Course Outline

MOC 20487D: Developing Microsoft Azure and Web Services

Course Summary

Description

In this course, students will learn how to design and develop services that access local and remote data from various sources. Students will also learn how to develop and deploy services to hybrid environments, including on-premises servers and Microsoft Azure.

Objectives

By the end of this course, students will be able to:

- Describe the basic concepts of service development and data access strategies using the .NET platform.
- Describe the Microsoft Azure cloud platform and its compute, data, and application hosting offerings.
- Design and develop a data-centric application using Visual Studio 2017 and Entity Framework Core.
- Design, implement, and consume HTTP services using ASP.NET Core.
- Extend HTTP services using ASP.NET Core.
- Host services on-premises and in Microsoft Azure.
- Deploy services to both on-premises and cloud environments and manage the interface and policy for their services.
- Choose a data storage solution, cache, distribute, and synchronize data.
- Monitor, log, and troubleshoot services.
- Describe claim-based identity concepts and standards, and implement authentication and authorization with Azure Active Directory.
- Create scalable service applications.

Topics

- Overview of service and cloud technologies
- Querying and Manipulating Data Using Entity Framework
- Creating and Consuming ASP.NET Core Web APIs
- Extending ASP.NET Core HTTP Services
- Hosting Services On-Premises and in Azure
- Deploying and Managing Services
- Implementing Data Storage in Azure
- Diagnostics and Monitoring
- Securing services on-premises and in Microsoft Azure
- Scaling Services

Audience

This course is designed for .NET developers who want to learn how to develop services and deploy them to hybrid environments and .NET developers with Web application development experience who are exploring developing new applications or porting existing applications to Microsoft Azure.

Course Outline

MOC 20487D: Developing Microsoft Azure and Web Services

Course Summary

Prerequisite

Before attending this course, students must have:

- Experience with C# programming, and concepts such as lambda expressions, LINQ, and anonymous types
- Understanding the concepts of n-tier applications
- Experience with querying and manipulating data with ADO.NET

Duration

Five Days

Course Outline

MOC 20487D: Developing Microsoft Azure and Web Services

Course Outline

I. ***Overview of service and cloud technologies***

This module provides an overview of service and cloud technologies using the Microsoft .NET Core and the Azure. The first lesson, “Key Components of Distributed Applications,” discusses characteristics that are common to distributed systems, regardless of the technologies they use. Lesson 2, “Data and Data Access Technologies” describes how data is used in distributed applications. Lesson 3, “Service Technologies,” discusses two of the most common protocols in distributed system and the .NET Core technologies used to develop services based on those protocols. Lesson 4, “Cloud Computing,” describes cloud computing and how it is implemented in Azure.

A. Key Components of Distributed Applications
B. Data and Data Access Technologies
C. Service Technologies
D. Cloud Computing
E. Manipulating Data

- Lab : Exploring the Work Environment
  - Creating an ASP.NET Core project
  - Create a simple Entity Framework model
  - Create a web API class
  - Deploy the web application to Azure

II. ***Querying and Manipulating Data Using Entity Framework***

In this module, you will learn about the Entity Framework data model, and about how to create, read, update, and delete data. Entity Framework is a rich object-relational mapper, which provides a convenient and powerful application programming interface (API) to manipulate data. This module focuses on the Code First approach with Entity Framework.

A. ADO.NET Overview
B. Creating an Entity Data Model
C. Querying Data

- Lab : Creating a Data Access Layer using Entity Framework
  - Creating a data model
  - Query the Database
- Lab : Manipulating Data
- Create repository methods
- Test the model using SQL Server and SQLite

.

III. ***Creating and Consuming ASP.NET Core Web APIs***

ASP.NET Core Web API provides a robust and modern framework for creating Hypertext Transfer Protocol (HTTP)-based services. In this module, you will be introduced to the HTTP-based services. You will learn how HTTP works and become familiar with HTTP messages, HTTP methods, status codes, and headers. You will also be introduced to the Representational State Transfer (REST) architectural style and hypermedia. You will learn how to create HTTP-based services by using ASP.NET Core Web API. You will also learn how to consume them from various clients. After Lesson 3, in the lab "Creating an ASP.NET Core Web APIs", you will create a web API and consume it from a client.

A. HTTP Services
B. Creating an ASP.NET Core Web API
C. Consuming ASP.NET Core Web APIs
D. Handling HTTP Requests and Responses

Course Outline (cont.)

E. Automatically Generating HTTP Requests and Responses
- Lab : Creating an ASP.NET Core Web API
  - Create a controller class
- Use the API from a browser
  - Create a client

IV. *Extending ASP.NET Core HTTP Services*

ASP.NET Core Web API provides a complete solution for building HTTP services, but services often have various needs and dependencies. In many cases, you will need to extend or customize the way ASP.NET Core Web API executes your service. Handling needs such as applying error handling and logging integrate with other components of your application and supporting other standards that are available in the HTTP world. Understanding the way ASP.NET Core Web API works is important when you extend ASP.NET Core Web API. The division of responsibilities between components and the order of execution are important when intervening with the way ASP.NET Core Web API executes. Finally, with ASP.NET Core Web API, you can also extend the way you interact with other parts of your system. With the dependency resolver mechanism, you can control how instances of your service are created, giving you complete control on managing dependencies of the services.

