DEVELOPING MICROSERVICES EĞİTİMİ

5 GÜN

Digital Vizyon
Akademi

www.digitalvizyon.net

İçindekiler

Eğitim Hakkında

"Developing Microservices Eğitimi", microservices tabanlı uygulamaların geliştirilmesi sürecini kapsamlı bir şekilde öğretir. Bu eğitim, microservices tabanlı uygulamaların geliştirilmesinde gereken temel konuları ele alır.

Eğitim, microservices mimarisinin temel prensiplerini, veri yönetimi, hata toleransı, mesajlaşma ve diğer kilit kavramları öğretir. Katılımcılar, gerçek hayattan örnekler ve projeler aracılığıyla microservices geliştirme becerilerini artırırlar.

Eğitim ayrıca, uygulamaların microservices tabanlı olarak geliştirilmesi sürecinde hangi araç ve teknolojilerin nasıl kullanılabileceğini de öğretir. Katılımcılar, büyük veri işleme, real-time analiz, hata toleransı ve diğer özelliklerin nasıl yönetileceğini öğrenirler. Ayrıca, microservices mimarisi tabanlı uygulamaların nasıl geliştirilip yönetileceğini de anlarlar.

"Developing Microservices Eğitimi", microservices tabanlı uygulama geliştirme sürecinin tümünü kapsar. Katılımcılar, modern uygulamalar geliştirmeye başlamadan önce ihtiyaç duyacakları temel becerileri kazanırlar. Eğitim, örnekler, pratik uygulamalar ve projeler yoluyla öğrenmeyi sağlar ve katılımcıların microservices tabanlı uygulama geliştirmeye başlamalarına yardımcı olur.

Eğitim programı, microservices mimarisinin temelleriyle başlar. Katılımcılar, veri yönetimi, mesajlaşma sistemlerinin yönetimi ve hata toleransı gibi temel kavramları öğrenirler. Ayrıca, microservices tabanlı modern uygulamaların geliştirilmesi ve yönetilmesi konusunda nasıl bir rol oynadığına dair bilgi sahibi olurlar. Bu bilgiler, katılımcıların uygulama geliştirme sürecinde ihtiyaç duyacakları temel yapı taşlarını oluşturur.

Eğitimde, microservices mimarisi ve bununla ilişkili temel özellikler ve bileşenler üzerinde duruyoruz. Bu, katılımcılara real-time veri işleme, büyük veri akışları ve hata toleransı gibi yetenekleri kazandırır. Konu akışları, veri yönetimi ve dağıtık sistemlerin yönetimi gibi konular işlenir.

Son olarak, bir uygulamanın nasıl microservices tabanlı geliştirileceği hakkında bilgi veriyoruz. Bu süreç, uygulamanın son testlerini yapmayı, veri yönetimini, ve en sonunda uygulamanın microservices tabanlı geliştirilmesini içerir. Bu bilgiler, katılımcıların uygulamalarını başarılı bir şekilde microservices tabanlı geliştirmelerine yardımcı olur.

Neler Öğreneceksiniz?

Developing Microservices Eğitiminde, aşağıdaki konuları öğreneceksiniz:

- Mikroservis mimarisi kavramı ve prensipleri
- Mikroservis mimarisi kullanarak uygulama tasarımı ve dağıtımı
- RESTful web servisleri kullanarak mikroservislerin oluşturulması
- Mikroservisler arasındaki veri ve hizmetlerin etkileşimi
- Mikroservislerin test edilmesi ve çalıştırılması

- Mikroservislerin yönetilmesi ve çalışmasını izleme
- Mikroservis platformları ve araçları hakkında bilgi
- Mikroservis mimarisiyle bulut tabanlı uygulama geliştirme

Bu eğitim, mikroservis mimarisi hakkında temel bilgi sahibi olmanızı ve bu mimari kullanarak uygulama geliştirme becerilerini geliştirmenizi amaçlar.

