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

Mis à jour le 11/10/2023

S'inscrire

Formation Steeltoe : Microservices .NET C#

4 jours (28 heures)

Présentation

Steeltoe est un puissant toolkit qui vous aidera à construire vos microservices dans la technologie .NET de Microsoft avec le langage C#. Fin 2017, Steeltoe a été donné par Pivotal à la fondation .NET, l'objectif est d'offrir la même expérience que Spring Boot & NetflixOSS issu du monde Java, aux développeurs .Net. Ce projet open source vise à faciliter l'adoption des microservices, il est spécialement conçu pour aider les développeurs à faire évoluer une application d'une architecture monolithique Windows .NET 4.x ou bien .NET Core sous Linux vers un ensemble de microservices .NET.

Qu'est-ce qu'un microservice ? C'est un style de programmation avec une architecture qui est construite à partir de services découplés avec des responsabilités uniques. Ainsi cela favorise le développement d'applications complexes en tant qu'ensemble de petits services indépendants. Cette formation vous aidera à comprendre et à développer ce type d'architecture dans vos projets d'entreprise.

Dans cette formation pour devenir un expert sur le sujet nous allons vous présenter un overview général des possibilités offertes dans le domaine des microservices en .Net et des technologies associées. Nous commencerons par examiner ce que sont les microservices et leurs principales caractéristiques ainsi que l'architecture CQRS. Nous développerons un système distribué utilisant l'architecture microservice et nous mettrons en place un Bus de Service (RabbitMQ, Azure Event Hub / Kafka), pour envoyer des messages à travers les services séparés, tels que MongoDB, qui est une base de données NoSQL. Dans le cadre de cette formation, nous nous concentrerons sur la création d'API HTTP qui servira de passerelle vers l'ensemble du système "Activities Service", responsable de la gestion des messages entrants (ou plutôt des commandes qui seront distribuées par le bus de services). Nous mettrons également en œuvre le service d'identité, qui servira de jeton Web au travers de JSON (JWT) pour authentifier les requêtes entrantes depuis l'API.

Enfin nous mettrons en place, Steeltoe au cœur de notre applicatif afin d'obtenir les meilleurs composants autour de la configuration, de la découverte de services et du traçage distribué. Ces composants sont basés sur NetflixOSS. Cela signifie qu'ils ont été vérifiés dans le monde réel (notamment avec Netflix), dans des scénarios de production, et qu'ils ont prouvé qu'ils peuvent s'adapter aux besoins de l'une des architectures de microservices les plus exigeantes

du monde. Nous verrons donc :

- Les liaisons Steeltoe à Spring Cloud Config Server, qui fournit un moyen de pousser la configuration vers un ensemble de microservices d'une manière tardive (c'est-à-dire de lire la config à partir d'un service au lieu d'un fichier de configuration).
- Le client Eureka fournit des liaisons à Eureka Server, la solution de Netflix pour la découverte de services. Ce qui est intéressant à propos d'Eureka par rapport à d'autres outils (Consul, etcd), c'est qu'il permet de faire des compromis en termes de cohérence et de disponibilité qui sont avantageux pour le scénario de découverte de service. Plus précisément, Eureka choisit la haute disponibilité avant la consistance afin de fonctionner malgré une partition réseau, permettant aux microservices de continuer à s'enregistrer et à résoudre d'autres services dans une partition donnée. Eureka est également utile pour la disponibilité multirégionale.
- Connecteurs Cloud : Ces connecteurs simplifient la manière dont les applications se connectent aux services et permettent de mieux connaître l'environnement d'exploitation des plates-formes de cloud computing, telles que Cloud Foundry. Les connecteurs Spring Cloud sont conçus pour l'extensibilité : utilisez l'un des connecteurs cloud fournis ou écrivez-en un pour votre propre plate-forme cloud. De plus, vous pouvez utiliser le support intégré pour les services couramment utilisés (bases de données relationnelles, MongoDB, Redis, RabbitMQ) ou étendre Cloud Connectors pour travailler avec vos propres services.

Comme toutes nos formations, celle-ci vous présentera la dernière version stable en date à savoir Steeltoe 3.

Objectifs

- Utiliser la plateforme .NET Core pour construire une architecture de microservices à l'aide du Toolkit Steeltoe
- Envoyer des messages par un système distribué à l'aide d'un bus de service
- Stocker les données dans une base de données de type NoSQL avec MongoDB
- Stocker les identités des utilisateurs et authentifier les demandes à l'aide de JWT
- Déployez l'application sur le cloud avec Docker et Docker Compose
- Explorez les commandes, les événements, les gestionnaires et autres modèles de conception
- Mettre en place des tests unitaires et d'intégration du système distribué

Public visé

Développeurs .NET, Architectes

Pré-requis

Connaissances avancées en .NET & langage C#

Programme de la Formation Steeltoe : Microservices .NET

INTRODUCTION

- Bonnes pratiques et patterns
- La méthodologie des 12 facteurs
- Introduction au spécification pattern

ARCHITECTURES MODERNES

- Introduction au CQRS
- Introduction à l'EventSourcing
- Introduction au Domain Driven Design

