Unit 2-12

Express를 통한 사이트 개선

UT-NodeJS / 04.14.2023

ut-nodejs.github.io

Contents / 내용

12. Express.js를 통한 사이트 개선

```
. <-- 터미널에서 본 루트 폴더의 트리구조 (p. 175-176)
|____ public
|  |____ css
|  |____ js
|  |____ img
|____ views
|  |____ layout.ejs
|  |____ index.ejs
|  |____ courses.ejs
|  |____ contact.ejs
|  |____ thanks.ejs
|  |____ error.ejs
|____ controllers
|  |____ homeController.js
|  |____ errorController.js
|____ main.js
|____ package.json <-- npm init으로 만드십시오
|____ package-lock.json <-- npm install로 만드십시오
```

12

캡스톤 프로젝트 2

Express.js를 통한 사이트 개선

p. 173-185

Express.js를 통한 사이트 개선

이 애플리케이션을 만들기 위해 다음과 같은 단계를 거친다.

1. 애플리케이션 package.json의 초기화 (**`npm init`**)
2. 프로젝트 디렉터리 구조 셋업 (p. 175-176)
3. main.js에서 메인 애플리케이션의 설정 (p. 176-177)
4. 라우트의 추가 (`courses`, `contact`, `thanks`) (p. 178)
5. 뷰로 라우팅 (p. 179-180)
    - **`layout.ejs`**
    - `index.ejs`
    - `courses.ejs`
    - `contact.ejs`
    - `thanks.ejs`
    - `error.ejs`
6. 정적 뷰 제공 (p. 181)
7. 뷰에 콘텐츠 전달 (p. 181-183)
8. 에러 처리 (p. 183-184)
9. 애플리케이션 실행

① 애플리케이션의 초기화

```
{
  "name": "4-express-capstone-soln",
  "version": "1.0.0",
  "description": "A website using Express",
  "main": "main.js",
  ▷ Debug
  "scripts": {
    "test": "jest",
    "start": "nodemon main.js"
  },
  "author": "Aaron Snowberger",
  "license": "ISC",
  "dependencies": {
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "express-ejs-layouts": "^2.5.1",
    "http-status-codes": "^2.2.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.21"
  }
}
```

터미널에서
프로젝트 디렉터리에서 .

```
npm init

npm install
http-status-codes
express
express-ejs-layouts
ejs

code .
```

② 애플리케이션 디렉터리 구조의 이해

```
. <-- 루트 폴더 (p. 175-176)
|____ public
|  |____ css
|  |____ js
|  |____ img
|____ views
|  |____ layout.ejs
|  |____ index.ejs
|  |____ courses.ejs
|  |____ contact.ejs
|  |____ thanks.ejs
|  |____ error.ejs
|____ controllers
|  |____ homeController.js
|  |____ errorController.js
|____ main.js
|____ package.json <-- npm init으로 만드십시오
|____ package-lock.json <-- npm install로 만드십시오
```

새로운 파일을 추가하기 전에 디렉터리 구조를 설정하려 한다. 최종 프로젝트 구조는 옆에와 같으며, 다음과 같은 항목을 추가하려 한다.

- HTML 페이지를 위한 views 폴더 *(포함)*
- 클라이 언트 사이드 에셋을 위한 css, js, images를 포함한 public 폴더 *(포함)*
- 라우팅 함수를 위한 controllers 폴더

public 폴더는 제공되지만 원하면 변경할 수 있습니다.
views 폴더는 포함된 HTML 파일을 EJS로 변환해야 합니다.

③ main.js에서 메인 애플리케이션의 설정

Listing 12.3 main.js에서 메인 애플리케이션의 설정

```
const express = require("express"),
  app = express();

app.set("port", process.env.PORT || 3000);

app.get("/", (req, res) => {
  res.send("Welcome to Confetti Cuisine!");
});

app.listen(app.get("port"), () => {
  console.log(
    `Server running at http://localhost:${app.get(
      "port"
    )}`
  );
});
```

express를 요청

express 애플리케이션의 인스턴스화

홈페이지를 위한 라우트 생성

3000번 포트로 리스닝 설정

Listing 12.4 main.js 상단에 body-parser의 추가

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

URL 인코드와 JSON 파라미터 처리를 위한 body-parser의 사용을 Express.js에 선언

④ 라우트의 추가 **(courses, contact, thanks)**

Listing 12.5 homeController.js에서 홈 컨트롤러로의 라우팅

homeController.js

```
exports.showCourses = (req, res) => {
  res.render("courses");
};
exports.showSignUp = (req, res) => {
  res.render("contact");
};
exports.postedContactForm = (req, res) => {
  res.render("thanks");
};
```

특정 라우트를 위한 콜백 함수 추가

main.js에서 Listing 12.6에서 보이는 것과 같이 이어지는 라우트를 추가하고 원래 홈페이지 라우트를 홈 컨트롤러를 사용하기 위해 수정했다.

