

01. 백엔드 개발

백엔드 개발 기본 개념 다지기

Node.js는 무엇일까?

Node.js 설치하기

패키지 관리는 npm으로 해요

웹 개발 기본 용어

클라이언트와 서버

- 정보를 요청하는 컴퓨터 : 클라이언트
- 정보를 처리하는 컴퓨터 : 서버

서버를 만든다고요?

- 서버에서 어떤 서비스를 제공하는지에 따라 서버의 역할이 달라짐.
- 서버에서 서비스를 제공하려면 그에 맞는 프로그램을 실행해야 한다.
- 서버를 만든다 → 서버에서 실행할 프로그램을 만든다

(예) 웹 서버를 만든다는 뜻은 컴퓨터를 웹 서버로 동작하게 하는 프로그램을 만든다는 뜻

웹 개발 기본 용어

프론트엔드와 백엔드는 어떻게 다를까?

- 프론트엔드 개발은 사용자와 만나는 영역 → 개발의 초점이 '사용자'에 맞춰져 있다.
- 사용자가 쉽게 정보를 요청하고, 서버에서 받은 정보를 사용자가 보기 쉽게 만드는 것이 목표
- 백엔드 개발은 개발의 초점이 '자료'에 맞춰져 있다.
- 사용자가 보낸 요청을 분석하고 자료를 처리. 처리한 결과를 데이터베이스에 안전하게 저장하기도 하고 다시 사용자에게 넘겨주기도 한다.
- 웹 개발은 이렇게 두 개의 영역으로 나뉘지고 각 영역마다 사용하는 기술과 언어가 다르다.

HTTP 프로토콜

- 클라이언트와 서버 간에 자료를 주고 받기 위한 규칙
- 백엔드 개발을 위해서는 HTTP 프로토콜을 알아야 한다.

백엔드 개발 기초 지식

- **Node.js**

- Node.js는 언어가 아니라 자바스크립트라는 언어로 서버를 개발할 수 있도록 도와주는 도구
- 백엔드 개발에서 자바스크립트를 사용할 수 있게 환경을 만들어 주는 것

- **Express 프레임워크**

- 서버를 만들 때마다 반복되는 패턴과 복잡한 기능을 처리해 주는 함수를 제공한다.
- 더욱 빠르게 서버를 만들고 기능을 추가할 수 있다.
- 프레임워크를 사용하면 개발의 효율성뿐만 아니라 생산성도 높일 수 있다.

- **몽고DB**

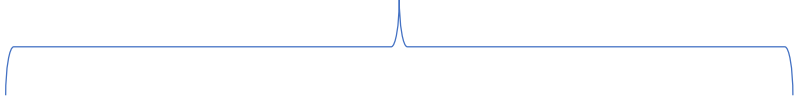
- 데이터베이스도 여러 형태. 프로젝트나 회사 방침에 따라 선택
- 여기에서는 몽고DB 사용

- **API 구축하기**

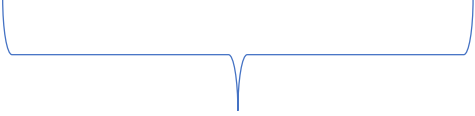
- API는 프로그램과 프로그램을 연결해 주는 프로그램.
- 클라이언트 프로그램 서버 프로그램 사이를 연결해 주는 역할.
- HTTP 프로토콜을 사용한 API를 RESTful API라고 한다.

Node.js의 정의

구글의 크롬 V8 엔진을 사용해서 웹 브라우저 밖에서
자바스크립트를 사용할 수 있도록 만들었다는 뜻



Node.js는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.



- 자바스크립트를 실행할 수 있는 환경
- Node.js가 V8 엔진을 포함하고 있기 때문에 웹 브라우저 밖에서도 자바스크립트를 실행할 수 있다

Node.js의 장점

- **하나의 언어로 프론트엔드와 백엔드를 개발할 수 있습니다**
 - 프론트엔드 개발에서는 HTML, CSS, 자바스크립트(리액트) 언어 사용
 - 백엔드 개발에서는 PHP, 루비, 파이썬, 자바스크립트(Node.js) 등 사용
 - 자바스크립트를 사용하면 프론트엔드와 백엔드 개발을 한꺼번에 할 수 있다.
- **개발자 커뮤니티가 크고 다양합니다**
 - 커뮤니티가 많다는 것은 여러 환경에서 사용자들끼리 피드백이 활발하고 지역별 또는 주제별로 다양한 논의가 이루어진다는 뜻
- **서드파티 모듈이 많습니다**
 - 서드파티 모듈이란 Node.js 제작 업체와 개발자 외에 다른 사람이 만들어 놓은 기능 묶음
 - 이미 다른 사람들이 만들어 놓은 모듈이 많으므로 가져다 사용하면 된다.

