

시작하기 및 CLI

Feathers.js는 실시간(WebSockets)과 REST API를
동시 지원하는 데이터 지향 Node.js/TypeScript 프레임워크입니다.

- 프로젝트 생성: `npm create @feathersjs/cli my-app`
- 서비스 생성: `npx feathers generate service`
- 인증 설정: `npx feathers generate authentication`
- 훅(Hook) 생성: `npx feathers generate hook`
- 개발 서버 실행: `npm run dev / npm run compile`

서비스 아키텍처 (Services)

서비스는 Feathers의 핵심 단위로, 표준 CRUD 메서드를 제공합니다.

서비스 기본 인터페이스

```
import type { Params, Id } from
 '@feathersjs/feathers'

export class MessageService {
  async find(params?: Params):
  Promise<Message[]> { ... }
  async get(id: Id, params?: Params):
  Promise<Message> { ... }
  async create(data: Partial<Message>,
  params?: Params): Promise<Message>
  { ... }
  async update(id: Id, data: Message,
  params?: Params): Promise<Message>
  { ... }
  async patch(id: Id, data:
  Partial<Message>, params?: Params):
  Promise<Message> { ... }
  async remove(id: Id, params?: Params):
  Promise<Message> { ... }
}

// 서비스 등록
app.use('messages', new
MessageService())

서비스 클라이언트 호출
const messageService =
app.service('messages')
const messages = await
messageService.find({ query: { read:
```

```
false } })
const newMsg = await
messageService.create({ text:
'Hello!' })
```

훅 (Hooks Architecture)

훅은 서비스 메서드 실행 전(before), 후(after), 에러 발생 시(error), 또는 메서드를 감싸서(around) 실행되는 미들웨어입니다.

Hook 등록 및 컨텍스트

```
import type { HookContext,
NextFunction } from '@feathersjs/
feathers'

// Around Hook 예시 (실행 시간 측정 및 로깅)
const logRuntime = async (context:
HookContext, next: NextFunction) => {
  const start = Date.now()
  await next() // 서비스 메서드 실행
  console.log(`${context.path}:
${context.method} took ${Date.now() -
start}ms`)
}

// 서비스에 훅 연결
app.service('messages').hooks({
  around: {
    all: [logRuntime],
  },
  before: {
    create: [
      async (context) => {
        context.data.createdAt = new
Date().toISOString()
      },
    ],
  },
})
```

실시간 이벤트 및 채널 (Real-time Channels)

Socket.io를 통해 실시간 데이터 동기화를 제공합니다.

```
// 연결 이벤트 및 채널 할당
app.on('connection', (connection) => {
```

```
app.channel('anonymous').join(connection)
})
```

```
// 인증된 사용자를 채널에 추가
app.on('login', (authResult,
{ connection }) => {
  if (connection) {
```

```
app.channel('authenticated').join(connection)
}
})
```

```
// 메시지 서비스 이벤트 브로드캐스트 설정
app.service('messages').publish('created',
(data, context) => {
  return app.channel('authenticated')
})
```

인증 (Authentication)

JWT, Local(비밀번호), OAuth 전략을 기본 지원합니다.

```
import { authenticate } from
 '@feathersjs/authentication'

// 특정 서비스 전체에 인증 훅 적용
app.service('messages').hooks({
  before: {
    all: [authenticate('jwt')],
  },
})
```

데이터베이스 어댑터

- `@feathersjs/knex`: PostgreSQL, MySQL, SQLite SQL 지원
- `@feathersjs/mongodb`: MongoDB 지원
- `@feathersjs/memory`: 메모리 기반 인메모리 스토어 (테스트용)

내장 쿼리 문법

- `$limit`, `$skip`: 페이지네이션
- `$sort`: 정렬 (`{ createdAt: -1 }`)
- `$select`: 특정 필드만 선택
- `$in`, `$nin`: 목록 포함 여부
- `$lt`, `$lte`, `$gt`, `$gte`, `$ne`: 비교 연산자