Listing 12.6 main.js에서 각 페이지 및 요청 타입을 위한 라우트 추가

```
app.get("/courses", homeController.showCourses);
app.get("/contact", homeController.showSignUp);
app.post("/contact", homeController.postedSignUpForm)
```

코스 페이지, 연락처 페이지, 연락처 제출 양식을 위한 라우드의 추가

⑤ 뷰로 라우팅

우선 애플리케이션의 레이아웃 뷰를 생성한다. 이는 페이지 탐색과 페이지마다 어떻게 보여줘야 하는지 일반적인 사이트 구조를 다룬다. 레이아웃이 작동하려면 메인 애플리케이션 파일에 이를 include해야 하며, Listing 12.7과 같이 바로 Express.js 모듈의 초기화 아랫부분에 위치한다.

Listing 12.7 main.js에서 ejs 레이아웃 렌더링

```
const layouts = require("express-ejs-layouts");

app.set("view engine", "ejs");
app.use(layouts);
```

express-ejs-layout의 요청

ejs를 사용하기 위한 애플리케이션 세팅

layout 모듈 사용을 위한 애플리케이션 세팅

표 12.1 Confetti Cuisine 뷰

파일명	목적
layouts.ejs	애플리케이션의 메인 스타일링과 기본 탐색 기능 수행
index.ejs	홈페이지 콘텐츠를 만들어냄
courses.ejs	코스 콘텐츠의 디스플레이
contact.ejs	연락처 양식의 디스플레이
thanks.ejs	양식 제출 후 감사 메시지 디스플레이
error.ejs	페이지를 찾을 수 없는 경우 디스플레이

- 이 파일을 views 폴더에 있는 `layout.ejs` 파일에 추가한다. 이 파일에서의 키 부분은 `<%body%>` 부분이며, 이는 렌더링된 타깃 콘텐츠로 대체된다. 이어지는 뷰에서는 이 레이아웃을 사용해 (파일들 간의 코드 반복을 피하기 위해서) 시각적 일관성을 맞춘다. 이제 `view` 폴더 내에, `index.ejs, courses.ejs, contact.ejs, thanks.ejs` 그리고 `error.ejs`을 만들려고 한다.

⑥ 정적 뷰 제공

http를 이용한 첫 번째 애플리케이션에서, 정적 에셋을 제공한다는 것은 큰 장애물이었다. 각 새로운 에셋을 프로젝트 디렉토리에 추가할 때마다 새로운 라우트를 생성하고 이를 적절하게 처리하는 과정 이 필요했다. 다행히도 Express.js가 애플리케이션에서 제공해야 할 정적 에셋 처리를 깔끔하게 대신해준다. 정적 에셋을 제공하기 위해 `app.use(express.static("public"))` 을 애플리케이션 파일 내의 Express.js 초기화 부분 아래에 추가해 Express.js의 static 함수를사용한다.

main.js 위쪽

```
app.use(express.static("public"))
```

⑦ 뷰에 콘텐츠 전달

Listing 12.9 homeController.js에서 콘텐츠의 설정과 렌더링된 뷰에 전달

```
var courses = [
  {
    title: "Event Driven Cakes",
    cost: 50
  },
  {
    title: "Asynchronous Artichoke",
    cost: 25
  },
  {
    title: "Object Oriented Orange Juice",
    cost: 10
  }
];          ← 코스를 위한 배열 정의

exports.showCourses = (req, res) => {
  res.render("courses", {
    offeredCourses: courses
  });
};
```

코스 배열 데이터를 뷰로 전달

[노트] 뷰 안에서는 offeredCourses라는 변수명을 통해 배열에 접근할 수 있다 홈 컨트롤러 내에서 이 배열은 courses라는 이름으로 다룬다.