02. 자바스크립트 기초 문법과 모듈

자바스크립트 기초 문법

자바스크립트 비동기 처리

Node.js의 모듈이란

Node.js 코어 모듈

템플릿 리터럴(template literals)

- 기존의 연결 연산자(+) 대신 백틱(`) 기호를 사용해
문자열과 변수, 식을 섞어서 하나의 문자열을 만드는 표현 형식
- 어떤 방법을 사용하든 결과는 같지만 문자열과 변수를 연결해서 사용할 경우 템플릿 리터럴이 더 간편.

```
let num1 = 10;  
let num2 = 20;
```

// 연결 연산자를 사용할 경우

```
console.log(num1 + "과 " + num2 + "를 더하면 " + (num1 + num2) + "입니다.");
```

// 템플릿 리터럴을 사용할 경우

```
console.log(`${num1}과 ${num2}를 더하면 ${num1 + num2}입니다.`);
```

```
PS C:\Users\kyrie\Desktop\basics\02> node literals  
10과 20를 더하면 30입니다.  
10과 20를 더하면 30입니다.
```


여러 형태의 함수

1) 기본 방법 : 선언한 후 필요할 때 호출해서 사용

```
// 함수 선언
function greeting(name) {
  console.log(` ${name} 님, 안녕하세요?`);
}
// 함수 호출
greeting("홍길동");
```

매개변수(parameter) → name
인수(argument) → "홍길동"

```
PS C:\Users\kyrie\Desktop\basics\02> node function-1
홍길동 님, 안녕하세요?
```

2) 함수 표현식 : 이름없이 변수에 할당해서 사용

```
// 함수 선언
const greeting = function(name) {
  console.log(` ${name}님, 안녕하세요?`);
}
// 함수 호출
greeting("홍길동");
```

여러 형태의 함수

3) 즉시 실행 함수 : 선언과 동시에 실행

```
// 함수 선언과 호출을 동시에  
(function (a, b) {  
  console.log(`두 수의 합: ${a + b}`);  
})(100, 200); // 300
```

함수 선언

함수 실행

```
PS C:\Users\kyrie\Desktop\basics\02> node function-3  
두 수의 합: 300
```

4) 화살표 함수 : => 기호 사용

```
() => { ... }  
(매개변수) => { ... }
```

- 매개변수가 없다면 괄호를 비워 둠

```
// 함수 표현식  
let hi = function() {  
  return '안녕하세요?';  
}  
console.log(hi());
```

```
// 화살표 함수  
let hi = () => { return '안녕하세요?' };  
console.log(hi());
```

자바스크립트 비동기처리

동기 처리와 비동기 처리

- 프로그램은 기본적으로 코드가 작성된 순서대로 실행됨
- 중간에 시간 걸리는 작업이 있다면?
→ JS에서는 빨리 처리할 수 있는 것부터 실행하고 시간 걸리는 작업은 나중에 처리
- 시간 걸리는 작업 다음에 이어지는 작업이 있다면?
→ 시간걸린다고 무조건 뒤로 돌리면 안됨.
처리 순서를 조절할 수 있어야 함

자바스크립트 비동기처리

자바스크립트에서 비동기 처리할 때 사용하는 방법

방법	버전	기능
콜백 함수	기존부터 사용	함수 안에 또 다른 함수를 매개변수로 넘겨서 실행 순서를 제어합니다. 콜백 함수가 많아지면 가독성이 떨어질 수 있습니다.
프로미스	에크마스크립트 2015 부터 도입	프로미스 객체와 콜백 함수를 사용해서 실행 순서를 제어합니다.
async/await	에크마스크립트 2017 부터 도입	async와 await 예약어를 사용해서 실행 순서를 제어합니다.

CommonJS 모듈 시스템 vs ES 모듈 시스템

- CommonJS 모듈 시스템 : Node.js의 기본 모듈 시스템. (require, module.exports 사용)
- ES 모듈 시스템 : ECMAScript의 표준 모듈 시스템. (import, export 사용)

Callback example

The callback form takes a completion callback function as its last argument and invokes the operation asynchronously. The arguments passed to the completion callback depend on the method, but the first argument is always reserved for an exception. If the operation is completed successfully, then the first argument is `null` or `undefined`.

```
import { unlink } from 'node:fs';

unlink('/tmp/hello', (err) => {
  if (err) throw err;
  console.log('successfully deleted /tmp/hello');
});
```

CJS ☒ ESM

ES 모듈 시스템을 사용한 예제 코드

Callback example

The callback form takes a completion callback function as its last argument and invokes the operation asynchronously. The arguments passed to the completion callback depend on the method, but the first argument is always reserved for an exception. If the operation is completed successfully, then the first argument is `null` or `undefined`.

```
const { unlink } = require('node:fs');

unlink('/tmp/hello', (err) => {
  if (err) throw err;
  console.log('successfully deleted /tmp/hello');
});
```

CJS ☒ ESM

CommonJS 모듈 시스템을 사용한 예제 코드

노드 코어 모듈

- Node.js에 포함되어 있는 모듈

기능	모듈명	설명
파일 시스템	fs	파일이나 폴더에 접근할 수 있는 기능을 제공합니다. 예를 들어 파일 읽기/쓰기/삭제/이동/이름 변경이나 폴더 작업을 처리할 수 있습니다. 자세한 사용법은 ###쪽에서 설명합니다.
HTTP	http	HTTP 서버를 만들고 요청을 처리하는 기능을 제공합니다. 익스프레스 같은 프레임워크 없이 HTTP 서버를 만드는 방법은 ##쪽에서 설명합니다.
경로	path	파일 경로와 관련된 작업을 하는 기능을 제공합니다. 예를 들어 파일 경로를 지정하거나 상대 경로를 계산하는 작업을 할 수 있습니다. 자세한 사용법은 ###쪽에서 설명합니다.
스트림	streams	데이터 스트림을 처리하는 기능을 제공합니다. 예를 들어 파일이나 네트워크와 같은 스트림에서 데이터를 읽거나 쓰는 작업을 할 수 있습니다. 스트림이 무엇인지, streams 모듈을 사용하는 방법은 ###쪽에서 설명합니다.
암호화	crypto	암호화와 관련된 기능을 제공합니다. 해시 함수, 암호화 알고리즘, 암호화 및 복호화 등을 지원합니다. 해시 함수나 암호화, 복호화는 ###쪽에서 설명합니다.
운영체제	os	운영체제와 상호 작용하는 기능을 제공합니다. 예를 들어 운영체제의 정보를 알아내거나 시스템 리소스 정보를 확인할 수 있습니다.
유틸리티	util	다양한 유틸리티 함수를 제공합니다. 예를 들어 객체 상속, 비동기 작업을 프로미스로 변환하는 등의 작업을 할 수 있습니다.
이벤트	events	이벤트 기반 프로그래밍을 지원하는 기능을 제공합니다. 이벤트 생성, 등록, 처리 등을 할 수 있으며, 커스텀 이벤트를 사용하여 비동기 작업을 다룰 수 있습니다.

