MCROSERVICES WITH SPRING BOOT AND SPRING CLOUD EĞİTİMİ

4 GÜN

Digital Vizyon
Akademi

www.digitalvizyon.net

İçindekiler

Eğitim Hakkında

"Spring Boot ve Spring Cloud ile Mikroservisler" eğitimi, Java tabanlı mikroservis uygulamalarının geliştirilmesi üzerine bir eğitimdir. Eğitimde, katılımcılar Spring Boot ve Spring Cloud framework'lerinin kullanımını ve avantajlarını öğrenirler. Bu süreçte, mikroservis mimarisine dair temel bilgiler edinirler ve bu mimariyi kullanarak uygulama geliştirme becerilerini artırırlar.

Eğitim, uygulama geliştirme sürecinde çeşitli konuları ele alır. Bunlar arasında uygulamanın parçalanması, servislerin birbirine bağlanması, veri depolama, uygulamanın test edilmesi ve dağıtılması bulunur. Bu konuların üzerinden geçilerek, katılımcıların geliştirme süreçlerini daha iyi yönetmeleri sağlanır.

Spring Cloud ile uygulamanın yapılandırılması, eş zamanlılık, güvenlik ve performans gibi konular da eğitimde yer alır. Bu sayede, katılımcılar uygulamanın genişletilebilirliğini, esnekliğini ve skalenebilirliğini optimize etme konusunda bilgi ve beceri kazanır.

Eğitim sürecinde uygulama örnekleri, laboratuvar çalışmaları ve proje çalışmaları gibi öğrenme yöntemleri kullanılır. Katılımcılar, eğitime başlamadan önce Java programlama diline web uygulamalarına dair temel bilgilendir.

Eğitim sonucunda, katılımcılar mikroservis uygulamalarının nasıl geliştirileceğini ve yönetileceğini öğrenirler. Spring Boot ve Spring Cloud framework'lerini kullanma becerisine sahip olurlar. Eğitimden edindikleri bilgilerle, uygulamaların performansını izleme, problemleri çözme ve uygulamayı güncelleme gibi görevleri yerine getirebilirler.

Eğitim, uygulamanın güvenliği ve performansına da önem verir. Katılımcılar, bu konularda nasıl daha iyi performans göstermeleri gerektiğini öğrenirler. Endüstri standartları ve iyi uygulamalar hakkında bilgi edinirler. Böylece uygulamalarının kullanıcı tarafından daha iyi benimsenmesini ve kolay kullanılmasını sağlarlar.

Sonuç olarak, "Spring Boot ve Spring Cloud ile Mikroservisler" eğitimi, Java tabanlı mikroservis uygulamalarını yönetmek ve geliştirmek isteyen yöneticiler, sistem yöneticileri ve uygulama geliştiricileri için ideal bir eğitimdir. Bu eğitim, katılımcıların mikroservis mimarisini ve Spring Boot ve Spring Cloud framework'lerini etkili bir şekilde kullanmalarını sağlar. Böylece, kendi uygulamalarını daha verimli ve kullanıcı dostu bir şekilde geliştirebilirler.

Neler Öğreneceksiniz?

"Microservices with Spring Boot and Spring Cloud" eğitimi sırasında şunları öğrenebilirsiniz:

- Microservices mimarisinin prensiplerini ve avantajlarını
- Spring Boot framework'ünün nasıl kullanılabileceğini
- RESTful API'lerin nasıl oluşturulabileceğini
- Spring Cloud framework'ünün nasıl yapılandırılabileceğini

- Config server, discovery server, API gateway gibi dağıtık sistemlerin nasıl kullanılabileceğini
- Uygulama içi mesajlaşma için RabbitMQ gibi araçların nasıl kullanılabileceğini
- Microservices mimarisi kullanarak büyük ve karmaşık uygulamaları nasıl parçalara bölebileceğini
- Parçaları nasıl test edebileceği ve yönetebileceğini öğreneceksiniz.

Bu eğitim ile, microservices mimarisi kullanarak Java ile geliştirilen uygulamaların tasarımını ve yapılandırmasını anlayabilecek ve uygulamalı olarak öğrenebileceksiniz..

Ön Koşullar

“Microservices with Spring Boot and Spring Cloud” eğitimine katılmak için genel olarak aşağıdaki ön koşullar gerekmektedir:

- Java programlama dili hakkında temel bilgi
- Object Oriented Programming (OOP) kavramları hakkında bilgi
- RESTful API kavramları hakkında temel bilgi
- Dağıtık sistemler ve mimari kavramları hakkında temel bilgi

Bu ön koşulların yanı sıra, Spring Boot ve Spring Cloud gibi araçlar hakkında da önceki tecrübe size eğitimin daha verimli geçmesini sağlayabilir

Kimler Katılmalı

“Microservices with Spring Boot and Spring Cloud” eğitimi, aşağıdaki kişiler için uygun olabilir:

- Java programlama dili ile uygulama geliştirmek isteyen Full-Stack geliştiriciler
- Microservices mimarisi hakkında bilgi sahibi olmak isteyen geliştiriciler
- Spring Boot ve Spring Cloud gibi araçları kullanarak uygulamalar geliştirmek isteyen geliştiriciler
- RESTful API’lerin nasıl oluşturulabileceğini ve yapılandırılabileceğini öğrenmek isteyen geliştiriciler
- Büyük ve karmaşık uygulamaları parçalara bölerek yönetmek isteyen geliştiriciler

Bu eğitim, microservices mimarisi ve Spring Boot/Spring Cloud kullanarak uygulama geliştirmek isteyen geliştiricilere yönelik bir eğitimdir ve bu alanda çalışmak isteyen ya da ilerlemek isteyen kişiler için faydalı olabilir.