이 사이트는 때때로 코스 목록을 변경한다. 이 앱을 위해 JavaScript 객체로 제공 코스의 배열을 보여줄 필요가있다. 그러면 이 객체를 렌더링된 뷰로 보낼 수 있다. Listing 12.9와 같은 코드를 homeController.js에 추가한다. courses 변수를 JavaScript 배열 객체에 할당함으로써, 뷰에서 이 리스트를 쓸 수 있
고 특정 키를 타겟팅 할 수 있다. res.render 메소드는 courses 객체를 뷰로 전달할 수 있게 하며, 해당 페이지의 offeredCourses로서 이를 참조한다.

Listing 12.10 courses.ejs에서 뷰의 동적 콘텐츠의 출력

```
<h1>Our Courses</h1>
<% offeredCourses.forEach(course => { %>
  <h5> <%= course.title %> </h5>
  <span>$ <%= course.cost %> </span>
<% }); %>
```

뷰에서 코스 배열 접근을 위한 루프

⑧ 에러 처리

Listing 12.11 errorController.ejs에서의 에러 처리 라우트 추가

```
const httpStatus = require("http-status-codes");

exports.pageNotFoundError = (req, res) => {
  let errorCode = httpStatus.NOT_FOUND;
  res.status(errorCode);
  res.render("error");
};

exports.internalServerError = (error, req, res, next) => {
  let errorCode = httpStatus.INTERNAL_SERVER_ERROR;
  console.log(`ERROR occurred: ${error.stack}`);
  res.status(errorCode);
  res.send(`${errorCode} | Sorry, our application is taking a nap!`);
};
```

앞에서 처리되지 못한 모든 요청 처리

내부 서버 에러의 처리

애플리케이션은 정확한 콘텐츠와 좋은 사용자 경험을 제공해야 한다. 아직 이 애플리케이션은 사용자 추적 로직이 없다. 하지만 만일 애플리케이션에서 에러가 발생하면 에러 메시지를 애플리케이션 사용자들에게 보내려고 한다.

Listing 12.12 main.js에서 에러 처리 라우트

```
app.use(errorController.pageNotFoundError);
app.use(errorController.internalServerError);
```

미들웨어 함수로 에러 처리 추가

[노트] 라우트의 순서는 중요하다. 이 라우트는 기존에 존재하는 라우트 **아래에** 와야 한다. 기존 라우트는 범용으로 사용되는 것이며 하위 라우트에 오버라이딩되기 때문이다.

8 실행

index.ejs

CONFETTI CUISINE

Home Courses Contact

Welcome!

Please check out all the cool new courses by our innovative and creative chefs.

If you want to see a new course, click the courses page and explore!

When you see a course you like you can contact us to join! Soon you'll be cooking in your own home.

개인 정보

8 실행

courses.ejs

CONFETTI CUISINE

Home | Courses | Contact

Learn to cook cutting edge food

Our Courses

Event Driven Cakes

$ 50

Asynchronous Artichoke

$ 25

Object Oriented Orange Juice

$ 10

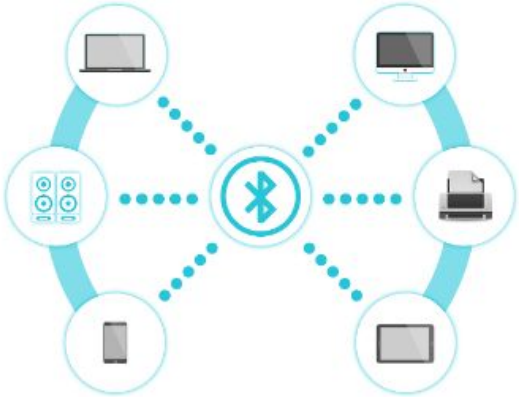
개인 정보

8 실행

contact.ejs

CONFETTI CUISINE

Home | Courses | Contact



Reach out to us!

Enter your email if you're interested to learn more:

Name

Email

submit

We'll get back to you soon!

개인 정보

8 실행

thanks.ejs

CONFETTI CUISINE

Home Courses Contact

Thank you for submitting!

Go to home!

개인 정보

8 실행

error.ejs

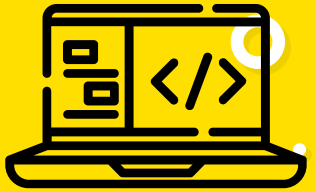
CONFETTI CUISINE

Home | Courses | Contact

Oops! Something went wrong, or the page you are looking for is not available.

Go to home!

개인 정보



Coding 과제!

캡스톤 프로젝트

Express.js를 통한 사이트 개선

p. 173-185

과제 타임!

한번 해 보자!~