네트워크 기초 및 서버 만들기

HTTP 기초

HTTP 프로토콜

클라이언트와 서버 사이에 똑같이 인식하는 규칙

요청과 응답

HTTP 프로토콜을 이용해 자료를 요청하고 받을 때는 **요청**과 **응답**을 사용해 프로그래밍

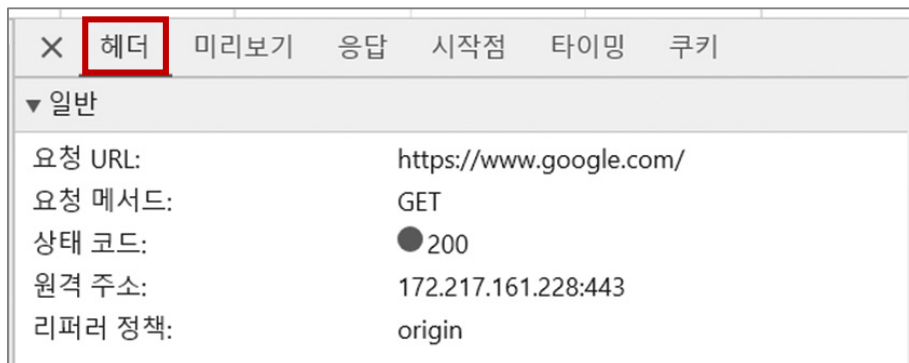


HTTP 기초

요청과 응답 살펴보기

- ① 웹 브라우저 창에서 구글 웹 사이트(www.google.com)로 접속
- ② 웹 개발자 도구 창에서 [네트워크] 탭 클릭 후 브라우저 새로 고침
- ③ 내려 받은 파일 중에서 하나를 선택
- ④ 응답 본문을 확인하려면 [응답] 탭 클릭

헤더 정보 살펴보기



- 요청 URL : 서버에게 정보를 보내달라고 요청하는 URL
- 요청 메서드 : 요청하는 정보를 어떻게 처리할지 지정

요청 메서드	설명
GET 메서드	서버에서 정보를 가져올 때 사용합니다. 예를 들어 구글 웹 사이트 URL을 입력해 서버로 보내거나 웹 사이트에 있는 링크를 클릭하면 GET 요청이 서버로 전송되고, 서버는 해당 URL의 문서를 응답으로 반환합니다.
POST 메서드	서버에 데이터를 저장할 때 사용합니다. 예를 들어 회원 가입을 하거나 로그인할 때 사용자가 입력한 정보는 POST 메서드를 사용해 서버로 넘겨줍니다.
PUT 메서드	서버에 있는 데이터를 수정(업데이트)할 때 사용합니다. 예를 들어 서버에 이미 저장되어 있는 사용자 정보에서 일부를 수정할 때 PUT 메서드를 사용해 서버로 보냅니다.
DELETE 메서드	서버에서 데이터를 삭제할 때 사용합니다. 예를 들어 블로그 글이나 파일을 삭제할 때 DELETE 메서드를 사용해 삭제할 정보를 서버로 전송합니다.

헤더 정보 살펴보기

- 상태 코드 : 요청의 성공 여부를 알려주는 코드

코드	메시지	설명
1xx	Informational	계속 처리 중
2xx	Successful	요청 성공
200	OK	요청이 성공적으로 처리되었습니다.
201	Created	요청이 성공적으로 처리되어 새로운 자료가 생성되었습니다.
204	No Content	요청이 성공적으로 처리되었지만 응답으로 반환할 내용이 없습니다.
3xx	Redirection	다른 위치로 이동
301	Moved Permanently	요청한 데이터가 새 URL로 옮겨졌습니다.
4xx	Client Error	클라이언트 오류
400	Bad Request	클라이언트 요청이 잘못되었거나 유효하지 않습니다.
401	Unauthorized	권한이 없어 거절되었지만 인증을 다시 시도할 수 있습니다.
403	Forbidden	권한이 없어 거절되었고 인증을 시도하면 계속 거절됩니다.
404	Not Found	해당 데이터를 찾을 수 없습니다.
5xx	Server Error	서버 오류
500	Internal Server Error	서버에 요청을 처리하는 동안 오류가 발생했습니다.
503	Service Unavailable	요청한 서비스를 이용할 수 없습니다.