Outline

Introduction to the Spring Framework

- What is the Spring Framework?
- Spring Philosophies

- Why Spring?
- Spring Modules
- Requirements and Supported Environments
- Using Spring with Servers
- Role of Spring Container
- Spring Example
- Avoiding Dependency on Spring
- Additional Spring Projects/Frameworks
- Summary

Spring Annotation Configuration

- Spring Containers
- Annotation-based Spring Bean Definition
- Scanning for Annotation Components
- Defining Component Scope Using Annotations
- JSR-330 @Named Annotation
- JSR-330 @Scope
- Annotation-based Dependency Injection
- Wiring Bean using @Inject
- @Autowired – Constructor
- @Autowired – Field
- @Autowired – method
- @Autowired – Collection
- @Autowired – Maps
- @Autowired & @Qualifier with Constructors, Fields, and Methods
- @Autowired & Custom Qualifiers
- @Autowired & Simple Custom Qualifier Field
- @Autowired & Simple Custom Qualifier Method
- @Autowired & CustomAutowireConfigurer
- Dependency Injection Validation
- @Resource
- @PostConstruct and @PreDestroy
- Summary

Spring Framework Configuration

- Java @Configuration Classes
- Defining @Configuration Classes
- Loading @Configuration Classes
- Modularizing @Configuration Classes
- Qualifying @Bean Methods
- Trouble with Prototype Scope
- Configuration with Spring Expression Language

- Resolving Text Messages
- Spring Property Conversion
- Spring Converter Interface
- Using Custom Converters
- Spring PropertyEditors
- Registering Custom PropertyEditors
- Summary

Introduction to Spring Boot

- What is Spring Boot?
- Spring Boot Main Features
- Spring Boot on the PaaS
- Understanding Java Annotations
- Spring MVC Annotations
- Example of Spring MVC-based RESTful Web Service
- Spring Booting Your RESTful Web Service
- Spring Boot Skeletal Application Example
- Converting a Spring Boot Application to a WAR File
- Externalized Configuration
- Starters
- The 'pom.xml' File
- Spring Boot Maven Plugin
- HOWTO: Create a Spring Boot Application
- Summary

Spring MVC

- Spring MVC
- Spring Web Modules
- Spring MVC Components
- DispatcherServlet
- Template Engines
- Spring Boot MVC Example
- Spring MVC Mapping of Requests
- Advanced @RequestMapping
- Composed Request Mappings
- Spring MVC Annotation Controllers
- Controller Handler Method Parameters
- Controller Handler Method Return Types
- View Resolution
- Spring Boot Considerations
- Summary

Overview of Spring Boot Database Integration

- DAO Support in Spring
- Spring Data Access Modules
- Spring JDBC Module
- Spring ORM Module
- DataAccessException
- @Repository Annotation
- Using DataSources
- DAO Templates
- DAO Templates and Callbacks
- ORM Tool Support in Spring
- Summary

Using Spring with JPA or Hibernate

- Spring JPA
- Benefits of Using Spring with ORM
- Spring @Repository
- Using JPA with Spring
- Configure Spring Boot JPA EntityManagerFactory
- Application JPA Code
- “Classic” Spring ORM Usage
- Spring JpaTemplate
- Spring JpaCallback
- JpaTemplate Convenience Features
- Spring Boot Considerations
- Spring Data JPA Repositories
- Summary

Introduction to MongoDB

- MongoDB
- MongoDB Features
- MongoDB’s Logo
- Positioning of MongoDB
- MongoDB Applications
- MongoDB Data Model
- MongoDB Limitations
- MongoDB Use Cases
- MongoDB Query Language (QL)
- The CRUD Operations
- The
- find
- Method

- The
- findOne
- Method
- A MongoDB QL Example
- Data Inserts
- MongoDB vs Apache CouchDB
- Summary

Working with Data in MongoDB

- Reading Data in MongoDB
- The Query Interface
- Query Syntax is Driver-Specific
- Projections
- Query and Projection Operators
- MongoDB Query to SQL Select Comparison
- Cursors
- Cursor Expiration
- Writing Data in MongoDB
- An Insert Operation Example
- The Update Operation
- An Update Operation Example
- A Remove Operation Example
- Limiting Return Data
- Data Sorting
- Aggregating Data
- Aggregation Stages
- Accumulators
- An Example of an Aggregation Pipe-line
- Map-Reduce
- Summary

