

Next.js

REST API 서버 구축

<https://nextjs.org/>

<https://nextjs.org/showcase>



프레임워크 vs 라이브러리

프레임워크는 **기반구조** vs 라이브러리는 개발 편의 **도구들**

제어 주도권 **프레임워크**가 가짐 vs 제어 주도권 **사용자**가 가짐

프레임워크는 **기계**(like 굴삭기) vs 라이브러리는 **공구**(like 펜치니퍼)

프레임워크 자유도 상대적으로 **작음** vs 라이브러리 자유도 상대적으로 **큼**

* 프레임워크 장점

개인이 해야 할 **고민**들을 프레임워크 개발자가 **미리**하고 **반영**

특정 **디자인 패턴**이나 동작과 기능들을 위한 **정의**와 **방식**을 정리해 둬

여러 개발자가 함께 **협업**할 때 기준점이 됨

Next.js 란

Next.js는 **Vercel**에서 개발한 React 기반의 풀스택 웹 프레임워크입니다.

서버 사이드 렌더링(SSR), 정적 사이트 생성(SSG), 그리고 **API Routes**를 통한 백엔드 기능을 제공합니다.

NEXT.js Framework



React

Frontend Library
(Component Based)



Node.js

Backend Runtime
(Server Environment)



빌드 도구

Webpack / Turbo
(Bundling)

- > **파일 기반 라우팅**

(File-based Routing)

- > **API Routes**

(Serverless Functions)

- > **이미지 최적화**

(Image Optimization)

- > **서버 사이드 렌더링**

(SSR)

- > **정적 사이트 생성**

(SSG)

- > **자동 코드 분할**

(Code Splitting)

NextJS 프로젝트 생성하기

```
pnpm create next-app nextjs-basic ₩  
  --typescript ₩  
  --eslint ₩  
  --app ₩  
  --src-dir ₩  
  --import-alias "@/*" ₩  
  --no-tailwind ₩  
  --yes
```

옵션	설명
nextjs-basic	프로젝트 이름 (소문자만 가능)
--typescript	TypeScript 사용
--eslint	ESLint 설정 포함
--app	App Router 사용 (기본값)
--src-dir	src/ 디렉토리 사용
--import-alias "@/*"	경로 별칭 설정 (@/lib/data 등)
--no-tailwind	Tailwind CSS 제외
--yes	모든 프롬프트 자동 승인

App Router와 Route Handlers

Route Handlers

Route Handlers는 App Router에서 **API 엔드포인트**를 정의하는 방법입니다.

파일: /app/api/todos/route.ts

URL: GET http://localhost:3000/api/todos

파일: /app/api/todos/[id]/route.ts

URL: GET http://localhost:3000/api/todos/123

// route.ts에서 export 가능한 함수들

```
export async function GET(request: Request) { }
```

```
export async function POST(request: Request) { }
```

```
export async function PUT(request: Request) { }
```

```
export async function PATCH(request: Request) { }
```

```
export async function DELETE(request: Request) { }
```

```
export async function HEAD(request: Request) { }
```

```
export async function OPTIONS(request: Request) { }
```

라우팅 규칙 : 파일 기반 라우팅 (File-based Routing)

폴더 구조 URL

`app/api/route.ts` → `/api`

`app/api/todos/route.ts` → `/api/todos`

`app/api/todos/[id]/route.ts` → `/api/todos/:id`

`app/api/users/[...slug]/route.ts` → `/api/users/*`

폴더 구조만으로 별도의 라우팅 설정 없이 URL이 자동으로 매핑됨!

RESTful 설계 패턴

컬렉션 (Collection) /api/todos

- GET → 목록 조회
- POST → 새 리소스 생성

리소스 (Resource) /api/todos/:id

- GET → 단일 조회
- PUT → 전체 수정
- PATCH → 부분 수정
- DELETE → 삭제

NextRequest & NextResponse

NextRequest

NextRequest는 들어오는 HTTP 요청을 나타냄.

next/server에서 가져올 수 있으며, **Middleware**와 **Route Handlers** (App Router의 API)에서 주로 사용됩니다.

기본적인 Request 메서드(json(), formData(), text() 등)를 모두 사용할 수 있음.

NextResponse

NextResponse는 서버에서 클라이언트로 보내는 응답을 생성.

next/server에서 가져오며, NextRequest와 짝을 이뤄 사용됨.

Next.js REST API 서버 실습

1. 타입 정의 확인 (`src/types/todo.ts`)

2. 데이터 레이어 확인 (`src/lib/data.ts`)

: 메모리 기반 데이터 저장소 (서버 재시작 시 초기화) → 데이터베이스 역할

3. GET/POST 핸들러 구현

– 컬렉션 엔드포인트 (`/api/todos`) : `src/app/api/todos/route.ts`

`src/app/api/todos/`

—— route.ts	← 컬렉션 엔드포인트
	GET /api/todos (전체 조회)
	POST /api/todos (생성)
—— [id]/	
—— route.ts	← 리소스 엔드포인트
	GET /api/todos/:id (단일 조회)
	PUT /api/todos/:id (수정)
	DELETE /api/todos/:id (삭제)