- 원격 주소 : 인터넷에서 서버에 접속하는 실제 IP 주소와 포트 번호



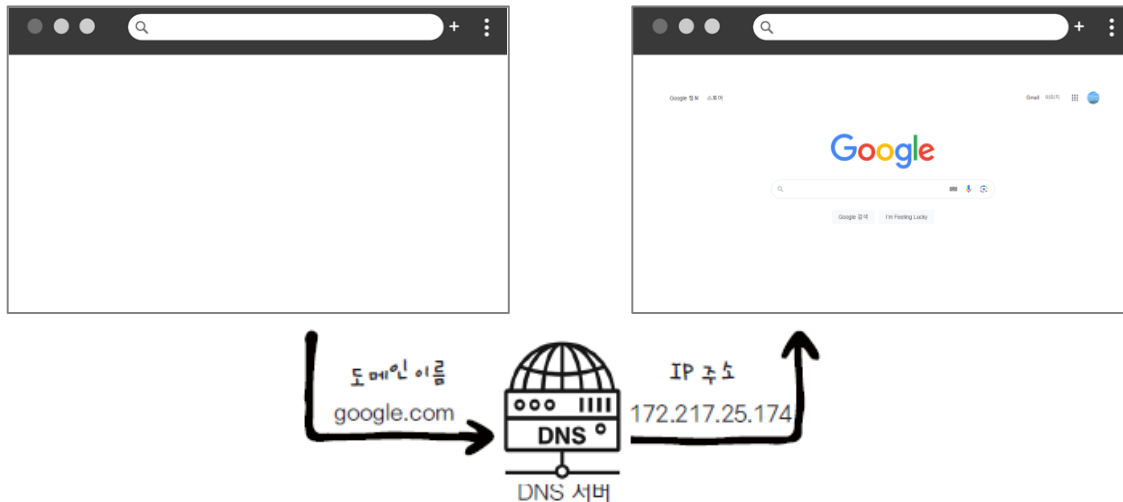
파일마다 IP 주소가 다르다면 파일들이 서로 다른 서버에서 왔다는 뜻

×	헤더	미리보기	응답	시작점	타이밍	쿠키
▼	일반					
요청 URL:	https://www.gstatic.com/inputtools/images/tia.png					
요청 메서드:	GET					
상태 코드:	● 200					
원격 주소:	142.250.206.227:443					
리퍼러 정책:	strict-origin-when-cross-origin					

×	헤더	미리보기	응답	시작점	타이밍	쿠키
▼	일반					
요청 URL:	https://www.google.com/tia/tia.png					
요청 메서드:	GET					
상태 코드:	● 200					
원격 주소:	142.250.76.132:443					
리퍼러 정책:	origin					

IP 주소

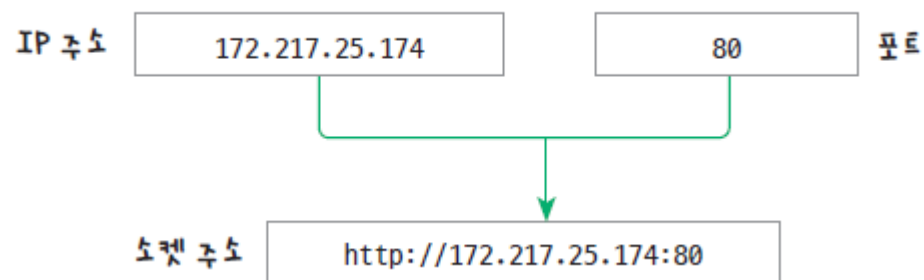
- IP 주소 : 인터넷에 연결된 수많은 컴퓨터를 구별하기 위한 주소
- IP 주소는 네 자리 또는 여섯 자리 숫자로 구성되어 있어 모든 사이트의 IP 주소를 기억하는 것은 거의 불가능
- 숫자 대신 google.com처럼 영문으로 된 주소를 사용하면 서버에서 자동으로 IP 주소로 변환해서 사용
 - 사용자가 기억하기 쉬운 사이트 주소 → 도메인 이름
 - 도메인 이름을 IP 주소로 변환하는 컴퓨터 → DNS 서버



포트

- 서버에는 여러 가지 프로그램이 실행 중
- 서버에 있는 프로그램 중에서 어떤 프로그램을 실행할지 지정하기 위해 포트 번호 사용
- 도메인 이름이나 IP 주소 뒤에 콜론(:) 기호와 함께 포트 번호를 적는다

(예) 172.217.25.174 서버에서 80번 포트에 접속한다면 → `http://172.217.25.174:80`



자주 사용하는 포트 번호

번호	기능
20, 21	FTP(파일 전송 프로토콜)
25	SMTP(메일 발송)
53	DNS 서버
80	웹(HTTP)
110	POP3(메일 수신)
443	HTTPS

라우팅

- 라우팅이란 클라이언트에서 들어오는 요청에 따라 그에 맞는 함수를 실행하는 것
- (예) nodejs.org 사이트에서
nodejs.org/en/about 로 요청하면 About Node.js 화면으로 이동
nodejs.org/en/download라는 URL로 요청하면 Downloads 화면으로 이동
- 라우팅을 이용하면 사용자가 입력하는 URL에 따라 다른 내용을 보여 줄 수 있음
- GET이나 POST, PUT, DELETE 같은 요청 메서드에 따라 처리할 함수를 다르게 연결할 수도 있음

