Dot Net Framework, C#, ASP.NET, ASP.NET MVC, Data Access

## Course Description:

## This course will help you to learn what Dot Net Framework is. Not only that, during the course, participants will be learning one of the many programming languages supported by .NET Framework and that is C#. After learning the language, participants will be able to build web applications using ASP.NET technology and later in the course, using ASP.NET MVC technology. Also, data access technology in .NET Framework will be discussed briefly, so that participants can access from the applications. Throughout the entire application participants will use LINQ (Language Integrated Query) along with Entity Framework (EF) for data access.

## Duration:

## 10 Days

## Pre-requisites:

* Participants should have working knowledge on any of the object oriented programming language, such as, C++ or Java as well as basic understanding of how web applications work, i.e., HTML, CSS, JavaScript (additionally jQuery, JSON etc. will help).

## Recommended Learning Resources:

* <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/index>
* <https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx>
* <https://www.asp.net/mvc>
* <https://www.asp.net/>

## Software & Hardware requirements:

* S/W:
  + Visual Studio 2013/2015 Ultimate Version
  + SQL Server 2012 (Professional/Enterprise Edition)
  + Internet Information services (IIS) as web server
* H/W:
  + Windows 8/8.1/10 (windows 10 is preferable) OS
  + Minimum 100GB HDD (min 60GB for C drive)
  + LAN/WiFi connection

## High Level Course Contents

* .NET Framework
* C# programming language
* C# basics
* OOP in C#
* Exception handling
* Collections and generics
* Advanced C# topics, like serialization, file handling etc.
* MVC architecture
* ASP.NET MVC
* Creating controller, view and model in ASP.NET MVC
* Different types of views
* Controller actions and action results
* Accessing data and passing through controller to view
* Layout, ViewStart files in ASP.NET MVC
* Models and Model Binding mechanism
* Model Validation
* Filters in ASP.Net MVC
* ASP.NET Web API
* .NET Core Framework
* ASP.NET MVC Core applications – web page and Web API applications
* EF Core

Day wise coverage:

**Course Contents**

**Day 1:**

**Introduction to .NET Framework**

* .NET Framework-Introduction
* History of versions - 1.0, 1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.6, 4.7
* .NET Framework Application Execution Cycle, CLS and CTS
* Assembly Brief Idea, Components of Assembly
* Viewing Assembly Components - ILDASM tool
* Advantages of ,NET Framework

**Visual Studio:**

* Introduction to Visual Studio
* How to create a simple C# application using Visual Studio 2012/2013/2015/2017
* Using different windows: a. Solution Explorer, b. Server Explorer, c. Properties Window
* Using "Find all references", refactoring, generating method stubs, alignment of code, organize usings etc.,
* Debugging Features in VS, Using debugging windows: a. Autos window, b. Locals window, c. Watch and Quickwatch windows, d. Immediate and Command Window
* Designing class using class diagram, Testing feature in VS, Compiling and Running application in VS
* Versioning of an assembly
* Security of an assembly - strong naming, Creating public/private key pair using SN tool/VS feature and linking them to assembly, GAC, Shared Assesmbly, using GACUTIL tool

**Introduction to C#**

* Introduction to C#, Features of C# language
* Data Types in C#, Control flow statements in C#, Operators in C#
* Writing methods in C#, Keywords - ref, out
* Writing a basic C# application, featuring all the previously discussed topics and compiling and running the same, debugging the previous application
* Namespaces, Boxing and Unboxing, using checked and unchecked

**Introduction to OOP in C#**

* Class and objects, How to create a class in C# using VS
* OOP Principles – Abstraction
* Members of a class - Fields (data members) and methods (functionalities), Access specifiers of class (internal and public) and class members (public, private, protected, internal and protected internal)
* static and non-static members of a class
* Declaring constructors, static constructor
* Properties in C#
* Readonly and WriteOnly properties, static properties
* Using "Refactor-->Encapsulate filed" fetaure, using "propfull" to auto-generate properties

**Arrays in C#:**

* Arrays in C#
* Declaring one and multi-dimensional arrays
* creating Jagged arrays (array of arrays)
* using "params" keyword - Param array

