

Feathers.js Cheat Sheet

Getting Started & CLI

Feathers.js is a lightweight data-driven framework for building real-time applications and REST APIs in Node.js and TypeScript.

- **Create Project:** `npm create @feathersjs/cli my-app`
- **Generate Service:** `npx feathers generate service`
- **Generate Authentication:** `npx feathers generate authentication`
- **Generate Hook:** `npx feathers generate hook`
- **Start Dev Server:** `npm run dev / npm run compile`

Services Architecture

Services are the core building blocks in Feathers, providing standard CRUD methods.

Standard Service Interface

```
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> { ... }
}

// Register service
app.use('messages', new
MessageService())
```

Client / Service Usage

```
const messageService =
app.service('messages')
const messages = await
messageService.find({ query: { read:
false } })
const newMsg = await
messageService.create({ text:
'Hello!' })
```

Hooks Architecture

Hooks are middleware functions executed before, after, error, or wrapping (around) service methods.

Hook Registration & Context

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

// Around Hook example (Execution timing
& logging)
const logRuntime = async (context:
HookContext, next: NextFunction) => {
  const start = Date.now()
  await next() // Call service method
  console.log(`${context.path}:
${context.method} took ${Date.now() -
start}ms`)
}

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

Real-time Events & Channels

Real-time pub/sub synchronization powered by WebSockets / Socket.io.

```
// Connection event & default channel
app.on('connection', (connection) => {

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

// Join authenticated channel on login
app.on('login', (authResult,
{ connection }) => {
  if (connection) {

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

// Publish service events to channel
app.service('messages').publish('created',
(data, context) => {
  return app.channel('authenticated')
})

Authentication
Built-in JWT, Local (password), and OAuth authentication strategies.
import { authenticate } from '@feathersjs/authentication'
```

```
// Guard service with JWT authentication
app.service('messages').hooks({
  before: {
    all: [authenticate('jwt')],
  },
})
```

Database Adapters

- `@feathersjs/knex`: SQL databases (PostgreSQL, MySQL, SQLite)
- `@feathersjs/mongodb`: MongoDB support
- `@feathersjs/memory`: In-memory data store for tests

Built-in Query Syntax

- `$limit`, `$skip`: Pagination controls