HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

WEB-APPLIKATIONEN

MVC 1: MODEL

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

MVC

ALLGEMEINES ÜBER MVC

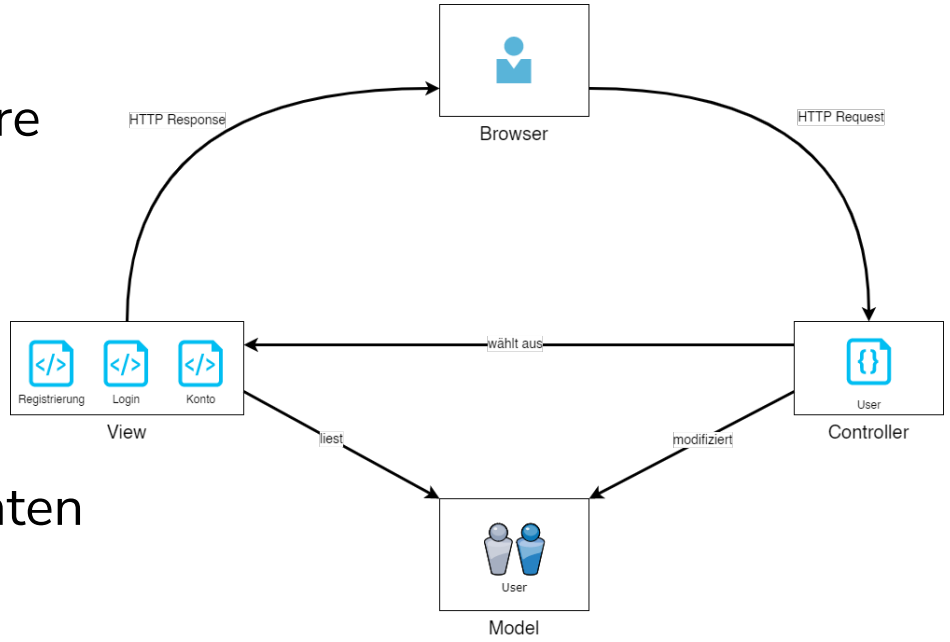
› Model View Controller (MVC)

› (Entwurfs-) Muster zur Unterteilung von Software

› Bestandteile:

- Datenmodell (Model)
- Präsentation (View)
- Programmsteuerung (Controller)

› Wiederverwendbarkeit der einzelnen Komponenten



MODEL

› Model enthält Daten

› Kann Methoden zur Änderung von Daten enthalten (je nach Umsetzung)

Beispiel: Model Auto

ID: 1

Farbe: Rot

Türen: 4

Reifen: 4

HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

WEB-APPLIKATIONEN

JSON

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

JSON

› JavaScript Object Notation (JSON)
  - Einfach lesbare Textform
  - Datenaustausch zwischen Anwendungen

› JSON ist unabhängig von der Programmiersprache
  - Parser existieren in allen gängigen Sprachen

› Besteht aus Attribut – Wert (Int, Float, String, …) paaren

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

DATENTYPEN

› Objekt wird durch geschweifte Klammern gekennzeichnet

› JSON kennt folgende Datentypen

- String
- Fließkommazahl
- Objekt
- Boolean
- Array (Liste !!!)
- Ganzzahl
- Null

```
{
  "Herausgeber": "Xema",
  "Nummer": "1234-5678-9012-3456",
  "Deckung": 2e+6,
  "Waehrung": "EURO",
  "Inhaber":
  {
    "Name": "Mustermann",
    "Vorname": "Max",
    "maennlich": true,
    "Hobbys": ["Reiten", "Golfen", "Lesen"],
    "Alter": 42,
    "Kinder": [],
    "Partner": null
  }
}
```

JSON IN JAVASCRIPT

› JSON String in JavaScript

```
let text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

› JSON Objekt erstellen (Parse)

```
const obj = JSON.parse(text);
```

› JSON Objekt zu String (Stringify)

```
const myJSON = JSON.stringify(obj);
```

› JSON-Werte abrufen („Anna“)

```
obj.employees[1].firstName
```

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

WEB-APPLIKATIONEN

DATEIEN LESEN UND SCHREIBEN

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

DATEIEN LESEN/SCHREIBEN IN NODE.JS

› Modul: „fs“ (FileStream)

› Lesen
- readFileSync (Synchron)
- readFile (Asynchron)

› Schreiben
- writeFileSync (Synchron)
- writeFile (Asynchron)

```
const fs = require('fs');

let rawdata = fs.readFileSync('student.json');
let student = JSON.parse(rawdata);
```

```
const fs = require('fs');

let student = {
    name: 'Mike',
    age: 23,
    gender: 'Male',
    department: 'English',
    car: 'Honda'
};

let data = JSON.stringify(student);
fs.writeFileSync('student-2.json', data);
```

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

DATEIEN LESEN/SCHREIBEN IN NODE.JS

› JSON mit require lesen

```
var data = require('student.json');
```

› Lesen

- Synchron
- Wird nur 1x gelesen (cache)
- Umwandlung von JSON zu Objekt erfolgt automatisch

```
var studierende = {
    studi: require('student.json'),
    setStudi: function (data) {this.studi = data}
};
```

› JS-Objekt mit Daten und Methoden

- studierende.studi enthält Daten
- setStudi speichert neue Inhalte in Struktur

```
var neuerStudi = {
  vorname: req.body.vorname,
  nachname: req.body.nachname,
  matrikel: req.body.matrikelnummer
}
studierende.setStudi([...studierende.studi, neuerStudi]);
```

- Speichert NICHT in Datei!

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

ASYNCHRONES SCHREIBEN

› Bei größeren Dateien/Datensätzen

› Funktion muss als Asynchron deklariert werden

```
const fsPromises = require('fs').promises;
async function writeStudierende(data) {
    try {
        await fsPromises.writeFile(
            'student.json',
            JSON.stringify(data,null,2)
        );
        console.log(data);
    } catch (err) {
        res.status(500).json({ 'message': err.message });
    }
}
```

› Aufruf der Funktion (z.B. innerhalb einer POST-Route)

```
writeStudierende(studierende.studi);
```

FILE-UPLOAD

› File-Upload installieren

```
npm install express-fileupload
```

› Middleware in App.js hinzufügen

```
//File-Upload
var fileupload = require('express-fileupload');
app.use(fileupload({
  createParentPath:true
}));
app.use(express.static('public/uploads'));
```

FILE-UPLOAD

› index.js

```
router.post('/upload', function(req, res, next) {
  let file = req.files.file;
  file.mv('public/uploads/'+file.name);
  console.log(file.name);
  res.render('index', { title: 'Express' });
});
```

› formular.ejs

```
<form method="post" action="/upload" enctype="multipart/form-data">
    <label for="datei">Datei:</label>
    <input type="file" id="datei" name="file">
    <input type="submit" name="senden">
</form>
```

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

ZUSAMMENFASSUNG

› Model-View-Controller erhöht die Wiederverwendbarkeit von Software

› Model enthält Daten

› JSON ist unabhängig von der Programmiersprache

Beinhaltet Attribut-Wert-Paare

› Über FileStream können Dateien gelesen und geschrieben werden

Synchrones und Asynchrones lesen/schreiben kann Vor- und Nachteile haben