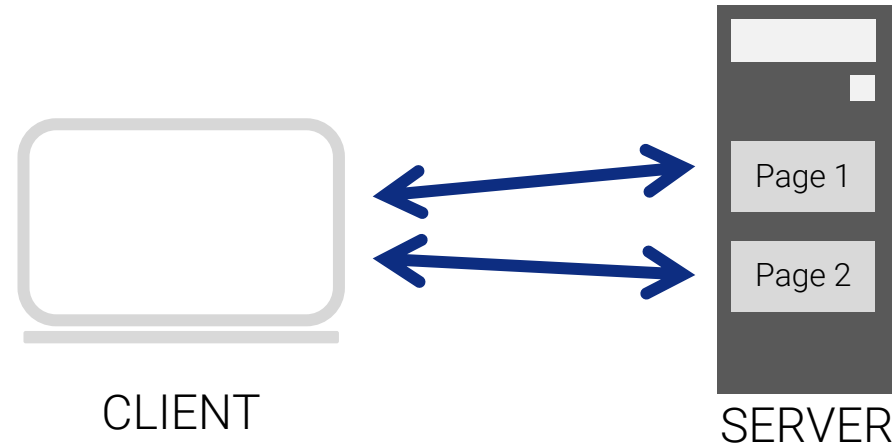


05 — WEBTECHNOLOGIEN EXPRESS, ROUTING UND FORMULARE

PROF. DR. MARKUS HECKNER

EXPRESS UND LINKEN - CLIENT FRAGT SEITE VOM SERVER AB UND USER KLICKT AUF LINK ZU EINER WEITEREN SEITE



```
<a href="/about">About us</a>
```

Antwort als HTML-String

```
a(href="/about") About us
```

HTML-Attribute in pug



EXPRESS UND AUSLIEFERN VON STATISCHEM CONTENT – HIER CSS UND JAVASCRIPT

- Client fragt Seite ab – Server erzeugt HTML Antwort, diesmal mit CSS im header
- Express so konfigurieren, dass CSS als Antwort auf weitere requests gesendet wird

```
doctype html
html
  head
    link(rel="stylesheet", type="text/css", href="/stylesheets/styles.css")

  body
    h1 Hello! I am a server that also serves static CSS files!
```



ERWEITERUNG AIRLINE APP

ANZEIGE ALLER FLÜGE

/flights

1. Client sendet GET Request an einen "Resource Path":



Ressourcentyp – Früher meist html-Dateien,
heute oft dynamisch generierte Antworten des Webserver...

2. Server "rendert" response und schickt Antwort zurück an den Client
Hier: Alle Flüge

ERWEITERUNG AIRLINE APP

ANZEIGE VON FLUGDETAILS UND PASSAGIERE FÜR EINEN BESTIMMTEN FLUG ANZEIGEN

/flights/3

1. Client sendet GET Request an einen "Resource Path":

Identifier, d.h. welche Ressource meinen wir?
Hier: Identifier = Primärschlüssel der Datenbank

Ressourcentyp

2. Server "rendert" response und schickt Antwort zurück an den Client

AIRLINE APP - ERWEITERT

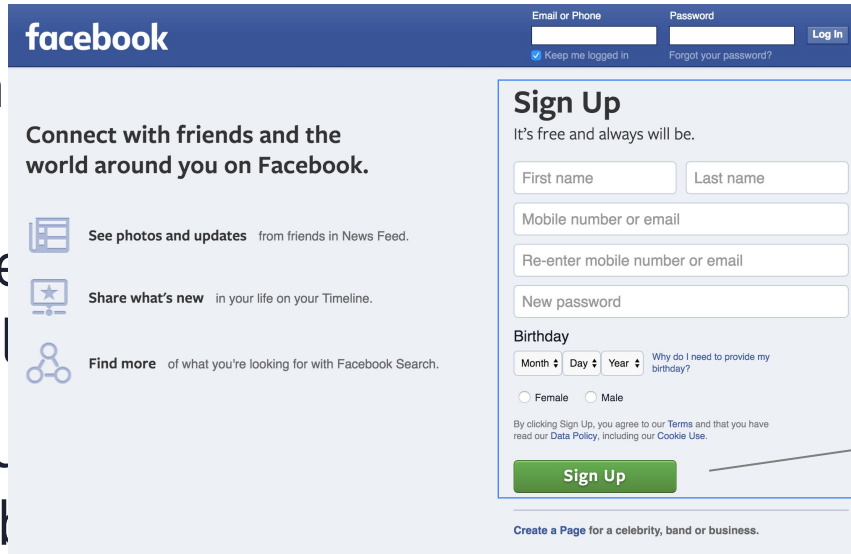
- Nutzer kann einen gültigen Flug über Dropdown auswählen
- Nutzer kann seinen Namen eingeben und den Flug per Klick buchen (d.h. Nutzer wird als Passagier in der Datenbank abgespeichert)
- Airline App bestätigt dem Nutzer die erfolgreiche Buchung
- Zusätzlich kann Nutzer die Seite /flights aufrufen und dort alle bestehenden Flüge sehen
- Bei Klick auf einen Flug werden Details zum Flug und den Passagieren des Flugs angezeigt – Dazu eine spezielle Route /flights/:id mit Parametern



