

WEBTECHNOLOGIEN

05 – FORMULARE

PROF. DR. MARKUS HECKNER

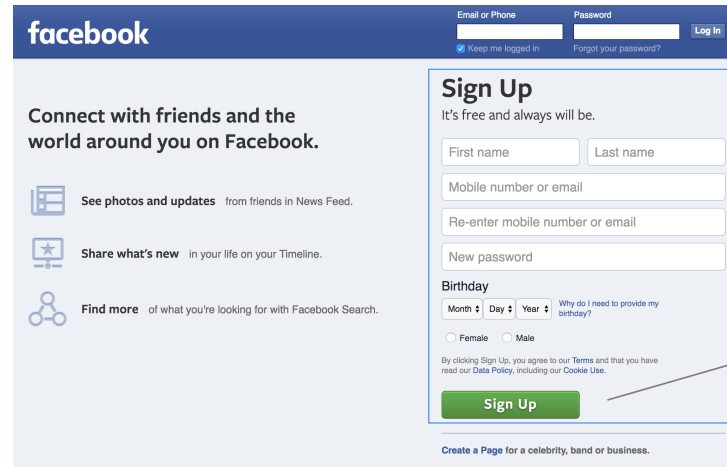
THE STORY SO FAR

- HTML und CSS für das Frontend der Website
- Web-App als dynamischer Webserver, der die aufgerufenen Seiten an den Client zurückgibt, der auf Anfragen reagieren kann und diese nach dem Model View Controller-Prinzip verarbeitet
- PostgreSQL als Datenbank für die Models
- Anfragen an die Web-App nur über URLs, z.B. `/playlist/1` zur Abfrage der Playlist mit der ID 1

Jetzt: Wie können Nutzer Eingaben an die Web-App schicken?

HTML FORMULARE ERMÖGLICHEN DIE ERFASSUNG UND WEITERGABE VON NUTZEREINGABEN

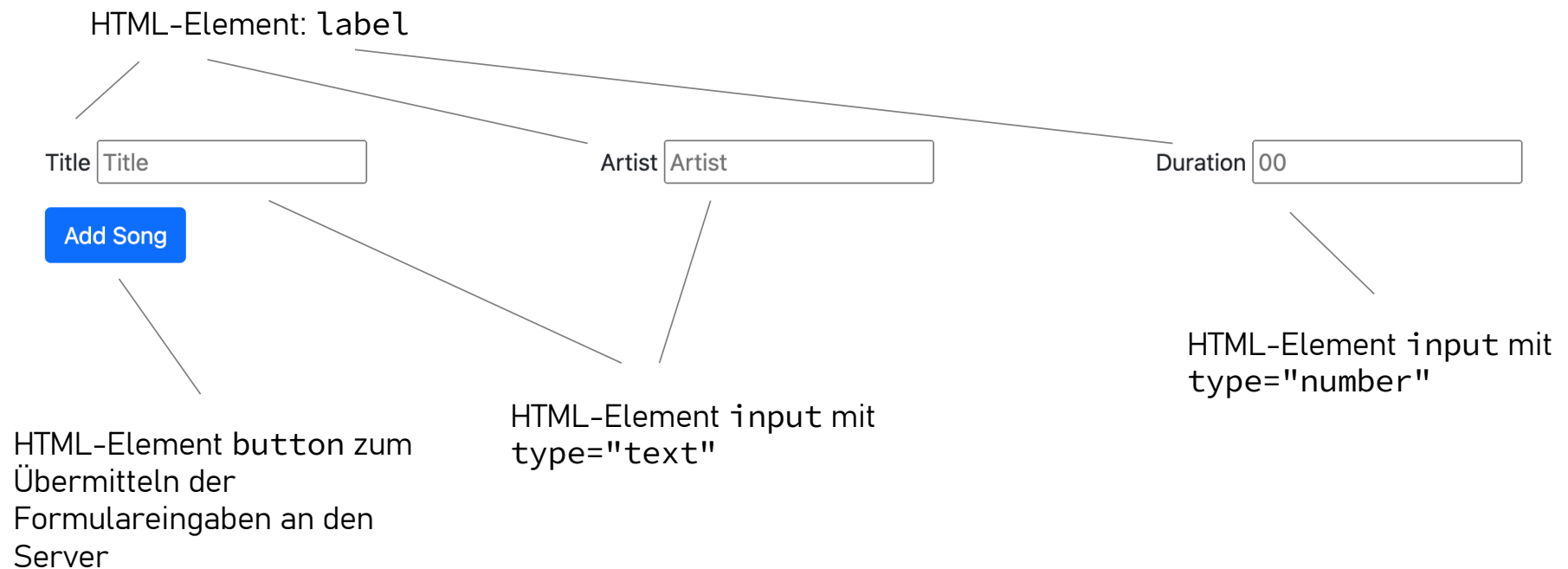
- Interaktionsmöglichkeit für Nutzer mit Webseiten
- Daten werden an Webserver zur weiteren Verarbeitung übermittelt (z.B. als Werte für Abfragen an Datenbanken, Überprüfung von Login-Daten, Bestellen von Artikeln bei Amazon, usw.)
- Formulare bestehen aus HTML Elementen zur Eingabe (Text, Radiobuttons, Checkboxes, ...) und zum Absenden des Formulars (Button)

A screenshot of the Facebook 'Sign Up' page. The page has a blue header with the Facebook logo and login options. The main content area is light blue and contains the 'Sign Up' heading, a subtext 'It's free and always will be.', and several input fields: 'First name', 'Last name', 'Mobile number or email', 'Re-enter mobile number or email', and 'New password'. Below these is a 'Birthday' section with dropdowns for 'Month', 'Day', and 'Year', and radio buttons for 'Female' and 'Male'. At the bottom is a green 'Sign Up' button. A line from the text 'Formular mit Feldern' points to the input fields, and another line from 'Übermitteln der Formulareingaben an den Server' points to the 'Sign Up' button.