익스프레스로 서버 만들기

왜 익스프레스인가

- HTTP 모듈을 사용해서 서버를 만들 수도 있지만
- 익스프레스에는 HTTP 모듈의 기능 외에도 다양한 기능이 포함되어 있음

기능	설명
라우팅	HTTP 모듈을 사용할 때는 if 문이나 switch 문으로 요청 메서드나 요청 URL에 따라 라우팅해야 했습니다. 하지만 익스프레스에서는 더욱 간편한 방법으로 라우팅할 수 있습니다. 라우팅은 06-2절에서 자세히 설명합니다.
미들웨어	익스프레스에는 ‘미들웨어’라는 개념이 있어서 요청과 응답 사이에서 여러 가지 기능을 실행할 수 있습니다. 이미 많은 사용자들이 미들웨어를 만들어서 패키지로 제공하므로 자주 사용하는 미들웨어는 따로 만들 필요 없이 가져와서 사용할 수 있습니다. 미들웨어는 07-1절에서 자세히 설명합니다.
템플릿 엔진	HTML 페이지는 기본적으로 정적이지만 서버와 함께 사용해서 동적인 HTML 페이지를 만들 수 있습니다. 애플리케이션에서 보이는 부분인 뷰 ^{view} 를 담당하죠. 익스프레스에서 동적인 화면을 구성하는 템플릿 엔진은 10-1절에서 자세히 설명합니다.
정적인 파일 지원	익스프레스에서 동적인 파일만 생성하는 것은 아닙니다. CSS 파일이나 JS 파일, 이미지처럼 정적인 파일을 쉽게 서비스할 수 있는 기능도 제공합니다. 정적인 파일을 사용하는 방법은 10-1절에서 자세히 설명합니다.

- 많은 사용자들이 선택한 프레임워크

익스프레스 응답 객체의 함수

함수	설명
res.download	파일을 내려받습니다.
res.end	응답 프로세스를 종료합니다.
res.json	JSON 응답을 전송합니다.
res.jsonp	JSONP 지원을 통해 JSON 응답을 전송합니다.
res.redirect	요청 경로를 재지정해서 강제 이동합니다.
res.render	뷰 템플릿을 화면에 렌더링합니다(10-2절에서 설명합니다.).
res.send	어떤 유형이든 res.send() 괄호 안의 내용을 전송합니다.
res.sendFile	지정한 경로의 파일을 읽어서 내용을 전송합니다.
res.sendStatus	상태 메시지와 함께 HTTP 상태 코드를 전송합니다.
res.status	응답의 상태 코드를 설정합니다.

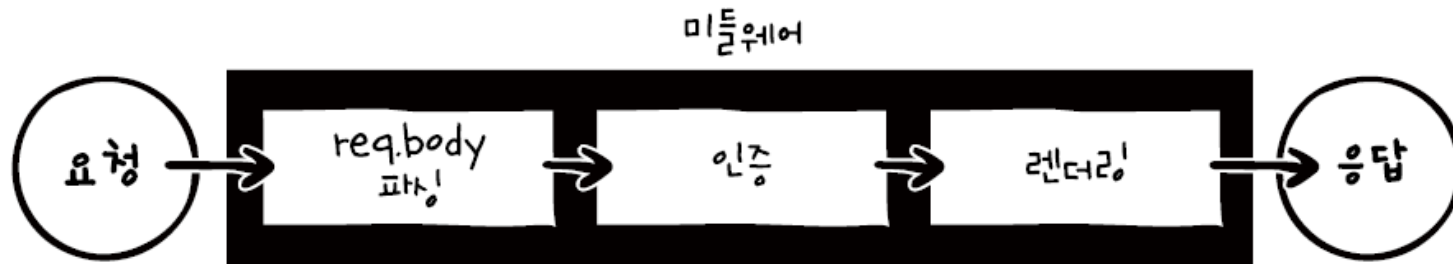
미들웨어

미들웨어란

요청과 응답 중간에 있으면서 요청을 처리하거나 원하는 형태로 응답을 수정하는 기능을 가진 함수

(예) 로그인 미들웨어

- ① 요청 안에 포함된 아이디와 비밀번호의 값을 애플리케이션에서 읽을 수 있는 형태로 변환(파싱이라고 함)
- ② 아이디와 비밀번호의 값을 사용해서 사용자 인증
- ③ 처리 결과를 다음 단계(예를 들어, 화면 렌더링)로 넘겨 줌



미들웨어의 역할

속성	설명
요청 전처리	요청이 서버에 도착하기 전에 실행하는 작업을 담당합니다. 예를 들어 사용자 인증이나 폼 내용 검증 등의 작업을 처리하죠.
라우팅 처리	지금까지 특정 URL로 들어온 요청을 미리 만들어 둔 함수(라우트 핸들러)로 연결했습니다. 이런 라우트 코드를 좀 더 읽기 쉽고 관리하기 쉽도록 모듈화하는 라우터 미들웨어도 있습니다.
응답 처리	서버에서 클라이언트로 응답을 보낼 때 자료를 적절한 형태로 변환하거나 오류를 처리하는 작업을 합니다.

라우터 미들웨어

① router 객체 만들기

```
const router = express.Router();
```

② router 객체를 사용해 라우트 코드 작성

③ app.js에서 router 객체를 미들웨어로 등록

```
app.use(router);
```

요청 경로별로 라우트하기

요청 URL	요청 방식	설명
/contacts	GET	전체 연락처 보기
/contacts	POST	새 연락처 추가하기
/contacts/id	GET	연락처 상세 보기
/contacts/id	PUT	연락처 수정하기
/contacts/id	DELETE	연락처 삭제하기

몽고DB 데이터베이스

데이터베이스의 종류

관계형 데이터베이스

- 저장할 자료의 구조를 행과 열로 구분해서 표 형태로 관리
- 표와 표를 연결해서 사용하기도 함
- SQL 언어를 사용하므로 SQL 데이터베이스라고도 함

contacts

id	name	email	phone
1	Kim	kim@aaa.bbb	12345
2	Lee	lee@aaa.bbb	67890
3	Park	park@aaa.bbb	98746
5	Choi	choi@aaa.bbb	56321

users

name	region	registered
Yang	서울	2023.01.05
Kang	부산	2022.11.23
Kim	인천	2023.04.15
Baek	수원	2000.12.15

