

Developing Microservices with Node.js

Table of contents

Chapter 1: Microservices Architecture

- Need for microservices
 - Monolithic software
 - Microservices in the real world
 - Microservice-oriented architectures
 - How is it better?
 - Shortcomings
- Key design principles
 - Business units, no components
 - Smart services, dumb communication pipes
 - Decentralization
 - Technology alignment
 - How small is too small?
- Key benefits
 - Resilience
 - Scalability
 - Technology heterogeneity
 - Replaceability
 - Independence
 - Why is replaceability important?
 - Easy to deploy
- SOA versus microservices
- Why Node.js?
 - API aggregation
 - The future of Node.js

Chapter 2: Microservices in Node.js – Seneca and PM2 Alternatives

- Need for Node.js
 - Installing Node.js, npm, Seneca, and PM2
 - Learning npm
 - Our first program – Hello World
 - Node.js threading model
 - Modular organization best practices
 - Javascript
 - SOLID design principles
- Seneca – a microservices framework
- Inversion of control done right
- Pattern matching in Seneca
 - Patrun – a pattern-matching library
 - Reusing patterns

- Writing plugins
- Web server integration
- PM2 – a task runner for Node.js
 - Single-threaded applications and exceptions
 - Using PM2 – the industry-standard task runner

Chapter 3: From the Monolith to Microservices

- First, there was the monolith
 - How to tackle organic growth?
 - How abstract is too abstract?
- Then the microservices appeared
 - Disadvantages
 - Splitting the monolith
 - Problems splitting the monolith – it is all about the data
- Organizational alignment

Chapter 4: Writing Your First Microservice in Node.js

- Micromerice – the big picture
- Product Manager – the two-faced core
 - Fetching products
 - Fetching by category
 - Fetching by ID
 - Adding a product
 - Removing a product
 - Editing a product
 - Wiring everything up
 - Integrating with Express – how to create a REST API
- The e-mailer – a common problem
 - How to send e-mails
 - Defning the interface
 - Setting up Mandrill
 - Hands on – integrating Mandrill in your microservice
 - The fallback strategy
- The order manager
 - Defning the microservice – how to gather non-local data
 - The order manager – the code
 - Calling remote services
 - Resilience over perfection
- The UI – API aggregation
 - Need for frontend microservice
 - The code
 - Service degradation – when the failure is not a disaster
 - Circuit breakers
 - Seneca – a simple puzzle that makes our lives easier
 - Seneca and promises
- Debugging

Chapter 5: Security and Traceability

- Infrastructure logical security
 - SSH – encrypting the communications
- Application security
 - Common threats – how to be up to date
 - Injection
 - Cross-site scripting
 - Cross-site request forgery
 - Open redirects
 - Effective code reviews
- Traceability
 - Logging
 - Tracing requests
 - Auditing
 - HTTP codes
 - 1xx – informational
 - 2xx – success codes
 - 3xx – redirection
 - 4xx – client errors
 - 5xx – server errors
 - Why HTTP codes matter in microservices

Chapter 6: Testing and Documenting Node.js Microservices

- Functional testing
 - The pyramid of automated testing
 - Unit tests
 - Integration tests
 - End-to-end tests
 - How much testing is too much?
 - Testing microservices in Node.js
 - Chai
 - Mocha
 - Sinon.JS – a mocking framework
 - Testing a real microservice
 - Manual testing – the necessary evil
- Documenting microservices
 - Documenting APIs with Swagger
 - Generating a project from the Swagger definition

Chapter 7: Monitoring Microservices

- Monitoring services
 - Monitoring using PM2 and Keymetrics
 - Diagnosing problems
 - Monitoring application exceptions
 - Custom metrics

Chapter 8:

- Node.js event loop – easy to learn and hard to master
- Deploying Node.js applications