МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №2
По дисциплине «Web-программирование»
Шаблонизация веб-страниц приложения

Выполнили студент группы M33081
Аль Даббагх Харит Хуссейн

Проверил
Приискалов Роман Андреевич

САНКТ-ПЕТЕРБУРГ

2022

СОДЕРЖАНИЕ

HBS AND PARTIALS

hbs installation

hbs installation and registration was discussed in the previous lab.

Partials

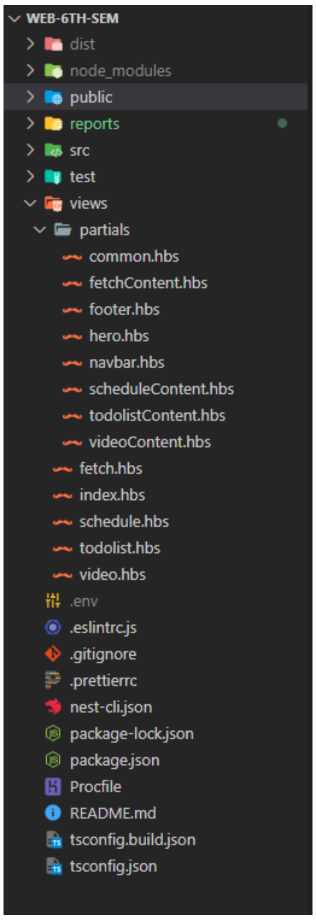
In handlebars, template reuse is done using partials. Partials maybe called directly from other partials. They can be separated in .hbs files.

During the lab work we had to separated our reusable components (From our static resources from the previous semester) into partials as shown in the image to the right.

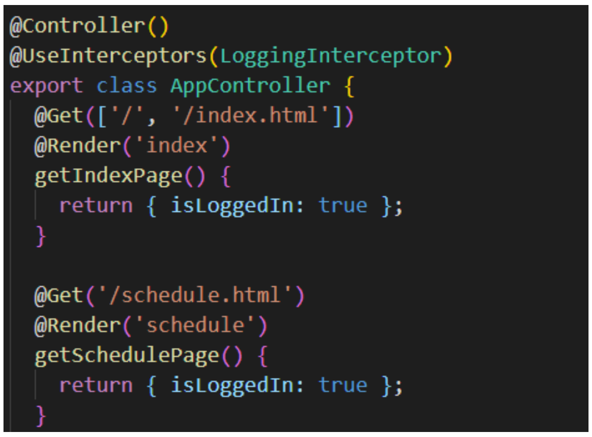
To call a reusable partial inside a template. We use {{> partial_name}}.

To let the different pages work, we need to add these routes in the controllers.

Example in the image below.



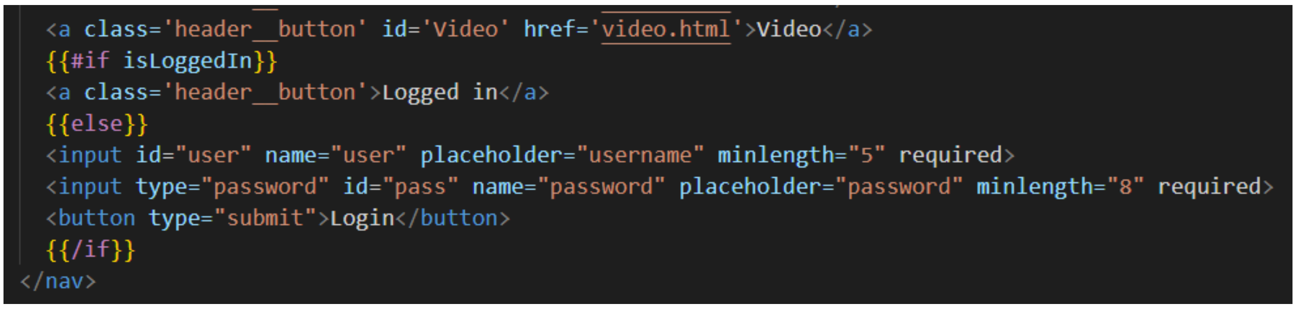
```
WEB-6TH-SEM
> dist
> node_modules
> public
> reports
> src
> test
v views
  v partials
      common.hbs
      fetchContent.hbs
      footer.hbs
      hero.hbs
      navbar.hbs
      scheduleContent.hbs
      todolistContent.hbs
      videoContent.hbs
    fetch.hbs
    index.hbs
    schedule.hbs
    todolist.hbs
    video.hbs
  .env
  .eslintrc.js
  .gitignore
  .prettierrc
  nest-cli.json
  package-lock.json
  package.json
  Procfile
  README.md
  tsconfig.build.json
  tsconfig.json
```



```
@Controller()
@UseInterceptors(LoggingInterceptor)
export class AppController {
  @Get(['/', '/index.html'])
  @Render('index')
  getIndexPage() {
    return { isLoggedIn: true };
  }

  @Get('/schedule.html')
  @Render('schedule')
  getSchedulePage() {
    return { isLoggedIn: true };
  }
```

