AMBIENT·IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 11/10/2023

Sign up

Steeltoe Training: .NET C# Microservices

4 days (28 hours)

Presentation

Steeltoe is a powerful toolkit that will help you build your microservices in Microsoft's .NET technology with the C# language. At the end of 2017, Steeltoe was donated by Pivotal to the .NET Foundation, with the aim of offering the same experience as Spring Boot & NetflixOSS from the Java world, to .Net developers. This open source project aims to facilitate the adoption of microservices, and is specifically designed to help developers evolve an application from a monolithic Windows .NET 4.x or Linux .NET Core architecture to a set of .NET microservices.

What is a microservice? It's a programming style with an architecture built around decoupled services with unique responsibilities. This encourages the development of complex applications as a set of small, independent services. This training course will help you to understand and develop this type of architecture in your enterprise projects.

In this training course to become an expert on the subject, we'll give you a general overview of the possibilities offered by microservices in .Net and related technologies. We'll start by examining what microservices are and their main characteristics, as well as the CQRS architecture. We'll develop a distributed system using microservice architecture and set up a Service Bus (RabbitMQ, Azure Event Hub / Kafka), to send messages across separate services, such as MongoDB, which is a NoSQL database. In this course, we'll be focusing on the creation of an HTTP API that will act as a gateway to the entire Activities Service system, responsible for handling incoming messages (or rather, commands that will be distributed by the service bus). We will also implement the Identity Service, which will serve as a Web token through JSON (JWT) to authenticate incoming requests from the API.

Finally, we'll be implementing Steeltoe at the heart of our application, in order to obtain the best components for configuration, service discovery and distributed tracing. These components are based on NetflixOSS. This means that they have been verified in the real world (notably with Netflix), in production scenarios, and have proven that they can adapt to the needs of one of the most demanding microservices architectures.

of the world. So we'll see:

- Steeltoe bindings to Spring Cloud Config Server, which provides a way to push configuration to a set of microservices in a delayed manner (i.e. read config from a service instead of a config file).
- The Eureka client provides links to Eureka Server, Netflix's solution for the discovery of services. What's interesting about Eureka compared to other tools (Consul, etcd) is that it allows trade-offs in terms of consistency and availability that are advantageous for the service discovery scenario. Specifically, Eureka chooses high availability over consistency in order to operate despite a network partition, allowing microservices to continue registering and resolving other services in a given partition. Eureka is also useful for multi-regional availability.
- Cloud Connectors: simplify the way applications connect
  connectors to services, and provide insight into the operating environment of cloud computing platforms such as Cloud Foundry. Spring Cloud connectors are designed for extensibility: use one of the cloud connectors provided, or write your own.
  cloud platform. What's more, you can use the built-in support for commonly used services (relational databases, MongoDB, Redis, RabbitMQ) or extend Cloud Connectors to work with your own services.

Like all our training courses, this one will introduce you to the latest stable version, Steeltoe 3.

Objectives

- Using the .NET Core platform to build a microservices architecture with the Steeltoe Toolkit
- Sending messages through a distributed system using a service bus
- Storing data in a NoSQL database with MongoDB
- Store user identities and authenticate requests using JWT
- Deploy applications in the cloud with Docker and Docker Compose
- Explore commands, events, handlers and other design templates
- Set up unit and integration tests for the distributed system

Target audience

.NET Developers, Architects

Prerequisites

Advanced knowledge of .NET & C# language

Steeltoe Training Program: .NET Microservices

INTRODUCTION

- Best practices and patterns
- The 12-factor methodology
- Introduction to the specification pattern

MODERN ARCHITECTURES

- Introduction to CQRS
- Introduction to EventSourcing
- Introduction to Domain Driven Design

INTRODUCTION TO CQRS

- Queries
- Les Commands
- Command vs query
- CRUD-based interface vs Task-based interface
- Decorator Pattern
- Command and Query Handlers
- CQRS vs DDD
- CQRS vs EventSourcing
- CQRS vs Specification pattern
- Demonstration