Ön Koşullar

Developing Microservices Eğitiminin ön koşulları

- İster Java, ister başka bir programlama dili ile birlikte, programlama dillerinde temel bilgi
- Web servisleri, API ve RESTful kavramları hakkında bilgi
- Veri yapıları ve veritabanı kavramları hakkında temel bilgi
- Agile ve DevOps yaklaşımları hakkında bilgi
- Bulut teknolojileri hakkında temel bilgi

Kimler Katılmalı

Developing Microservices Eğitimine aşağıdaki kişiler katılım sağlayabilir:

- Yazılım Geliştiricileri: Uygulama geliştirme projelerinde rol alan yazılım geliştiricileri
- Yazılım Mimarları: Uygulama mimarisi ve tasarımı hakkında sorumlu olan yazılım mimarları
- DevOps Engineer: Uygulama dağıtımı, çalıştırma ve yönetimi hakkında sorumlu olan DevOps mühendisleri
- IT Yöneticileri: IT ekibi ve projelerinin yönetimi hakkında sorumlu olan IT yöneticileri

Eğitimin amaçlarına ve içeriğine göre, diğer profesyonel kategoriler de eğitime katılabilir. Bu eğitim, mikroservis mimarisi hakkında bilgi sahibi olmak isteyen ve bu mimari kullanarak uygulama geliştirme becerilerini geliştirmek isteyen herkes için uygun olabilir.

Outline

Microservice Development

- What are Microservices?
- Microservices vs Classic SOA
- Principles of Microservices Architecture Design
- Business Domain-Centric Design
- Designing for failure
- Microservices Architecture – Pros
- Microservices Architecture – Cons
- Docker and Microservices
- Microservice Deployment with Docker – Workflow

- Writing Dockerfile
- Kubernetes
- What is OpenShift
- OpenShift Architecture
- Microservices and Various Applications
- Web Applications
- Web Applications – Reference Architecture
- Web Applications – When to use?
- Rich Client Applications
- Rich Client Applications – Reference Architecture
- Rich Client Applications – When to use?
- Rich Internet Applications
- Rich Internet Applications – Reference Architecture
- Rich Internet Applications – When to use?
- Mobile Applications
- Mobile Applications – Reference Architecture
- Mobile Applications – When to use?
- Service Applications
- Service Applications – Reference Architecture
- Service Applications – When to use?
- Single Page Applications
- Single Page Applications – Benefits
- Traditional Enterprise Application Architecture
- Sample Microservices Architecture
- Serverless & Event-driven Microservice – AWS Lambda
- Summary

REST Services

- Many Flavors of Services
- Understanding REST
- Principles of RESTful Services
- REST Example – Create
- REST Example – Retrieve
- REST Example – Update
- REST Example – Delete
- REST Example – Client Generated ID
- SOAP Equivalent Examples
- REST Example – JSON
- REST vs SOAP Communication
- More REST vs SOAP
- REST vs SOAP Summary
- RESTful Services Usage
- Additional Resources
- Summary

Advanced Objects and Functionality in JavaScript

- JavaScript Evolution
- Basic Objects

- Constructor Function
- More on the Constructor Function
- Object Properties
- Deleting a Property
- The instanceof Operator
- Object Properties
- Constructor and Instance Objects
- Constructor Level Properties
- Namespace
- Functions Are First-Class Objects
- Closures
- Closure Examples
- Private Variables with Closures
- Immediately Invoked Function Expression (IIFE)
- The Module Pattern
- Module Pattern Example
- Prototype
- Inheritance in JavaScript
- The Prototype Chain
- Traversing Prototype Property Hierarchy
- Prototype Chain
- Inheritance Using Prototype
- Extending Inherited Behavior
- Enhancing Constructors
- Improving Constructor Performance
- Inheritance with Object.create
- The hasOwnProperty Method
- Summary

React Overview

- What is React?
- What's in a Name?
- React Component Model
- What React Is Not
- What You Will Not Find in React
- Motivation for Creating React
- A React JavaScript Example
- One-Way Data Flow
- JSX
- A JSX Example
- The Virtual (Mock) DOM
- Only Sub-components that Actually Change are Re-Rendered
- React Libraries
- Summary

Programming with React API

- React Programming Options
- Components vs Elements

- Three Ways to Create a React UI Component
- React API On-Line Documentation
- Setting Up the Libraries
- The ReactDOM Object
- The ReactDOM Object (Cont'd)
- The React Object
- The React.createElement Method
- The ReactElement Object
- The ReactElement Structure
- The React.DOM Object
- The React.PropTypes Object
- The React.Children Object
- The propTypes Object
- Lifecycle Methods (Applied only to ES6 Classes)
- Summary

Basic Components and JSX

- What is JSX?
- JSX Transpilation to React Code Example
- Running the Transpiled Code
- Babel
- Babel JavaScript Library
- Script Import Skeleton Code
- Playing Around in CodePen
- React Components and Properties (Props)
- Ways to Create UI Components
- Creating a Functional Component Example
- Component Names Must Be Capitalized
- Creating a UI Component with React.createClass()
- The render Method Object
- Creating a UI Component Using ES6 Class Notation
- Using ES6 Classes with React
- Which UI Component Creation Syntax Should I Use?
- Components vs Elements
- Elements Are Immutable
- Properties
- Property Naming Convention
- Properties Default to 'True'
- Spread Attributes (an ES6 Feature)
- Expressions
- Summary