관계형 데이터베이스

NoSQL 데이터베이스

- SQL 언어를 사용하지 않기 때문에 NoSQL 데이터베이스라고 함
- 문서 형태로 자료 저장
- 새로운 필드를 추가하기도 쉽고 기존 필드를 수정할 수도 있음
- 따로 데이터베이스 언어를 공부하지 않아도 됨
- 자료를 여러 컴퓨터에 나누어 저장할 수도 있음

데이터베이스의 종류

관계형 데이터베이스	MySQL	<ul style="list-style-type: none">• 관계형 데이터베이스에서 가장 많이 사용하는 데이터베이스입니다.• 도구와 라이브러리가 다양합니다.
	PostgreSQL	<ul style="list-style-type: none">• 오픈소스 관계형 데이터베이스입니다.• 다양한 기능을 지원하며 확장성이 뛰어나므로 대용량 자료 처리에 적합합니다.
	SQLite	<ul style="list-style-type: none">• 이름에서 알 수 있듯이 가벼운 SQL 데이터베이스입니다.• 파일 하나에 모든 데이터를 저장하므로 설정하기도 쉽고 규모가 작은 프로젝트에 적합합니다.• 모바일 애플리케이션 등에서 자주 사용합니다.
NoSQL 데이터베이스	몽고DB (MongoDB)	<ul style="list-style-type: none">• JSON 형식으로 문서에 자료를 저장하는 데이터베이스입니다.• 이미 익숙한 JSON 형식을 사용하고 자료 구조를 쉽게 변경할 수 있습니다.
	레디스(Redis)	<ul style="list-style-type: none">• 키-값 형식으로 자료를 저장하는 데이터베이스입니다.• 자료 구조가 간단해서 세션 관리나 실시간 분석 등에서 자주 사용합니다.
	Neo4j	<ul style="list-style-type: none">• 자료를 노드와 관계 형식으로 저장하는 데이터베이스입니다.• 복잡한 관계를 다루는 데 특화되어 있어서 소셜 네트워크나 추천 시스템 등에서 사용합니다.

몽고DB란

- NoSQL 데이터베이스의 한 종류
- JSON 형식으로 자료 저장
- 서버에 데이터베이스를 만들 수도 있고 클라우드에서 데이터베이스를 사용할 수도 있음
→ 몽고DB 아틀라스

CRUD API 작성하기

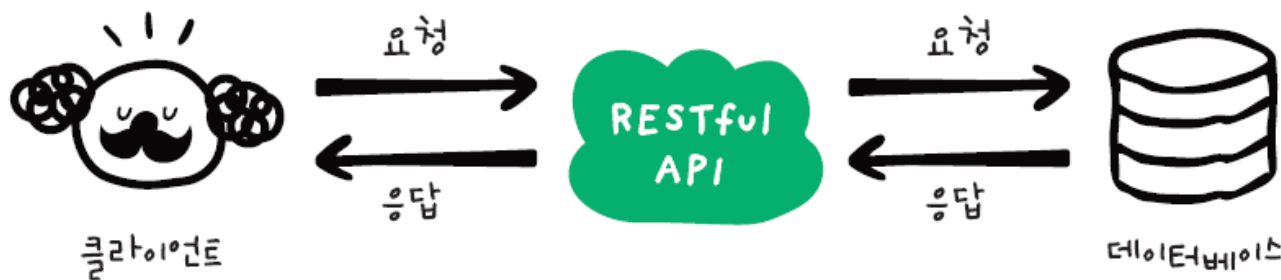
RESTful API

REST

- HTTP 프로토콜을 활용해서 자료를 주고받기 위해 약속된 구조
- representational state transfer의 줄임말. 여기에서 representational state는 데이터의 현재 상태를 볼 수 있게 나타낸 것
- (예) 온라인 쇼핑몰에서 장바구니에 어떤 상품을 담았는지 보여 주는 것이 상태를 나타내는 것
- 데이터 상태를 주고받을 때 사용하는 것이 REST

RESTful API

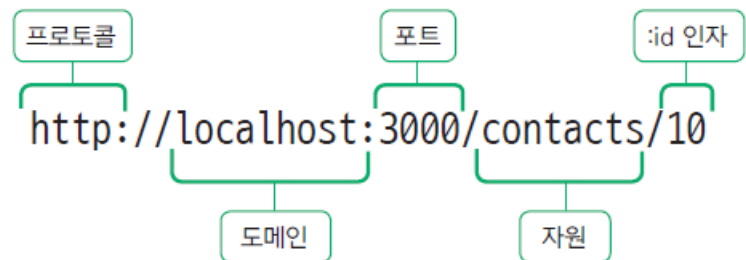
- REST를 잘 지켜서 개발한 API
- REST API라고도 함



RESTful API

URI로 자원 요청

- API에서 주고 받는 자원의 위치는 URI를 사용해 표현
- (예) localhost:3000/contacts/10



RESTful API

HTTP 요청 방식과 역할

HTTP 요청 방식	역할	설명
POST	Create	자원을 새로 만듭니다.
GET	Read	자원을 가져옵니다.
PUT	Update	자원을 수정합니다.
DELETE	Delete	자원을 삭제합니다.

