WEBTECHNOLOGIEN

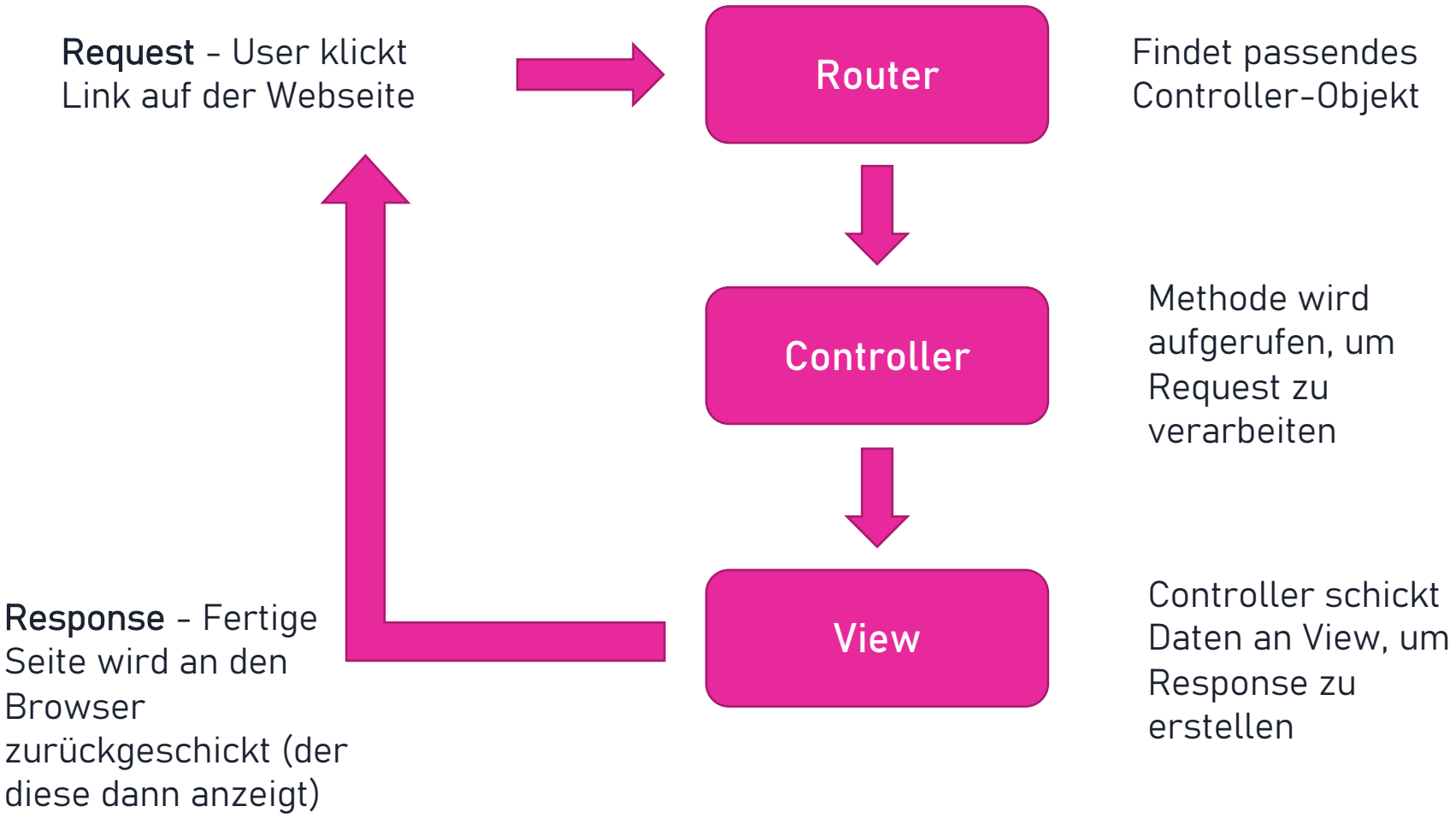
04 — WEB APPS — BACKEND

PROF. DR. MARKUS HECKNER

REQUEST – RESPONSE LEBENSZYKLUS

1. **Request** - User klickt Link auf der Webseite
2. **Router** – Findet passendes Controller-Objekt
3. **Controller** – Methode wird aufgerufen, um Request zu verarbeiten
4. **View** – Controller schickt Daten an View, um Response zu erstellen
5. **Response** – Fertige Seite wird an den Browser zurückgeschickt (der diese dann anzeigt)

ROUTER => CONTROLLER => VIEW



REQUEST - USER KLICKT LINK AUF DER WEBSEITE

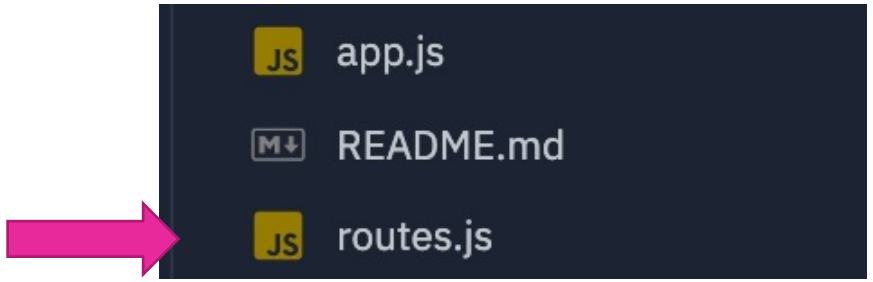
```html
<li class="nav-item">
  <a class="nav-link" id="about" href="/about">About</a>
</li>
```