Fehlt mehr zu Routing

HTML FORMULARE ERMÖGLICHEN DIE ERFASSUNG UND WEITERGABE VON NUTZEREINGABEN

- Interaktion
- Daten als Webformulare
- Formular-Elemente
- Radiobuttons

A screenshot of the Facebook 'Sign Up' form. The form is titled 'Sign Up' and includes a sub-header 'It's free and always will be.' Below this, there are input fields for 'First name', 'Last name', 'Mobile number or email', 'Re-enter mobile number or email', and 'New password'. There is also a 'Birthday' section with dropdowns for 'Month', 'Day', and 'Year', and radio buttons for 'Female' and 'Male'. At the bottom, there is a green 'Sign Up' button. The form is set against a light blue background with the Facebook logo at the top left.

Formular mit
Webseiten

eren Verarbeitung übermittelt (z.B. ken, Überprüfung von Login-Daten, usw.)

Übermitteln
der Daten an
den Server
enten zur Eingabe (Text,
um Absenden des Formulars (Button)

DESIGN UNSERES ERSTEN FORMULARS

A hand-drawn sketch of a contact form on graph paper. The form is titled "CONTACT" at the top left. It contains three input fields: "Name:" followed by a single-line text box, "E-mail:" followed by a single-line text box, and "Message:" followed by a large multi-line text area. At the bottom of the form is a rounded rectangular button labeled "SEND YOUR MESSAGE".

Bildquelle: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms/My_first_HTML_form

DAS HTML FORM ELEMENT LEGT DIE AKTION BEIM ABSENDEN DES FORMULARS FEST UND DIENT ALS CONTAINER FÜR FORMULARELEMENTE

```
<form action="/welcome" method="post">
```

<!-- Hier der Rest des Formulars -->

```
</form>
```

Was passiert, beim Absenden des Formulars?
Hier Aufrufen einer Seite, die dem Nutzer Feedback gibt und seine Daten noch einmal anzeigt...

Wie werden die Daten übermittelt, hier:
http-Methode POST

HINZUFÜGEN VON EINGABEELEMENTEN

```
<form action="/welcome" method="post">
```

```
  <div>
```

```
    <label for="name">Name:</label>
```

```
    <input type="text" id="name" />
```

```
  </div>
```

```
  <div>
```

```
    <label for="mail">E-mail:</label>
```

```
    <input type="email" id="mail" />
```

```
  </div>
```

```
  <div>
```

```
    <label for="msg">Message:</label>
```

```
    <textarea id="msg"></textarea>
```

```
  </div>
```

```
</form>
```

Label zur Beschreibung des Felds

Feld zur Eingabe des Namens
(einzellig)

Unterscheidet, was eingegeben
werden darf... Validierung des
Formulars beim Abschicken...

Feld zur Eingabe der Nachricht
(mehrzeilig)

for Attribut "bindet" das Label an das
Eingabeelement über das Attribut id des
Eingabeelements – Ermöglicht z.B. Klicken
auf das Label, um das Feld zu aktivieren
(Usability!) Vgl. z.B. trello.com!

ERGÄNZEN EINES BUTTONS ZUM ABSENDEN DES FORMULARS

```
<form action="/welcome" method="post">
```

```
...
```

```
  <div class="button">
```

```
    <button type="submit">Send your message</button>
```

```
  </div>
```

```
</form>
```

Text für den Button...

Type submit führt die in **action** des Formulars definierte Aktion aus – Beim Absenden werden auch die Eingabeelemente auf zulässige Werte geprüft (vgl. Email...)



HINZUFÜGEN VON CSS ANWEISUNGEN

```
label {
  padding-top: 12px;
  display: block;
}

input, textarea {
  padding: 10px;
  width: 80%;
}

...

<form action="/welcome" method="post">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" />
  </div>
  <div>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" />
  </div>
  <div>
    <label for="msg">Message:</label>
    <textarea id="msg"></textarea>
  </div>
</form>
```



UND WIE KOMMEN DIE DATEN JETZT BEIM SERVER AN?