C	————	Create	————	POST
R	————	Read	————	GET
U	————	Update	————	PUT
D	————	Delete	————	DELETE

MVC 패턴

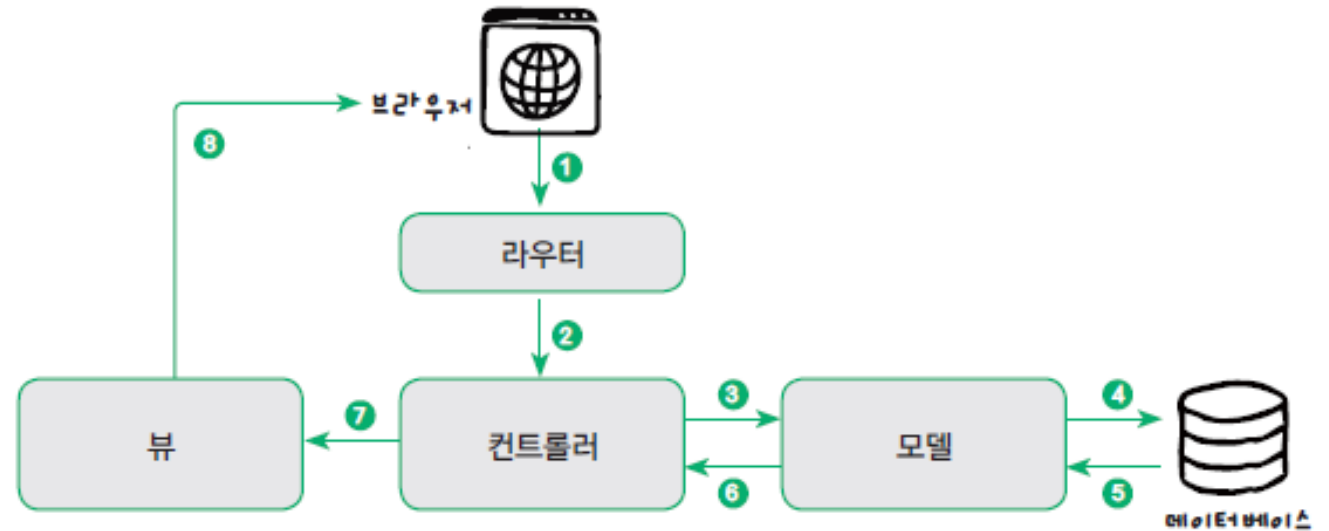
- 소프트웨어의 코드는 수천 수만 줄의 코드로 이루어짐 → 효율적으로 구성하려면?
- 코드의 기능이나 역할에 따라 여러 파일로 나눈 뒤 연결해서 사용 (디자인 패턴)
- MVC 패턴은 model, view, controller의 앞자를 따서 이름을 붙인 패턴

영역	설명
모델	<ul style="list-style-type: none">• 애플리케이션에서 처리할 대상입니다.• W데이터베이스를 통해 자료를 저장하거나 검색, 수정하는 함수들이 모델에 해당합니다.• 사용자에게 어떻게 보일지는 신경 쓰지 않고 처리할 대상에 집중합니다.
뷰	<ul style="list-style-type: none">• 컨트롤러나 모델의 처리 결과를 시각적으로 보여 줍니다.• 흔히 사이트나 애플리케이션에 표시되는 화면을 만듭니다.• 서버에서 가져온 동적 자료를 표시하므로 템플릿 형태로 처리합니다.
컨트롤러	<ul style="list-style-type: none">• 모델과 뷰 중간에 위치하면서 요청에 따라 모델이나 뷰를 수정하는 역할을 합니다.• Node.js에서 작성하는 라우트 코드가 컨트롤러에 해당합니다.• 코드를 가장 많이 작성하는 부분입니다. 그래서 이 책에서는 라우트 코드에서 함수 부분만 분리해서 작성합니다.

MVC 패턴

라우터 따로 사용하기

- 뷰와 컨트롤러 중간에 라우터 사용
- 나중에 라우트 코드는 건드리지 않고 함수 부분만 수정 가능



- ① 브라우저에서 모든 연락처 정보를 보여 달라고 요청합니다.
- ② 요청 정보는 라우터를 통해 컨트롤러로 연결됩니다.
- ③ 데이터베이스 정보에 접근해야 하므로 컨트롤러에서 모델로 다시 요청합니다.
- ④ 모델은 컨트롤러에게 받은 정보를 사용해서 데이터베이스에서 자료를 조회합니다.
- ⑤ 데이터베이스에서 찾은 정보를 모델로 넘겨줍니다.
- ⑥ 모델은 데이터베이스에서 받은 정보를 컨트롤러로 넘겨줍니다.
- ⑦ 컨트롤러는 모델에게서 받은 정보를 뷰에게 넘겨줍니다.
- ⑧ 뷰에서 지정한 형식대로 최종 결과를 브라우저 화면에 표시합니다.

템플릿 엔진으로 인터페이스 만들기

템플릿 엔진이란

- 사용자 동작(요청)에 따라 내용이 달라지는 동적인 콘텐츠를 가져와서 보여주려면 템플릿 엔진 필요
→ 템플릿을 만들어 놓고 데이터베이스에서 내용을 가져와 채워 넣으면 됨

템플릿 파일 : 데이터베이스에서 가져온 데이터 중 어떤 값을 어느 위치에 넣을지 미리 만들어 놓은 틀

템플릿 엔진 : 템플릿 파일을 만들고 데이터베이스에서 가져온 동적인 데이터를 템플릿 파일에 연결하는 역할

뷰 엔진 설정하기

노드에서 앞으로 어떤 템플릿 엔진을 사용할 것인지 알려주는 과정 필요

```
app.set(키, 값)
```

view engine

뷰에서 사용할 템플릿 엔진을 설정.