COMMUNICATION ARCHITECTURES

- Asynchronous communication
- Introduction to RPC
- Stateless vs. Stateful RPC
- Synchronous communication
- Service Bus
- Introduction to RabbitMQ
- Demonstration: RabbitMQ
- Single addressee communication
- Communication to several recipients
- Introduction to Azure Service Bus
- Demonstration: Azure Service Bus

INTRODUCTION TO MICROSERVICES

- Monolithic application vs. Microservices
- Terminology
- Software lifecycle with Microservices
- Communication in a Microservices architecture
- Application generators for microservices
- STEELTOE Initializr
- Demonstration: STEELTOE Initializr
- Data storage and service configurations
- STEELTOE App Configuration

- Demonstration: STEELTOE App Configuration
- STEELTOE Service Connectors
- Demonstration: STEELTOE Service Connectors
- Distributed services
- STEELTOE Service Discovery
- Demonstration: STEELTOE Service Discovery (Eureka Registry)

IDENTITY SERVICE

- Introduction to OpenID Connect
- Introduction to Auth2
- Authentication and authorization with OpenID Connect and Auth2
- Define roles and access permissions
- Introduction to JWT
- STEELTOE Cloud Security Providers
- Demonstration: STEELTOE Cloud Security Providers

TESTS

- Test-driven development
- Unit tests
- Integration testing
- End-to-end testing

MICROSERVICES DEPLOYMENT

- Microservices packaging and containers
- Container orchestration
- Continuous Delivery (CD: Build, Test, Deploy)
- Versioning

MICROSERVICES MONITORING

- Health checks
- Logging and exception tracking
- Anomaly detection
- Metrics: Performance and statistics
- Traffic control
- Alerts
- STEELTOE Cloud Management
- Demonstration: STEELTOE Cloud Management
- Introduction to the Circuit Breaker pattern
- STEELTOE Circuit Breakers (Netflix Hystrix)
- Demonstration: STEELTOE Circuit Breakers (Netflix Hystrix)

API GATEWAY - API MANAGEMENT

- API Gateway vs. API Management
- Caching
- API versioning
- Controlling and manipulating endpoints
- API Gateways Analytics

Q&A

ADVANCED MODULE + 2 ADDITIONAL DAYS

Introduction to Microservices

- Best practices: the "12 factors" methodology
- Monolithic application VS Microservices
- How do you move an application to microservices?

Command Query Responsibility Segregation (CQRS) architecture

- Task vs. CRUD interfaces
- Support for multiple, scalable, high-performance denormalized views
- Commands, queries, and command/query handlers
- Decorators / command and query handlers: integration with ASP.NET Core
- Separation at domain model level
- Separation at database level
- Synchronization strategies between read and write databases
- CQRS vs. Event Sourcing
- CQRS vs. Specification pattern
- Use a query from a command?
- Command handler decorators vs ASP.NET middleware

Steeltoe

- Configuration providers (Spring Cloud, Vault, etc.)
- Discovery client service (Netflix Eureka, etc.)
- CircuitBreaker (Netflix Hystrix, etc.)
- Management

Identity Service

- Defining a User Entity
- Hashing Passwords
- Storing User Data
- Registering & Logging In

JSON Web Tokens

- Authorization
- Information exchange
- Implementing JWT with HMAC
- Authentication

System Testing

- API Testing
- Activities Service Testing
- Identity Service Testing
- Functional testing
- Non-functional tests or performance tests
- Maintenance

API gateway

- Implementing Event Handlers
- Storing the Data
- Refactoring Endpoints
- Executing HTTP Requests
- Finalizing the API gateway
- Spring Cloud Zuul
- Caching Options
- Resource Expansion
- Protocol Conversion
- Zuul and ETags

Messaging: bus service

- Configuring RabbitMQ Service Bus
- Order creation
- Event creation
- Implementing helper classes and methods
- Implementing API endpoints

Azure Event Hubs

- Anomaly detection (fraud / outliers)
- Application registration
- Analysis pipelines & navigation paths
- Live dashboard
- Data archiving
- Transaction processing
- Telemetry processing by the user
- Continuous telemetry device

Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced computer technology, or to acquire specific business knowledge or modern methods.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Sanction

A certificate will be issued to each trainee who completes the course.