.NET Core

MICROSERVICES USING .NET CORE EĞİTİMİ

5 GÜN

Digital Vizyon Akademi

www.digitalvizyon.net

İçindekiler

Eğitim Hakkında

“Microservices Using .NET Core Eğitimi”, .NET Core kullanarak mikroservislerin nasıl geliştirileceğine dair geniş kapsamlı bir eğitim sunar. Bu eğitim, mikroservis mimarisi konularını ve bu alanda gereken başlangıç seviye becerileri ele alır.

Eğitim, .NET Core ve mikroservislerin temel özelliklerini, servis tasarımı, performans iyileştirme, güvenlik ve diğer kilit kavramları öğretir. Katılımcılar, gerçek hayattan örnekler ve projeler aracılığıyla bu konular hakkındaki bilgilerini artırırlar.

Ayrıca, eğitim, .NET Core kullanılarak mikroservislerin tasarlanması ve geliştirilmesi sürecinde hangi araç ve teknolojilerin nasıl kullanılabileceğini öğretir. Katılımcılar, veri işleme, performans iyileştirme, servis tasarımı ve güvenlik gibi temel özelliklerin nasıl yönetileceğini öğrenirler. Ayrıca, .NET Core ile mikroservislerin nasıl tasarlanıp yönetileceğini de anlarlar.

“Microservices Using .NET Core Eğitimi”, modern mikroservis çözümlerinin tasarlama ve yönetme sürecinin tüm temel yönlerini kapsar. Katılımcılar, etkili mikroservisler tasarlamaya ve geliştirmeye başlamadan önce ihtiyaç duyacakları temel becerileri kazanırlar.

Eğitim programı, .NET Core ve mikroservislerin temelleriyle başlar. Katılımcılar, servis tasarımı, performans iyileştirme ve güvenlik gibi temel kavramları öğrenirler. Ayrıca, .NET Core ile mikroservislerin tasarlanması ve yönetilmesi konusunu da kapsar. Katılımcılar tasarım sürecinde .NET’in nasıl bir rol oynadığına dair bilgi sahibi olurlar. Bu bilgiler, katılımcıların mikroservis tasarlama sürecinde ihtiyaç duyacakları temel yapı taşlarını oluşturur.

Eğitimde, .NET Core ve bununla ilişkili temel özellikler ve bileşenler üzerinde duruyoruz. Bu, katılımcılara veri işleme, servis tasarımı ve güvenlik gibi temel yetenekleri kazandırır. Konu akışları, servis tasarımı ve servis yönetimi gibi temel konuları işler.

Son olarak, bir mikroservis çözümünün nasıl .NET Core tabanlı geliştirileceği hakkında bilgi veriyoruz. Bu süreç, ilk olarak çözümün tasarımını, testlerini yapmayı ele alır. Sonrasında ise mikroservis çözümünü anlatır .NET Core tabanlı geliştirilmesini içerir. Bu bilgiler, katılımcıların mikroservis çözümlerini başarılı bir şekilde .NET Core ile tasarlamalarına ve geliştirmelerine yardımcı olur.

Neler Öğreneceksiniz?

Microservices Using .NET Core eğitiminde, katılımcılar şunları öğrenecekler:

- .NET Core framework’ünün temel kavramlarını ve nasıl kullanılabileceğini.
- RESTful API tasarımı ve geliştirme tekniklerini.
- Veritabanı bağlantılarının nasıl yapılacağını ve veri erişim tekniklerini.
- Mikro hizmet tabanlı uygulamaların nasıl dağıtılabileceğini.
- Uygulamaların güvenliğini, performansını ve skalabilitesini sağlama tekniklerini.
- Test ve debugging tekniklerini.

- Mikro hizmetler arasında iletişimi nasıl sağlanabileceğini.
- Mikro hizmetlerin nasıl yapılandırılabileceğini ve yönetilebileceğini.
- Mikro hizmet tabanlı uygulamaların avantajlarını ve dezavantajlarını anlamak.
- Kubernetes veya benzer bir dağıtım ortamında uygulamaları nasıl dağıtabileceğinizi.
- İş akışını ve uygulamalarınızın verimliliğini nasıl iyileştirebileceğinizi.