Introduction to Node.js

- What Is Node.js?
- Application of Node.js
- Installing Node.js and NPM
- "Hello, Node World!"
- How It Works

- Built on JavaScript: Benefits
- Traditional Server-Side I/O Model
- Disadvantages of the Traditional Approach
- Event-Driven, Non-Blocking I/O
- Concurrency
- Using Node Package Manager (NPM)
- Express
- Microservices with Node.js
- The Express Package
- Installing and Using Express
- Defining Routing Rules in Express
- Route Path
- The Response Object
- A Simple Web Service with Express Example
- Composite Services
- Example – Call an API Using a Promise
- Using the callApi() Function
- Summary

Extending React

- The Need to Extend React
- Redux
- Redux Design Ideas
- React Router
- React Router Code Examples
- Issues With Manual Module Management
- Webpack
- Testing React Apps: ReactTestUtils
- Testing React Apps: Jest
- Testing with Jest and Enzyme
- Summary

React Component Concepts

- Nesting JSX Elements
- Example of JSX Nesting
- Comments in JSX Code
- JSX Escapes Values
- Event Handling
- Event Handler Example
- Working with Lists of Items
- Keys in Lists
- Example List With Key
- Container vs. Presentational Components
- State
- Types of State Data
- State Hierarchy
- Lift State Up
- Props vs. State

- Pass Down a Function
- Immutability
- Immutability – Why?
- Virtual DOM and State
- Setting state
- Updating Input fields
- Passing Props to Components
- Passing Functions to Components
- Event Binding – DOs
- Event Binding – Don'ts
- Passing Parameters to Event Handlers
- App Development Workflow – 1/3
- App Development Workflow – 2/3
- App Development Workflow – 3/3
- 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 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

Spring REST Services

- REST Services With Spring MVC
- Spring MVC Components
- Spring MVC @RequestMapping with REST
- Working With the Request Body and Response Body
- @RestController Annotation
- Implementing JAX-RS Services and Spring
- JAX-RS Annotations
- Spring Security
- Spring Security Options
- Spring Security Features
- 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

Introduction to Couchbase

- What is Couchbase?
- Key Components of Couchbase
- Benefits of Couchbase
- Basics of Data Modeling
- Modeling One-to-many Relationship
- Modeling Many-to-many
- Doing a Query
- About Query Index
- Example MapReduce View
- Summary

Introduction to Couchbase Programming Using Java

- Getting Started
- Opening a Connection
- Creating Index
- Doing a Query Using MapReduce View
- Doing an N1QL Query
- Retrieve a Document by ID
- Adding a Document
- Updating a Document
- Deleting a Document
- Summary

Introduction to KAFKA

- Messaging Architectures – What is Messaging?
- Messaging Architectures – Steps to Messaging
- Messaging Architectures – Messaging Models
- What is Kafka?
- What is Kafka? (Contd.)
- Kafka Overview
- Kafka Overview (Contd.)
- Need for Kafka
- Kafka Partitions
- Kafka Architecture
- Core concepts in Kafka
- Kafka Topic
- Kafka Producer
- Kafka Consumer
- Kafka Broker
- Kafka Cluster
- Why Kafka Cluster?
- Sample Multi-Broker Cluster
- Overview of ZooKeeper
- Kafka Cluster & ZooKeeper
- Who Uses Kafka?
- Summary

Using Apache Kafka

- Installing Apache Kafka
- Configuration Files
- Starting Kafka
- Using Kafka Command Line Client Tools
- Setting up a Multi-Broker Cluster
- Using Multi-Broker Cluster
- Kafka Connect
- Kafka Connect – Configuration Files
- Using Kafka Connect to Import/Export Data
- Creating a Spring Boot Producer
- Adding Kafka dependency to pom.xml
- Defining a Spring Boot Service to Send Message(s)
- Defining a Spring Boot Controller
- Testing the Spring Boot Producer
- Creating a Nodejs Consumer
- Summary

Introduction to Kubernetes

- What is Kubernetes
- What is a Container
- Container – Uses
- Container – Pros
- Container – Cons

- Composition of a Container
- Control Groups
- Namespaces
- Union Filesystems
- Popular Containerization Software
- Microservices
- Microservices and Containers / Clusters
- Microservices and Orchestration
- Microservices and Infrastructure-as-Code
- Kubernetes Container Networking
- Kubernetes Networking Options
- Kubernetes Networking – Balanced Design
- Summary