(예) EJS 엔진을 사용한다면 `app.set("view engine", "ejs")`

EJS 엔진의 기본 사용법

템플릿 파일에서 동적인 콘텐츠 처리하기

```
<%= 변수 %>  
<% 자바스크립트 코드 %>  
<%- HTML 코드 %>
```

(예) heading 변수의 값 표시

```
<%= heading %>
```

(예) 자바스크립트 코드 작성

```
<% if (age > 10) { %>  
  <p>나이 : <%= age %></p>  
<% } else { %>  
  <p>나이가 너무 어립니다.</p>  
<% } %>
```

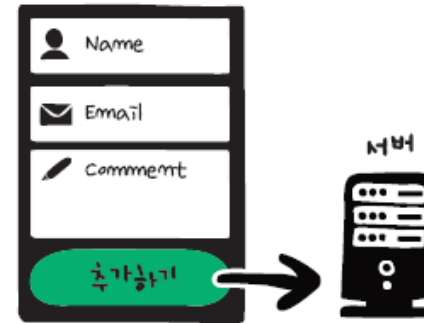
(예) HTML 코드 삽입

```
<%- include('include/header') %>
```

폼에서 라우트 처리하기

폼, 기억해 두세요

애플리케이션에서 로그인하거나 물건을 구입할 때
요청 정보는 모두 폼을 통해 서버로 보냄
→ 사용자 입력을 받는 폼이 필요하다



폼의 입력 필드에 name 속성이 있는지 확인

- 서버에서 폼 필드를 인식하려면 name 값 사용

```
<input type="text" class="form-control" name="name" id="name" placeholder="홍길동">  
<input type="text" class="form-control" name="email" id="email" placeholder=  
"hong@abc.def">  
<input type="text" class="form-control" name="phone" id="phone" placeholder="123-4567-  
8901">
```

폼의 요청 방식은 GET과 POST 뿐

- API에는 GET, POST, PUT, DELETE 등 여러 요청 방식이 존재
- GET과 POST 이외의 다른 요청 방식을 처리하려면 AJAX를 사용하거나 method-override 같은 모듈 사용

express.urlencoded 함수 이해하기

URL 인코딩 : URL에 입력된 내용을 서버 전송이 가능한 아스키 문자로 바꿔주는 것

파싱 : 인코딩된 내용을 애플리케이션 코드에서 사용하기 위해 풀어주는 것

express.urlencoded 함수를 사용한다. 파싱된 자료는 req.body에 저장됨.

express.urlencoded 함수에서 사용하는 extended 옵션

옵션	파싱에 사용하는 모듈
extended: false	<ul style="list-style-type: none">• Node.js의 querystring 모듈• 간단한 문자열이나 단순 배열을 파싱
extended: true	<ul style="list-style-type: none">• 익스프레스의 qs 모듈• 복잡한 객체나 배열까지 파싱

비밀번호 암호화하기 – bcrypt 모듈

해시 함수란

- 입력값을 받아서 또 다른 값을 반환하는 함수. 반환하는 값을 해시(hash)라고 함
- 해시 함수는 일방향 함수 – 해시만 보고 원래 값을 복원하는게 거의 불가능

bcrypt 모듈을 사용해 해시하기

기본형

`bcrypt.hash(data, saltRounds, callback)`

해시하려는 값

해시를 반복할 횟수

입력값을 해시한 후에 실행할 함수

(예)

```
const password = "1234";
bcrypt.hash(password, 10, (err, hash) => {
  try {
    // 해시화된 비밀번호(hash)를 데이터베이스에 저장하거나 다른 처리
  } catch (error) {
    // 오류 처리
  }
});
```

비밀번호 암호화하기 – bcrypt 모듈

비밀번호 확인하기

비밀번호를 비교할 때는 사용자가 입력한 비밀번호를 해시한 후 데이터베이스에 있는 해시값과 같은지 확인



기본형

```
bcrypt.compare(data, encrypted, callback)
```

비교할 입력값

비교할 대상

값 2개를 비교한 후에 실행할 함수

JWT(JSON Web Token)

JWT 토큰의 기본 형태



영역	속성
헤더Header	토큰의 알고리즘과 유형이 담겨 있습니다. 헤더에 있는 각각의 필드를 헤더 파라미터라고 합니다. <ul style="list-style-type: none">• "alg" 파라미터: 토큰에 사용한 서명 알고리즘• "typ" 파라미터: 토큰의 유형
페이로드Payload	사용자 인증 정보가 담겨 있습니다. 페이로드에 있는 각각의 필드를 클레임claim이라고 합니다. <ul style="list-style-type: none">• "sub" 클레임: 토큰 제목• "name" 클레임: 사용자 이름• "iat" 클레임: 토큰 발급 시간
서명Verify Signature	헤더와 페이로드 뒤에 붙이는 비밀 키입니다. 이 비밀키는 외부로 공개하면 안 되므로 .env 파일처럼 서버의 안전한 곳에 저장해 두고 사용합니다.

JWT 토큰의 사용자인증방법

1. 서버에서 토큰 만들기 – `jwt.sign` 함수

```
jwt.sign(페이로드, 비밀키, [옵션, 콜백])
```

2. 클라이언트에서 토큰 전송하기
3. 서버에서 토큰 검증하기

```
jwt.decode(토큰, 비밀키, [옵션])  
jwt.verify(토큰, 비밀키, [옵션])
```