INTRODUCTION AU CQRS

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

ARCHITECTURES DE COMMUNICATION

- Communication asynchrone
- Introduction à RPC
- Stateless vs Stateful RPC
- Communication synchrone
- Les Service Bus
- Introduction à RabbitMQ
- Démonstration : RabbitMQ
- Communication a destinataire unique
- Communication a plusieurs destinataires
- Introduction à Azure Service Bus
- Démonstration : Azure Service Bus

INTRODUCTION AUX MICROSERVICES

- Application monolithique vs Microservices
- Terminologie
- Cycle de vie d'un logiciel avec les Microservices
- Communication dans une architecture Microservices
- Les générateurs d'applications pour les Microservices
- STEELTOE Initializr
- Démonstration : STEELTOE Initializr
- Le stockage des données et les configurations des différents services
- STEELTOE App Configuration

- Démonstration : STEELTOE App Configuration
- STEELTOE Service Connectors
- Démonstration : STEELTOE Service Connectors
- Les services distribués
- STEELTOE Service Discovery
- Démonstration : STEELTOE Service Discovery (Eureka Registry)

SERVICE D'IDENTITE

- Introduction à l’OpenID Connect
- Introduction à Auth2
- Authentification et autorisation avec OpenID Connect et Auth2
- Définir les rôles et les permissions d’accès
- Introduction au JWT
- STEELTOE Cloud Security Providers
- Démonstration : STEELTOE Cloud Security Providers

LES TESTS

- Test-driven development
- Les tests unitaires
- Les tests d’intégration
- End-to-end tests

DEPLOIEMENT DES MICROSERVICES

- Le packaging et les conteneurs des Microservices
- Orchestration de conteneurs
- Livraison continue (CD : Build, Test, Deploy)
- Versionning

MONITORING DES MICROSERVICES

- Health checks
- Logging et tracking des exceptions
- Détection d’anomalies
- Métrique: Performance et statistiques
- Contrôle du trafic
- Les alertes
- STEELTOE Cloud Management
- Démonstration : STEELTOE Cloud Management
- Introduction au Circuit Breaker pattern
- STEELTOE Circuit Breakers (Netflix Hystrix)
- Démonstration : STEELTOE Circuit Breakers (Netflix Hystrix)

API GATEWAY - API MANAGEMENT

- API Gateway vs API Management
- Caching
- API versioning
- Contrôle et manipulations des endpoints
- API Gateways Analytics

Q&A

MODULE AVANCE + 2 JOURS SUPPLÉMENTAIRES

Introduction au Microservices

- Best practices : la méthodologie des "12 facteurs"
- Application monolithique VS Microservices
- Comment faire évoluer une application vers les microservices ?

Architecture CQRS (Command Query Responsibility Segregation)

- Task vs. CRUD interfaces
- Prise en charge de plusieurs vues dénormalisées multiples, évolutives et performantes
- Commands, queries, and command/query handlers
- Decorators / command and query handlers : intégration avec ASP.NET Core
- Séparation au niveau du modèle de domaine
- Séparation au niveau de la base de données
- Stratégies de synchronisation entre les bases de données en lecture et en écriture
- CQRS vs. Event Sourcing
- CQRS vs. Specification pattern
- Utiliser une requête à partir d'une commande ?
- Command handler decorators vs ASP.NET middleware

Steeltoe

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

Identity Service

- Définir un User Entity
- Hashing Passwords
- Storing User Data
- Registering & Logging In

JSON Web Tokens

- Autorisation
- Échange d'informations
- Implémentation de JWT avec HMAC
- Authentification

System Testing

- API Testing
- Activities Service Testing
- Identity Service Testing
- Tests fonctionnels
- Essais non fonctionnels ou essais de performance
- 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 et ETags

Messaging : service bus

- Configurer RabbitMQ Service Bus
- Création d'une commande
- Création d'événements
- L'implémentation de classes et de méthodes d'aide
- L'implémentation des points finaux de l'API

Azure Event Hubs

- Détection d'anomalies (fraude / valeurs aberrantes)
- Enregistrement des applications
- Les pipelines d'analyse & parcours de navigation
- Tableau de bord en direct
- Archivage des données
- Traitement des transactions
- Traitement de la télémétrie par l'utilisateur
- Dispositif de télémétrie en continu

Sociétés concernées

Cette formation s’adresse à la fois aux particuliers ainsi qu’aux entreprises, petites ou grandes, souhaitant former ses équipes à une nouvelle technologie informatique avancée ou bien à acquérir des connaissances métiers spécifiques ou des méthodes modernes.

Méthodes pédagogiques

Stage Pratique : 60% Pratique, 40% Théorie. Support de la formation distribué au format numérique à tous les participants.

Organisation

Le cours alterne les apports théoriques du formateur soutenus par des exemples et des séances de réflexions, et de travail en groupe.

Validation

À la fin de la session, un questionnaire à choix multiples permet de vérifier l’acquisition correcte des compétences.

Sanction

Une attestation sera remise à chaque stagiaire qui aura suivi la totalité de la formation.