Bu eğitim, mikro hizmet tabanlı uygulama geliştirme deneyimi olan veya bu alanda ileri seviyede beceri kazanmak isteyen yazılım geliştiricileri için uygundur.

Ön Koşullar

Microservices Using .NET Core eğitimi için ön koşullar şunlardır:

- .NET programlama dilleri (örneğin C#) ve web geliştirme temelleri hakkında bilgi.
- RESTful API ve HTTP protokolü hakkında temel bilgi.
- Veritabanı teknolojileri (örneğin SQL Server) ve veri erişim teknikleri hakkında temel bilgi.
- Terminal veya komut satırı kullanma becerileri.
- Web geliştirme deneyimi (örneğin ASP.NET MVC veya ASP.NET Core).
- Agile yazılım geliştirme metodolojisi ve SCRUM gibi çalışma yöntemleri hakkında bilgi.
- Docker veya benzer bir containerization teknolojisi hakkında temel bilgi.
- Azure veya benzer bir bulut tabanlı platformda uygulama dağıtma deneyimi.

Eğer bu becerilerin bir kısmına sahipseniz, Microservices Using .NET Core eğitiminde daha verimli ve hızlı bir şekilde öğrenme fırsatı bulabilirsiniz.

Kimler Katılmalı

- .NET programlama dilleri ve web geliştirme ile ilgilenen yazılım geliştiricileri.
- RESTful API ve microservices mimarisi konularında daha fazla bilgi edinmek isteyen yazılım geliştiricileri.
- Bulut tabanlı uygulama geliştirme ve dağıtma süreçlerini anlamak isteyen yazılım geliştiricileri.
- .NET Core ve Docker gibi teknolojilerle çalışmayı planlayan yazılım geliştiricileri.

Bu eğitim, yukarıda belirtilen profesyonel hedef kitleye yönelik olarak tasarlandı ve onların microservices mimarisi ve .NET Core ile birlikte çalışma becerilerini geliştirmelerine yardımcı olacaktır.

Outline

Introduction to Containers and Docker

- Understanding VM’s and Containers
- What is Docker?
- Docker Benefits

- Docker Architecture and Docker Taxonomy

Developing Docker Custom Images using .NET Core

- Docker Images for .NET Core
- Executing .NET Core applications in Containers
- Inspecting the Image Architecture
- Developing and Publishing .NET Core Applications
- Dockerfile and Building Docker Images
- Breaking down and understanding dockerfile in-depth

Multi-Stage Builds

- Multiple stages in dockerfile
- Hosting NET Applications in Docker

State and Data in Docker Applications

- Purpose of using Data Volumes
- Access Data in Docker Containers
- Creating a Container with Volumes
- Data volume containers

Docker Compose

- Overview
- Docker-compose features
- Building docker-compose.yml file
- Docker-compose command
- Working with multiple images in a single application
- Environment Variables and Configuration File

Microservices Architecture – Get perfect in concepts and practical

Understanding Microservices

- Understanding Monolithic Architecture
- What are Monolithic Applications
- Deploying
- Containerizing using Docker
- Scaling Applications
- Managing State and Data
- Benefits and Drawbacks of Monolithic Architecture

Microservices Architecture

- What are Microservices
- Monolithic vs Microservices Architecture

- Characteristics of Microservices Architecture
- Benefits of using Microservices Architecture
- Microservices Design Principles

SOA vs. Microservices

Handling Data in Microservices

Communication between Microservices

- Synchronous Communication across Microservices
- Asynchronous communication across Microservices
- API Gateway Pattern
- Microservices Patterns
    - Domain-Driven Design
    - Command and Query Responsibility Separation (CQRS)
    - Event Sourcing
- Creating Composite UI with Microservices
- Drawbacks of Microservices

Setup Microservice Based Application and Perform CRUD Operations

- Creating a Solution and Project Layout
- Implementing a CRUD microservice
- Writing Domain Classes and Controllers
- Data Context Class and Data Seeding
- Using Repository Classes
- Swagger and SwashBuckle Integration
- Practical Demonstration using eStoreApplication
    - Product CatalogService with SQL Server
    - Invoking both services using Swagger UI
    - Implementing Layered Architecture
    - Generic Repository Pattern