Requests erzeugt durch:

- Attribut `href` in Links (<a> Tags)
- Attribut `href` in Buttons
- Attribut `action` in Formularen (später)

ROUTER – FINDET PASSENDES CONTROLLER-OBJEKT

request landet beim Router



routes.js

```
1  const express = require("express");
2  const router = express.Router();
3
4  const home = require("./controllers/home.js");
5  const about = require("./controllers/about.js");
6
7  router.get("/", home.index);
8  router.get("/about", about.index);
9
10 module.exports = router;
```

Router importiert zwei Controller

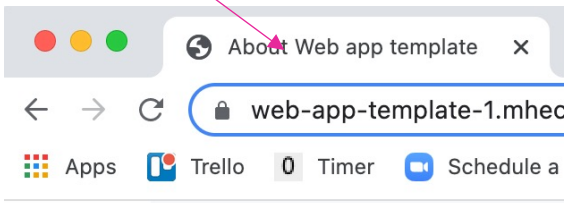
Router “matcht” die Anfrage auf die Controller-Objekte und leitet den request an die Controller-Methoden weiter

CONTROLLER – METHODE WIRD AUFGERUFEN, UM REQUEST ZU VERARBEITEN

controllers/about.js

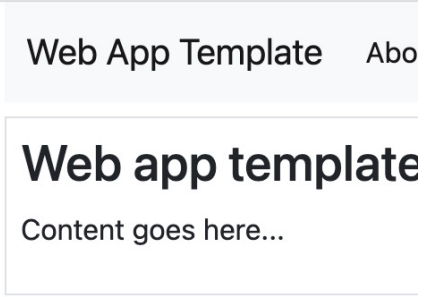
```
1  const logger = require("../utils/logger.js");
2
3  const about = {
4    index(request, response) {
5      logger.info("about rendering");
6      const viewData = {
7        title: "About Web app template"
8      };
9      response.render("about", viewData);
10   }
11 };
12
13 module.exports = about;
14
```

Controller schickt Daten an View, um Response zu erstellen – Template wird “gerendert”, d.h. mit Daten (hier: `title`) befüllt und die response wird an den Client zurückgesendet



views/layouts/main.hbs

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>{{title}}</title>
```



DAS ABOUT CONTROLLER OBJEKT

- Eine Methode `index` mit zwei Parametern:
  - Request – Objekt mit Details zur Anfrage des Clients
  - Response – Objekt, das benutzt wird um Anfrage an den Client zurückzusenden

controllers/about.js

```
1  const logger = require("../utils/logger.js");
2
3  const about = {
4    index(request, response) {
5      logger.info("about rendering");
6      const viewData = {
7        title: "About Web app template"
8      };
9      response.render("about", viewData);
10   }
11 };
12
13 module.exports = about;
14
```

Erzeugt Logausgabe auf der Konsole (in replit)

Erzeugt Objekt `viewData` mit einer Eigenschaft `title`

DER CONTROLLER SENDET DIE DATEN AN DEN VIEW UM EINE RESPONSE ZU ERZEUGEN

Importieren des Loggers

Export des about-Objekts, damit es vom Router verwendet werden kann

controllers/about.js

```
1  const logger = require("../utils/logger.js");
2
3  const about = {
4    index(request, response) {
5      logger.info("about rendering");
6      const viewData = {
7        title: "About Web app template"
8      };
9      response.render("about", viewData);
10   }
11 };
12
13 module.exports = about;
14
```

Render erzeugt die response (HTML-Code) und verschickt diese anschließend zurück an den Client

Name des zu rendernden Views

Daten die für das rendern des Views benötigt warden (warden in den View “injected” bzw. eingesetzt)

ZUSAMMENARBEIT ZWISCHEN BACK-END UND FRONTEND

controllers/about.js

```
1  const logger = require("../utils/logger.js");
2
3  const about = {
4    index(request, response) {
5      logger.info("about rendering");
6      const viewData = {
7        title: "About Web app template"
8      };
9      response.render("about", viewData);
10   }
11 };
12
13 module.exports = about;
14
```

about.hbs

```
{{> menu id="about"}}

<div class="border p-2 my-2">
  <h3>Web app template</h3>
  <p>Content goes here...</p>
</div>
```

menu.hbs

```
<nav class="navbar navbar-expand-lg n
  <div class="container-fluid">
    ...
  </div>
</nav>
```

main.hbs

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{{title}}</title>
    <meta charset="UTF-8">
    <link href="https://cdn.jsdelivr.net/npm/bo
    integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU
    <script src="https://cdn.jsdelivr.net/npm/b
    integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wX
    <script src="https://ajax.googleapis.com/aj
  </head>
  <body>
    <section class="container">
      {{{body}}}
    </section>
  </body>
</html>
```

FAZIT

- Request landet beim Router, der die Anfrage an die entsprechende Methode eines Controller-Objekts weiterleitet
- Der Controller holt sich Daten (hier bereits im Controller vorhanden) und befüllt den View mit Daten
- Abschließend schickt der Controller diesen View als response an den Client zurück
- Ein Controller-Objekt kann über mehrere Methoden verfügen die alle thematisch zusammengehörige requests verarbeiten (z.B. alle requests an ein Dashboard, oder alle requests für die Verwaltung von Nutzerdaten (login, logout, register, ...))
- Backend (Controller) und View (Templates) arbeiten zusammen um eine response zu erzeugen
- Was noch fehlt: Controller holt sich Daten aus Models (später mehr dazu)