Introduction to ASP.NET Core MVC

Duration: 28 hours

Prerequisites: Previous C# programming experience.

Course Description: The ASP.NET Core MVC training course is designed to provide an introduction to .NET Core for programmers who are already familiar with the C# language. The course focuses on core portions of the .NET Framework that are common across many application areas. This .NET Core course starts with an introduction to the architecture and key concepts of .NET, and then discusses class libraries, packages, metapackages and frameworks. Coverage includes working with delegates and events, I/O and serialization, memory management, threading and an introduction to the Task Parallel Library (TPL).

The course also provides a practical hands-on introduction to developing Web applications using ASP.NET Core MVC 6 and C#. This Web development framework from Microsoft emphasizes the separation of concerns in the architecture and testability of applications. This .NET Core course covers the fundamentals of the Model-View-Controller design pattern and its implementation in ASP.NET Core MVC. For lab work, Visual Studio 2019 with ASP.NET Core 3.0 is used as a productive platform for creating MVC Web applications.

After presenting the fundamentals of the technology with several examples, the main components of Model, Controller and View are covered in detail. The discussion of the Model incorporates Microsoft technologies for persisting data, which includes XML Serialization and ADO.NET with SQL Server. The routing mechanism of ASP.NET MVC is also covered, and the course includes an introduction to ASP.NET Web API as well.

Students Will Learn

- Gaining a thorough understanding of the philosophy and architecture of .NET Core
- Understanding packages, metapackages and frameworks
- Acquiring a working knowledge of the .NET programming model
- Implementing multi-threading effectively in .NET applications
- Gaining a thorough understanding of the philosophy and architecture of Web applications using ASP.NET Core MVC
- Gaining a practical understanding of .NET Core
- Acquiring a working knowledge of Web application development using ASP.NET Core MVC 6 and Visual Studio 2019
- Persisting data with XML Serialization and ADO.NET with SQL Server
- Creating HTTP services using ASP.NET Core Web API
- Deploying ASP.NET Core MVC applications to the Windows Azure cloud

Overview

.NET Fundamentals

Class Libraries

What is Microsoft .NET?
- Common Language Runtime
- Framework Class Library
- Language Interoperability
- Managed Code
- .NET Core and Cross-Platform Development

Packages and Frameworks

- Overview of NuGet Packages and the NuGet Gallery
- Explaining the Role of Packages, Metapackages and Frameworks in .NET
- How Packages and Metapackages are Used in .NET Core
- Using the Visual Studio Package Manager to Manage Packages in Solutions
- Installing Packages from the NuGet Gallery
- Creating and Using Your Own NuGet Packages
- Porting from .NET 4.6 to .NET Core

Delegates and Events

- Using Delegate Objects to Implement Callbacks
- Using the `Random` Class to Generate Random Test Data
- Using Aggregations of Delegate Objects
- Using Delegate Objects to Implement and Handle Event Notifications

.NET Threading

- Using the `Thread` Class to Implement Multithreading in .NET Applications
- Using the `Monitor` Class to Program Safe Concurrent Access to Shared Data
- Using the `ThreadPool` Class to Obtain Threads from a Pool that is Managed by the System
- Difference Between Foreground and Background Threads
- Overview of Different Classes that Can be Used for Synchronizing Threads
- Using the Task Parallel Library to Implement Task Parallelism and Data Parallelism in .NET Applications

Introduction to ASP.NET Core MVC

- Understanding How ASP.NET Core MVC is Used Within Visual Studio
- Creating Several Versions of a Simple ASP.NET Core MVC Application
- Understanding How Views are Rendered
- Using the Razor View Engine in ASP.NET Core MVC

Components in .NET
- Comparing Components in COM and .NET
- Creating Class Libraries
- Using Components in Client Programs by Obtaining References to Class Libraries
- Using References

I/O and Serialization

- Using .NET Framework Classes for Working with Directories and Files
- Explaining and Using Streams for Performing File I/O
- Explaining the Role of Serialization in Persisting and Transporting Objects
- Serialization Mechanisms Available in .NET Core Contrasted with Classical .NET
- Using XML Serialization in .NET Core Programs

.NET Programming Model

- Garbage Collection in .NET
- Implementing the Finalize/Dispose Pattern
- Using the Process and Thread Model for .NET Applications
    - Process and Thread Isolation
- Using the `Process` Class to Manage Processes in .NET Core Applications
- Working with Command-Line Arguments

Overview of ASP.NET MVC

- Advantages and Disadvantages of ASP.NET Web Forms
- Understanding the Model-View-Controller (MVC) Pattern
- Outlining the Parts of an ASP.NET MVC Application
- Advantages and Disadvantages of ASP.NET MVC
- Overview of ASP.NET Core
- Understanding the Use of Unit Testing in Creating ASP.NET MVC Applications

ASP.NET MVC Architecture

- Understanding the Controller in ASP.NET MVC and its Role
- Understanding the View in ASP.NET MVC and its Role
- Understanding the Model in ASP.NET MVC and its Role
- Use of Helper Methods for HTML in ASP.NET MVC

- Understanding How Dynamic Output Works
- Passing Input Data to an MVC Application in a Query String
- Rendering Views
- How Form Submission, Model Binding and Input Validation Work in ASP.NET MVC

The Model

- Working with More Complex Models in MVC Programs
- Overview of Various Microsoft Technologies Available to Use and Persist the Model
  - ADO.NET
  - LINQ
  - ADO.NET Entity Framework and LINQ to Entities
  - XML
- Using XML Serialization Technique to Persist a Complex Model
- Using the NuGet Package Manager and Use it to Install a New Package
- Persisting a Model Using ADO.NET

The Controller

- Facilities Provided by the `Controller` Base Class
  - Action Methods
  - Action Results
  - Filters
- Overview of How Controllers Can Receive Input
- Principal Types of Output from a Controller
- Using Attributes to Control How Actions are Invoked
- How Asynchronous Controllers Work and How they Can be Used

The View

- Responsibility of the View
- Using ViewBag as a Dynamic Type for Passing Data from a Controller to a View
- How Dynamic Objects Can be Used in ASP.NET Core MVC
- Using HTML Helpers
  - String Helpers
  - Link-Building Helpers
  - Form Helpers
  - Validation and Templated Helpers
- Using Validation Attributes in the Model

Routing

- Overview of the Use of Routing in ASP.NET Core MVC
- Understanding the Properties of Routes in ASP.NET Core MVC
- Understanding the Use of Parameters in Routing
- Understanding How to Register Routes and the Importance of Order
- Using Attribute Routing to Map Actions Directly to Route Templates by Means of Attributes

ASP.NET Core Web API

- Overview of the ASP.NET Web API
- Overview of RESTful Web Services
- Implementing HTTP Services Using the Web API with ASP.NET Core MVC
- Using Postman to Exercise HTTP Requests while Developing a Web API Service
- Implementing Web API clients

ASP.NET Core and Azure

- What Is Windows Azure?
- A Windows Azure Testbed
- Deploying an Application to Azure
- Updating an Application on Azure

Related Bootcamp

Track	Duration	Price
Microsoft .NET Developer: C#	5-course track 6-course track	$6,000 $7,200

	7-course track	$8,400
	8-course track	$9,600
	9-course track	$10,800

Contact Us

Address: 1 Village Square, Suite 3 Chelmsford, MA 01824

Phone: 978.250.4983

Mon - Thur: 9 am - 5 pm EST
Fri: 9 am - 4 pm EST

E-mail: info@developer-bootcamp.com