Formular mit
Feldern

Übermitteln
der
Formular-
eingaben an
den Server

FORMULAR IM BROWSER ZUM HINZUFÜGEN EINES NEUEN SONGS



FORMULARE ÜBERMITTELN DIE DATEN PER HTTP-POST AN DEN SERVER

- GET – Daten vom Server holen
- POST – Erstellen einer Ressource (Daten auf dem Server ablegen) } Formulare (heute)
- PATCH – Daten auf dem Server aktualisieren } Nicht in diesem Kurs
- DELETE – Löschen von Daten auf dem Server }
- ...

HANDLEBARS CODE DES FORMULARS (OHNE BOOTSTRAP)

Was passiert, beim Absenden des Formulars?

Hier Aufrufen einer parametrisierten Route ({{playlist.id}} wird serverseitig durch die ID der Playlist ersetzt), um einen neuen Song hinzuzufügen

http-Methode POST (wird im Router wichtig)

ROUTING UND FORMULARE

```
<form action="/playlist/{{playlist.id}}/addsong" method="POST">
  <label>Title</label> <input name="title" type="text" placeholder="Title" />
  <label>Artist</label> <input name="artist" type="text" placeholder="Artist" />
  <label>Duration</label> <input name="duration" type="number" placeholder="00" />
  <button type="submit">Add Song</button>
</form>
```

Eingabefelder

Schickt das Formular ab, d.h. die in action definierte URL wird mit der in method definierten http-Methode aufgerufen

name ermöglicht es dem Server die Eingaben aus dem Formular auszulesen

Welche Art von Daten wird eingegeben?

HTTP REQUEST HEADER – KURZ UND KNAPP

Wie werden die Daten
übermittelt? Hier: POST als Teil
des headers...

▼ Request Headers

:authority: playlist-2.mheckner.repl.co

:method: POST

:path: /playlist/2/addsong

:scheme: https

accept: text/html,application/xhtml+xml,application/

*;q=0.8,application/signed-exchange;v=b3;q=0.9

accept-encoding: gzip, deflate, br

accept-language: en-US,en;q=0.9,de;q=0.8,la;q=0.7

Welchen Pfad wollen wir?

▼ Form Data

view parsed

title=Song+2&artist=Blur&duration=122

Body (Inhalt) des Requests: Hier
stehen die Daten aus dem
Formular als Key-Value Paare...

Key entspricht dem name-Attribut
des Formulars!

ROUTER NIMMT DIE ABFRAGE ENTGEGEN

routes.js:

Nur für POST-requests

```
router.post('/playlist/:id/addsong', playlist.addSong);
```

Route - Parameter `:id` für die Playlist zu der der Song hinzugefügt werden soll.

Wird durch Handlebars ersetzt, bevor das Formular an den Client geschickt wird (vgl. http-Request header)

Aufrufen der Funktion `addSong` des `playlist-Controllers`

AUSLESEN DER FORMULARDATEN IM PLAYLIST-CONTROLLER

controllers/playlist.js:

```
async addSong(request, response) {  
  const playlistId = request.params.id;  
  const newSong = {  
    title: request.body.title,  
    artist: request.body.artist,  
    duration: Number(request.body.duration)  
  };  
  logger.debug("New Song", newSong);  
  await songStore.addSong(playlistId, newSong);  
  response.redirect("/playlist/" + playlistId);  
},
```

body ist der Inhalt des Requests und enthält die geparsen name-Attribute als Keys – Inhalte sind die Eingaben in den Formularfeldern (man muss den String aus Key-Value-Paaren nicht selbst zerlegen, das macht der body-parser)

BODY-PARSER WIRD BENÖTIGT, UM DIE FORMULARDATEN AM SERVER AUSZULESEN

app.js:

```
const bodyParser = require("body-parser");
```

requiren des Moduls body-parser in app.js,
und ...

```
app.use(bodyParser.urlencoded({ extended: false }));
```

... konfigurieren des Moduls (erst dann wird
das Objekt body dem request-Objekt
hinzugefügt (body ist ohne body-parser
undefined))

FORMULARE LASSEN SICH MIT BOOTSTRAP VISUELL ANPASSEN

Email address

We'll never share your email with anyone else.

Password

☐ Check me out

Submit

Vgl. Offizielle Dokumentation von Bootstrap 5: <https://getbootstrap.com/docs/5.0/forms/overview/>

FAZIT

- Formulare erlauben es Nutzern mit der Web-App zu interagieren (z.B. registrieren, einloggen, ausloggen, einen neuen Song hinzufügen, Songs löschen, etc.)
- Formulare werden in HTML mit dem Tag `<form>` gekennzeichnet
- Jedes Formular enthält ein oder mehrere `<input>`-Elemente, deren Attribut `name` festlegt unter welchem Key die Inhalte der `<input>`-Elemente auf dem Server ausgelesen werden können
- Formulare können mit Bootstrap-Klassen visuell angepasst werden