A. The ASP.NET Core Request Pipeline
B. Customizing Controllers and Actions
C. Injecting Dependencies into Controllers
- Lab : Customizing the ASP.NET Core Pipeline
  - Use Dependency Injection to Get a Repository Object
  - Create a Cache Filter
  - Create a Debugging Middleware

V. *Hosting Services On-Premises and in Azure*

In this module you will learn how to host your application on-premises and on Azure. You will also learn about Docker containers, and writing serverless applications with Azure functions.

A. Hosting Services on-premises
B. Hosting Services in Azure App Service
C. Packaging Services in Containers
D. Implementing Serverless Services
- Lab : Host an ASP.NET Core service in a Windows Service
  - Creating a new ASP.NET Core Application
  - Registering the Windows Service
- Lab : Host an ASP.NET Core Web API in an Azure Web App
  - Create a Web App in the Azure portal
  - Deploy an ASP.NET Core Web API to the Web App
- Lab : Host an ASP.NET Core service in Azure Container Instances
  - Publish the service to a Docker container
  - Host the service in Azure Container Instances
  - Lab : Implement an Azure Function
  - Develop the service locally
  - Deploy the service to Azure Functions

Course Outline

MOC 20487D: Developing Microsoft Azure and Web Services

Course Outline (cont.)

VI. ***Deploying and Managing Services***

In this module, you will learn about Web Deploy and how to deploy web applications by using Web Deploy in Visual Studio. You will also learn how to define continuous integration and continuous delivery pipelines and how to use Azure API Management and OpenAPI to provide robust, secure, and reliable APIs to your customers.

A. Web Deployment with Visual Studio 2017
B. Continuous Delivery with Visual Studio Team Services
C. Deploying Applications to Staging and Production Environments
D. Defining Service Interfaces with Azure API Management

- Lab : Deploying an ASP.NET Core web service on Linux
  - Publish the ASP.NET Core web service for Linux
  - Configure Nginx as a reverse proxy
- Lab : Deploying to Staging and Production
  - Deploy the application to production
  - Create a staging slot
  - Swap the Environments
- Lab : Publishing a Web API with Azure API Management
  - Creating an Azure API Management instance
  - Testing and managing the API

VII. ***Implementing Data Storage in Azure***

This module explains how to store and access data stored in Azure Storage. It also explains how to configure storage access rights for storage containers and content.

A. Choosing a Data Storage Mechanism
B. Accessing Data in Azure Storage
C. Working with Structured Data in Azure
D. Geographically Distributing Data with Azure CDN
E. Scaling with Out-of-Process Cache

- Lab : Storing Files in Azure Storage
  - Store publicly accessible files in Azure Blobs
  - Generate and store private files in Azure Blobs
- Lab : Querying Graph Data with CosmosDB
  - Create the CosmosDB graph database
  - Query the CosmosDB database
- Lab : Caching out-of-process with Azure Redis cache
  - Create the Azure Redis Cache service
  - Access the cache service from code
  - Test the application

VIII. ***Diagnostics and Monitoring***

This module explains how to monitor and log services, both on-premises and in Azure.

A. Logging in ASP.NET Core
B. Diagnostic Tools
C. Application Insights

- Lab : Monitoring ASP.NET Core with ETW and LTTng
  - Collect and view ETW events
  - Collect and view LTTng events
- Lab : Monitoring Azure Web Apps with Application Insights
- Add the Application Insights SDK
- Load test the web service
- Analyze the performance results

IX. ***Securing services on-premises and in Microsoft Azure***

This module describes claim-based identity concepts and standards, and how to implement authentication and authorization by using Azure Active Directory to secure an ASP.NET Core Web API service.

Course Outline (cont.)

A. Explaining Security Terminology
B. Securing Services with ASP.NET Core Identity
C. Securing Services with Azure Active Directory

- Lab : Using ASP.NET Core Identity
  - Add ASP.NET Core Identity middleware
  - Add authorization code
  - Run a client application to test the server
- Lab : Using Azure Active Directory with ASP.NET Core
  - Authenticate a client application using AAD B2C and MSAL.js

X. ***Scaling Services***

This module explains how to create scalable services and applications and scale them automatically using Web Apps load balancers, Azure Application Gateway and Azure Traffic Manager.

A. Introduction to Scalability
B. Automatic Scaling
C. Azure Application Gateway and Traffic Manager

- Lab : Load Balancing Azure Web Apps
  - Prepare the application for load-balancing
  - Test the load balancing with instance affinity
  - Test the load balancing without affinity
- Lab : Load Balancing with Azure Traffic Manager
  - Deploy an Azure Web App to multiple regions
  - Create an Azure Traffic Manager profile