Build UI Service

- Adding NET MVC Project
- Writing Model Classes
- Writing Service Classes
- Building Web Controller and Views
- Practical Demonstration using eStoreApplication
    - Writing Backend for Frontend (BFF)
    - UI Microservice to display Product Catalog

Hosting Microservices using Docker Containers

- Adding Docker Support to the Microservice Application

- Creating a Dockerfile file
- Designing and Developing Multi-Container Microservices
- Database Connection string and environment variables in Docker containers
- Handling Configuration Data
- Use a database server running as a container
- Practical Demonstration using eStoreApplication
    - Creating an SQL Server database container
    - Create docker images for Product Catalog Microservices
    - Create a docker image for UI Microservice
    - Writing a YAML for deploying and executing the application

Using Redis Cache in Microservices

- Understanding Redis Cache importance
- Programming Microservice to use Redis Cache
- Consuming Microservice in Web Client
- Practical Demonstration using eStoreApplication
    - Building Shopping Cart Microservice
    - Persisting Cart data in Redis Cache
    - Create a docker image for Shopping Cart Microservice

Understanding OAuth2 and OpenIdConnect

- Authentication and Authorization
- Introduction to Basic Authentication Workflow
- Understanding OAuth
- OAuth Grant Types
- Understanding OpenIDConnect
- Securing Services and Middleware in NET Core
- Using JWT Token to Authenticate and Authorize

IdentityServer4 Server

- What is IdentityServer4
- Authorization Grant Types
- Authorization Code
    - Implicit
    - Client Credentials
    - Resource Owner Password Credentials (ROPC)
- Including UserInfo in ID Token and Access Token

Implementing Security for Microservices

- The Big Picture
- Implementing Identity Microservice
- Using Client Credential Token to Access Microservice

- Using Access Token to call Microservices
- Authentication between Microservices
- Implementing Role-based and Policy-based Authorization
- Practical Demonstration using eStoreApplication
- Adding Authentication Microservice to Solution
    - Securing ProductCatalog Microservice
    - Accessing secure ProductCatalog in Swagger UI
    - Accessing secure ProductCatalog in Client Application

API Gateway Integration

- Introduction to API Gateway
- Understanding Ocelot Middleware
- Integrating API Gateway for Routing
- Handling Secure Microservices in API Gateway
- Practical Demonstration using eStoreApplication
    - Adding API Gateway Service using Ocelot Middleware
    - Updating ocelot configuration for routing to ProductCatalog and ShoppingCart Microservices
    - Deploying Gateway Service in Docker Container

Microservices Communication

- Synchronous Communication using REST API
- Asynchronous Communication using Service Bus
- Integration Events and Event Handlers
- Handing Atomicity and Resiliency when Publishing to EventBus
- Practical Demonstration using eStoreApplication
- Adding Shopping Cart Service with Redis Cache
- Adding Product to Shopping Cart Service for logged-in client
- Create docker images for Product Catalog Microservices
- Asynchronous Communicating with Notification Service and Service Bus Queue

Implementing the CQRS Pattern

- Overview of CRQS Pattern
- Understanding the Command Pattern
- Domain Model and Read Model
- Comparing CQRS with the traditional CRUD approach
- Apply CQRS and CQS approach in DDD microservice
- Practical Demonstration using eStoreApplication
    - Build a New ProductCatalog Service Demo
    - Update ShoppingCart if Product Price changes
    - Update ProductCatalog inventory if the order is placed

Domain-Driven Pattern

- Overview of Domain and Domain-Driven Design
- Layered Architecture in DDD Microservices
- Implementing the Command and Command Handler Pattern
- About Domain Events
- Command and Command Handler Classes
- Practical Implementation of DDD Pattern
- Practical Demonstration using eStoreApplication
    - Implementing DDD using Order Microservices

Handling Failures

- Handle Partial failure
- Implement retries and exponential backoff
- Using Polly policies
- Circuit Breaker Pattern
- Practical Demonstration using eStoreApplication
- Handle Order Service failure
- Handling temporary downtime of SQL Database

Microservices Deployment in App Services

Microservices and Kubernetes

- What is Monolithic Application
- What are Microservices
- Kubernetes Architecture
- Azure Kubernetes Service (AKS)
- Practical Demonstration using eStoreApplication