SENDEN DER FORMULARDATEN

/index

My first web form

Name:

The Waif

E-mail:

waif@faceless-men.org

Message:

Just wanted to say hello!

Send your message

Jetzt Übermittlung der Daten per POST
im http request...

AUSLESEN DER DATEN AUF DEM SERVER

/welcome

Thank you for your message!

Name: The Waif

E-Mail: waif@faceless-men.org

Message: Just wanted to say hello!

Server wertet Formulardaten aus und schickt
response an den Client zurück.

ÜBERMITTELN DER DATEN: DIE ZU ÜBERMITTELNDEN DATEN MÜSSEN "BENANNT" WERDEN

```
<form action="/welcome" method="post">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"/>
  </div>
  <div>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" name="mail_address"/>
  </div>
  <div>
    <label for="msg">Message:</label>
    <textarea id="msg" name="message"></textarea>
  </div>
  <div class="button">
    <button type="submit">Send your message</button>
  </div>
</form>
```

Attribut „name“ definiert den „Variablennamen“ für die zu sendenden Daten (Daten können so auf dem Server ausgelesen werden)

HTTP REQUEST HEADER – KURZ UND KNAPP

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

▼ Request Headers

view parsed

Welchen Pfad wollen wir?

```
POST /welcome HTTP/1.1
Host: localhost:3000
Connection: keep-alive
Content-Length: 75
Cache-Control: max-age=0
Origin: http://localhost:3000
```

▼ Form Data

view parsed

```
name=The+Waif&email=waif%40faceless-men.org&msg=Just+wanted+to+say+hello%21
```

Hier stehen die Daten aus dem
Formular als Key-Value Paare...
Key entspricht dem name Attribut
des Formulars!

HTTP RESPONSE HEADER – KURZ UND KNAPP

Alles OK, Webserver hat die angeforderte
Route gefunden...

▼ **Response Headers** view parsed

```
HTTP/1.1 200 OK
Date: Wed, 15 Jun 2016 06:40:31 GMT
Server: Apache
X-Frame-Options: sameorigin
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Was kommt zurück? Hier HTML, kodiert in
Unicode...

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8" />
  <title>Message received successful</title>
  <link rel="stylesheet" href="css/form_demo.css">
</head>
<body>
  <h1>Thank you for your message!</h1>
  ...
```

Dann erst der
„Content“ der Antwort...

FORMULAR IN DER SYNTAX VON PUG

```
form(id="mail_form", action="/welcome", method="post")
  div
    label(for="name") Name:
    input(type="text", name="name", id="name")
  div
    label(for="mail") E-mail:
    input(type="email", name="email" id="mail")
  div
    label(for="msg") Message:
    textarea(id="msg" name="msg")
  div(class="button")
    button(type="submit") Send your message
```

AUSLESEN UND DARSTELLEN DER DATEN AUF DEM SERVER (1/2)

Routing: Nur wenn der Nutzer die Seite `/welcome` über das Formular aufgerufen hat, wurden die Daten per **POST** übergeben...

Parser wird benötigt, um Post-Daten aus dem body des Requests auszulesen

```
app.post("/welcome", urlencodedParser, function(request, response) {  
    response.render("welcome", {data: request.body});  
});
```

Übergabe des kompletten bodys (d.h. hier die ganzen POST-Daten) an die Template Engine – Hier key-value pairs aus dem Formular enthalten...

AUSLESEN UND DARSTELLEN DER DATEN AUF DEM SERVER (2/2)

Ausgabe der empfangenen Daten auf der Website...

```
doctype html
html
  head
    link(rel='stylesheet' href='stylesheets/styles.css')
  body
    h1 Hello #{data.name}!
    h2 Thank you for your message!
    p E-Mail: #{data.email}
    p Message: #{data.msg}
```

Zugriff über die im HTML Formular definierten Keys (über das Attribut „name“)

TEMPLATE-INHERITANCE REDUZIERT “DUPLICATE CODE” IN DEN TEMPLATE-DATEIEN

```
doctype html
html
  head
    link(rel='stylesheet'
href='stylesheets/styles.css')
  body
    block content
```

layout.pug

```
extends layout.pug

block content
  h1 My first web form

  form(id=...)
```

index.pug

```
extends layout.pug

block content
  h1 Hello #{data.name}!
  ...
```

welcome.pug

hello_server_form_template_inheritance