Next we were required to add two states:



```
    <a class='header__button' id='Video' href='video.html'>Video</a>
    {{#if isLoggedIn}}
    <a class='header__button'>Logged in</a>
    {{else}}
    <input id="user" name="user" placeholder="username" minlength="5" required>
    <input type="password" id="pass" name="password" placeholder="password" minlength="8" required>
    <button type="submit">Login</button>
    {{/if}}
  </nav>
```

Handlebars is not a logical language, but it has some basic logical operators (helpers) as shown in the previous image.

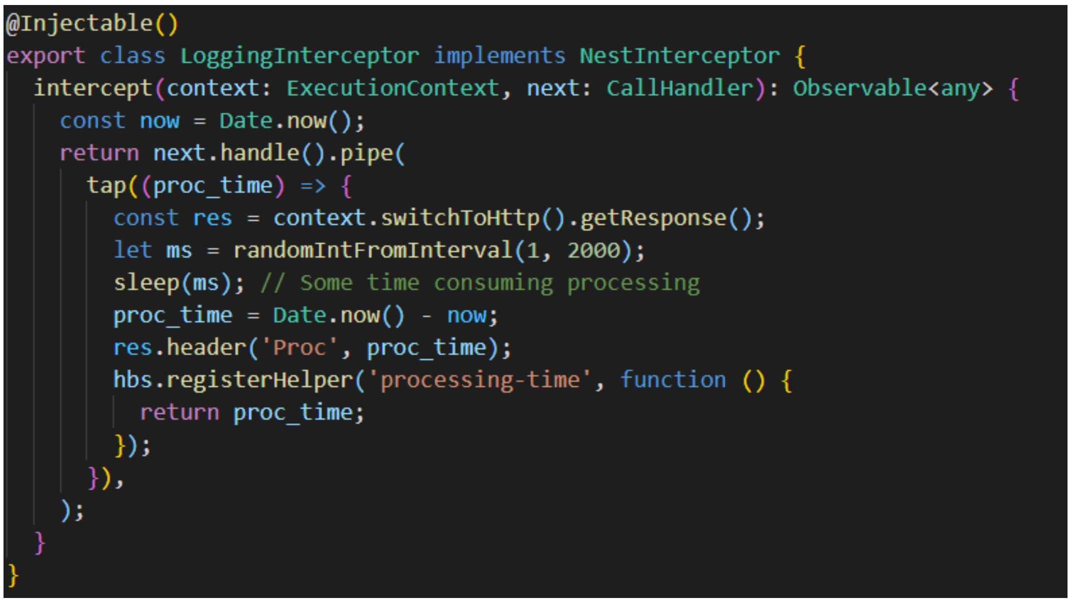
After that we were required to update the footer with the time required for the server to process the request. To do that we need to implement interceptors, which will intercept the response, add a header with the processing time, and update the HTML document before sending it to the client.

To quickly generate a new interceptor in nest, we can use its CLI:

```
nest g in logging
```

The g stands for “generate”, in stands for “interceptor” and logging is the name of the interceptor.

The following image shows the implementation for the interceptor:



```
@Injectable()
export class LoggingInterceptor implements NestInterceptor {
  intercept(context: ExecutionContext, next: CallHandler): Observable<any> {
    const now = Date.now();
    return next.handle().pipe(
      tap((proc_time) => {
        const res = context.switchToHttp().getResponse();
        let ms = randomIntFromInterval(1, 2000);
        sleep(ms); // Some time consuming processing
        proc_time = Date.now() - now;
        res.header('Proc', proc_time);
        hbs.registerHelper('processing-time', function () {
          return proc_time;
        });
      }),
    );
  }
}
```

In this method, a header is added to the response, plus hbs will register a new custom helper called ‘processing-time’ which will contain the processing time on the server side, which can then be injected in the footer of the page like so:

```
<div>
  <p class="footer__loadtime"></p><p class="footer__loadtime">{{processing-time}} ms (server)</p>
</div>
```

Keep in mind that the first <p> tag will be filled using a custom JS script from the previous semester.