Kubernetes – From the Firehose

- What is Kubernetes?
- Container Orchestration
- Kubernetes Basic Architecture
- Kubernetes Detailed Architecture
- Kubernetes Concepts
- Cluster and Namespace
- Node
- Master
- Pod
- Label
- Annotation
- Label Selector
- Replication Controller and Replica Set
- Service
- Storage Volume
- Secret
- Resource Quota
- Authentication and Authorization
- Routing
- Registry
- Using Docker Registry
- Summary

Docker Introduction

- What is Docker
- Where Can I Ran Docker?
- Docker and Containerization on Linux
- Linux Kernel Features: cgroups and namespaces
- The Docker-Linux Kernel Interfaces
- Docker Containers vs Traditional Virtualization
- Docker as Platform-as-a-Service
- Docker Integration
- Docker Services

- Docker Application Container Public Repository
- Competing Systems
- Docker Command-line
- Starting, Inspecting, and Stopping Docker Containers
- Summary

CI/CD with OpenShift, Jenkins, and Blue Ocean

- What is OpenShift
- OpenShift Online
- OpenShift Origin
- OpenShift Architecture
- OpenShift Origin Installation
- OpenShift CLI
- OpenShift CLI (Contd.)
- Jenkins Continuous Integration
- Jenkins Features
- Running Jenkins
- Downloading and Installing Jenkins
- Running Jenkins as a Stand-Alone Application
- Running Jenkins on an Application Server
- Installing Jenkins as a Windows Service
- Different types of Jenkins job
- Configuring Source Code Management(SCM)
- Working with Subversion
- Working with Subversion (cont'd)
- Working with Git
- Build Triggers
- Schedule Build Jobs
- Polling the SCM
- Maven Build Steps
- Jenkins / OpenShift Pipeline
- Jenkins / OpenShift Pipeline Output
- Installing Jenkins Plugins
- The Blue Ocean Plugin
- Blue Ocean Plugin Features
- New modern user experience
- Advanced Pipeline visualizations with built-in failure diagnosis
- Branch and Pull Request awareness
- Personalized View
- OpenShift Pipeline Output
- Creating OpenShift Blue Ocean Pipeline
- Summary

Operational Readiness

- What is Operational Readiness
- Telemetry
- End-to-end Requirements Traceability
- Log Strategy

- Monitoring Strategy
- Runbooks
- Summary

Application Modernization

- What is Application Modernization
- Typical App Modernization Projects
- Why Modernization?
- Goals for Application Modernization
- Modernization Process
- Modernization in a Nutshell
- Modernization in a Nutshell – Analyze
- Modernization in a Nutshell – Rationalize
- Modernization in a Nutshell – Modernize
- Modernization in a Nutshell – Supervise
- Twelve-factor Applications
- Twelve Factors, Microservices, and App Modernization
- 12-Factor Microservice Codebase
- 12-Factor Microservice Dependencies
- 12-Factor Microservice Config
- 12-Factor Microservice Backing Services
- 12-Factor Microservice Continuous Delivery
- 12-Factor Microservice Processes
- 12-Factor Microservice Data Isolation
- 12-Factor Microservice Concurrency
- 12-Factor Microservice Disposability
- 12-Factor Microservice Environment Parity
- 12-Factor Microservice Logs
- 12-Factor Microservice Admin Processes
- Monolithic revisited
- Monolithic vs. Microservices
- Maintaining State in App Modernization
- Cloud Service Fabric
- Summary

Introduction to Feign

- What is Feign
- Feign – Annotations
- Creating a REST client with Feign
- Benefits of using Feign
- Feign – Default Beans
- FeignFeign – Simple Example
- Multiple Interfaces
- Ribbon
- Ribbon Load Balancing Architecture
- Using Ribbon
- Ribbon and Feign
- Hystrix

- Hystrix Dependency
- Using Hystrix
- Summary

Activit Workflow

- Business Process Management
- Business Process Model and Notation
- BPMN (Contd.)
- BPMN – Elements
- BPMN 2.0
- What is Activiti
- Activiti – Components
- Activiti – Alternative Modeling GUI
- Activiti – Sample workflow in Activit Process Modeler
- Using Spring Boot with Activiti
- Spring Boot with Activiti – Getting Started
- Spring Boot with Activiti – Simple Application
- Spring Boot with Activiti – Add BPMN 2.0 process definition
- Spring Boot with Activiti – Create a CommandLineRunner
- Spring Boot with Activiti – Create a Bean
- Spring Boot with Activiti – Start the Application
- Summary