        $result = array();
        foreach ($this->queryPersons() as $val) {
            if ($val[0]->id == $id) {
                array_push($result, $val);
            }
        }
        return $result;
    }

    public function queryPersonByName($name)
    {
        $result = array();
        foreach ($this->queryPersons() as $val) {
            if ($val[0]->name == $name) {
                array_push($result, $val);
            }
        }
        return $result;
    }

    private static function getDemoData()
    {
        $demodata = [
            [new Person(1, "Jane", "Doe", "jane.doe@fhtw.at", 1234567, "Central IT")],
            [new Person(2, "John", "Doe", "john.doe@fhtw.at", 34345654, "Help Desk")],
            [new Person(3, "baby", "Doe", "baby.doe@fhtw.at", 54545455, "Management")],
            [new Person(4, "Mike", "Smith", "mike.smith@fhtw.at", 343477778, "Faculty")],
        ];
        return $demodata;
    }
}
```

Aufteilung des Backends

Möglicher Projektaufbau – Model Definition

Definition eines einfachen models

SimpleServer > models > person.php > ...

```php
<?php
class Person {
    public $id;
    public $firstname;
    public $lastname;
    public $email;
    public $phone;
    public $department;

    function __construct($id, $fn, $ln, $mail, $phone, $dept) {
        $this->id = $id;
        $this->firstname = $fn;
        $this->lastname=$ln;
        $this->email=$mail;
        $this->phone=$phone;
        $this->department=$dept;

    }
}
```

Aufteilung des Backends

JSON

Mittels Header wird der Content typ gesetzt!

Json_encode() erstellt formatiert ein Objekt Im JSON Format

Json_decode() für die andere Richtung verwenden

```php
function response($method, $httpStatus, $data)
{
    header('Content-Type: application/json');
    switch ($method) {
        case "GET":
            http_response_code($httpStatus);
            echo (json_encode($data));
            break;
        default:
            http_response_code(405);
            echo ("Method not supported yet!");
    }
}
```

Mittels echo() Methode wird der Response zum Client geschickt

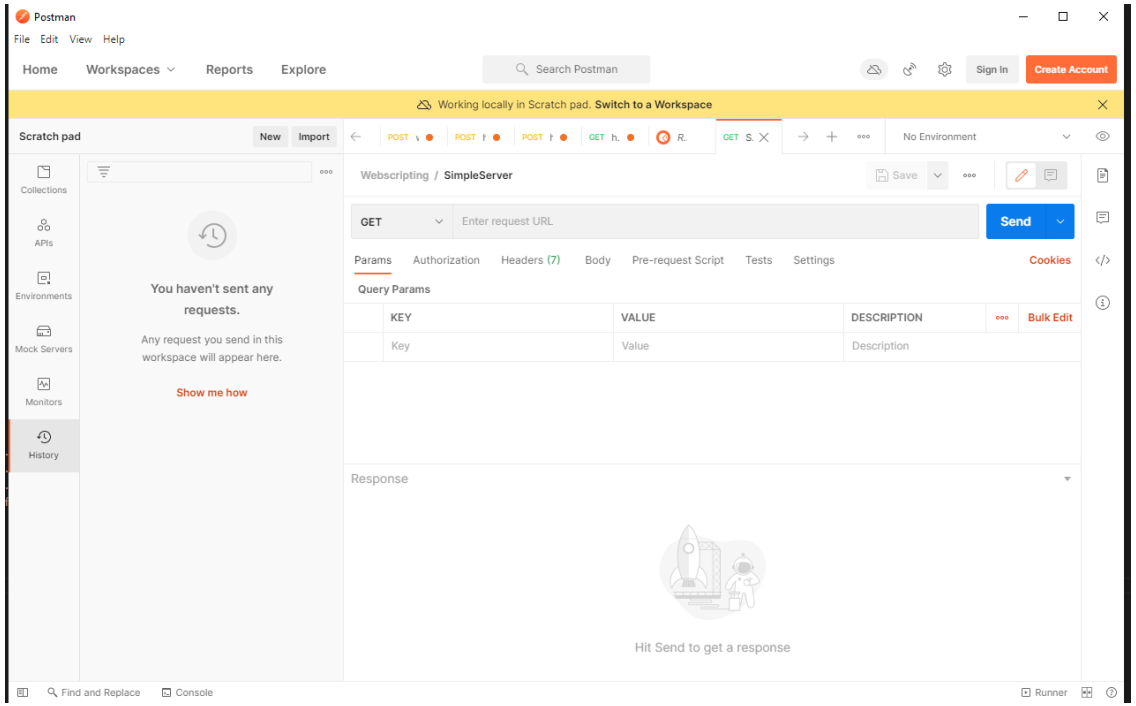
Testen des Backends

Mittels Software wie z.B: Postman

https://app.getpostman.com/app/download/win64

New => Request

- Request Name setzen
- Collection anlegen
- URL eingeben
- Parameter Setzen
- Send klicken
- Response einsehen



Testen des Backends

Postman

Method

URL

Params

Response