Spring Data with MongoDB

- Why MongoDB?
- MongoDB in Spring Boot
- xml
- Application Properties
- MongoRepository
- Custom Query Methods
- Supported Query Keywords
- Complex Queries
- Create JavaBean for Data Type
- Using the Repository

- Summary

Spring REST Services

- Many Flavors of Services
- Understanding REST
- RESTful Services
- REST Resource Examples
- REST vs SOAP
- REST Services With Spring MVC
- Spring MVC @RequestMapping with REST
- Working With the Request Body and Response Body
- @RestController Annotation
- Implementing JAX-RS Services and Spring
- JAX-RS Annotations
- Java Clients Using RestTemplate
- RestTemplate Methods
- Summary

Spring Security

- Securing Web Applications with Spring Security 3.0
- Spring Security 3.0
- Authentication and Authorization
- Programmatic v Declarative Security
- Getting Spring Security from Maven
- Spring Security Configuration
- Spring Security Configuration Example
- Authentication Manager
- Using Database User Authentication
- LDAP Authentication
- Summary

Spring JMS

- Spring JMS
- JmsTemplate
- Connection and Destination
- JmsTemplate Configuration
- Transaction Management
- Example Transaction Configuration
- Producer Example
- Consumer Example
- Converting Messages
- Message Listener Containers

- Message-Driven POJO's Async Receiver Example
- Message-Driven POJO's Async Receiver Configuration
- Spring Boot Considerations
- Summary

Microservices

- What is a "Microservice"?
- One Helpful Analogy
- SOA – Microservices Relationship
- ESB – Microservices Relationship
- Traditional Monolithic Designs and Their Role
- Disadvantages of Monoliths
- Moving from a Legacy Monolith
- When Moving from a Legacy Monolith
- The Driving Forces Behind Microservices
- How Can Microservices Help You?
- The Microservices Architecture
- Utility Microservices at AWS
- Microservices Inter-connectivity
- The Data Exchange Interoperability Consideration
- Managing Microservices
- Implementing Microservices
- Embedding Databases in Java
- Microservice-Oriented Application Frameworks and Platforms
- Summary

Spring Cloud Config

- The Spring Cloud Configuration Server
- Why Configuration Management is Important
- Configuration Management Challenges in Microservices
- Separation of Configuration from Code
- Configuration Service
- How the Configuration Service Works
- Cloud Config Server Properties File
- Git Integration
- Properties
- Configuration Client
- Sample Client Config File
- Sample Client Application
- Dynamic Property Updates – Server
- Dynamic Property Update – Client
- Dynamic Property Update – Execute

- Summary

Service Discovery with Netflix Eureka

- Service Discovery in Microservices
- Load Balancing in Microservices
- Netflix Eureka
- Eureka Architecture
- Communications in Eureka
- Time Lag
- Eureka Deployment
- Peer Communication Failure between Servers
- Eureka Server Configuration
- Eureka Client/Service
- Eureka Client Properties
- Spring Cloud DiscoveryClient Interface
- ServiceInstance JSON
- ServiceInstance Interface
- What about Services
- Eureka and the AWS Ecosystem
- Summary

Load-Balancing with Netflix Ribbon

- Load Balancing in Microservices
- Netflix Ribbon
- Server-side load balance
- Client-side Load Balance
- Architecture
- Load Balance Rules
- RoundRobinRule
- AvailabilityFilteringRule
- WeightedResponseTimeRule
- RandomRule
- ZoneAvoidanceRule
- IPing Interface (Failover)
- Using Ribbon
- YAML Configuration
- Configuration Class
- Client Class
- Client Class Implementation
- Integration with Eureka (Service Discovery)
- Using Ribbon in the Amazon AWS Cloud
- Summary

Application Hardening with Netflix Hystrix

- Netflix Hystrix
- Design Principles
- Design Principles (continued)
- Cascading Failures
- Bulkhead Pattern
- Circuit Breaker Pattern
- Thread Pooling
- Request Caching
- Request Collapsing
- Fail-Fast
- Fallback
- Using Hystrix
- Circuit Breaker Configuration
- Fallback Configuration
- Collapser Configuration
- Rest Controller and Handler
- Collapser Service (Part 1)
- How the Collapser Works
- Hystrix Monitor
- Enable Monitoring
- Turbine
- The Monitor
- Monitor details
- Summary

Edge Components with Netflix Zuul

- Zuul is the Gatekeeper
- Request Handling
- Filters
- Filter Architecture
- Filter Properties
- filterType()
- filterOrder()
- shouldFilter()
- Run()
- Cancel Request
- Dynamic Filter Loading
- Filter Communications
- Routing with Eureka and Ribbon
- Summary

Distributed Tracing with Zipkin

- Zipkin
- Zipkin Features
- Architecture
- The Collector
- Storage
- API
- GUI Console
- Zipkin Console Homepage
- View a Trace
- Trace Details
- Dependencies
- Dependency Details
- Zipkin in Spring Boot
- Zipkin Configuration
- Summar