**Day 2:**

**Deep Dive in OOP in C#**

* OOP principles - Inheritance
* Aggregation and Specialization
* Creating base and derived class
* Constructor execution sequence
* using " base" and "this" keyword and "sealed" keyword in case of inheritance
* OOP Principles - Polymorphism
* Different types of Polymorphism - Compile time and runtime polymorphism
* Compile time polymorphism - Method overloading and Operator overloading
* Runtime Polymorphism - having same method in base and derived class
* using "new" keyword for shadowing
* using "virtual" and "override" keywword for dynamic polymorphism
* Abstract class
* Creating abstract class
* Creating abstract class and abstract methods/properties in C#
* How to derive and implement abstract members of abstract class in derived class - use VS faclities to avoid manual implementation in derived class
* "sealed" keywrod in case of overridable methods
* Interface
* Declaring interface
* Do's and Don'ts of interfaces
* Implicit and explicit implementation of interface members
* Implicit and explicit invocation of interface members
* Implemeting interface members - using VS facility rather than manual typing

**Memory Management**

* Memory Management in .Net (Resource Management)
* Managed Heap v/s Unmanaged Heap
* Managed v/s Unmanaged Resources
* Object Dispose Pattern

**Dependency Injection**

* SOLID Principles
* Dependency Injection
* Types of Dependency Injection

**Day 3:**

**Generics and collections and Generic Collection Classes:**

* What is Generics?
* Generic class, interface and methods
* Template v/s .Net Generics
* Type Conditions in Generics
* Indexer
* Collections and Generic collection classes
* Advantages of Collections and Generic collection classes over array
* Different collection interfaces and classes - ArrayList, Stack, Queue, HashTable, SortedList, DictionaryEntry classes and IEnumerable, ICollection, IList, IEnumerator interfaces
* Generic classes
* Advantages of Generic collection classes over regular collection classes
* Different Generic classes and interfaces: List<T>, Stack<T, Queue<T>, Dictionary<TKey,TValue>, SortedList<TKey,TValue>, KeyValuePair<TKey,TValue>, HashSet<T> classes and IEnumerable<T>, ICollection<T>, IList<T>, IEnumerator<T> interfaces
* Internalization of Comparison - Implementing CompareTo method of IComparable and IComparable<T> interface in class to provide compariosn logic between two objects
* Externalization of Comparison - Implementing Compare method of IComparer and IComparer<T> interface in class to provide compariosn logic between two objects
* using Sort method to sort elements in List<T> class
* HashSet<T> for values types and reference types

**Day 4:**

**Delegates and Events:**

* Delagets and Events
* single-cast and multi-cast delegate
* anonymous method and anonymus delegate
* Generic delegates
* in built delegates: Action<T> and Func<T> and Predicate<T>
* Events

**New Features in C#**

* Implicitly Typed varibales and array
* anonymous types
* Auto-implemented properties
* Object initializer and collection initializer
* Extension methods
* Partial class and methods
* Default value and optional arguments
* named arguments
* Lambda Expression
* dynamic type

**Introduction to LINQ:**

* LINQ - introduction
* Advantages of LINQ
* LINQ query operator syntax
* LINQ Method query syntax
* Role of Delegates and Collections in LINQ
* Using different namespaces
* LINQ against object
* LINQ against SQL
* LINQ against XML
* LINQ operators: select, where, group by, join etc.
* Inserting, updating, deleting and selecting data
* Using Join query to link two or more tables
* Performing group by on the data

**File Handling:**

* File handling
* System.IO namespace and different classes
* File, FileInfo, Directory, DirectoryInfo, Path
* Stream classes: Stream, MemoryStream, BufferedStream
* Reader and Writer classes: TextReader, TextWriter, StreamReader and StreamWriter

**Day-5**

**Internals of CLR**

* AppDomain
* Cross Domain Communicataion
* Object Serialization
* Reflection
* Attributes (AOP)

**Concurrent Programming: Threads**

* Concurrency programming in .Net (Threads)
* Managed Thread v/s Process Thread
* Application Thread v/s Background Thread
* Thread Lifecycle
* Thread Synchronization
* Thread class
* ThreadPool class

**The Task Parallel Library**

* Task based programming
* Parallel Programming in the .NET Framework
* The Task Parallel Library (TPL)
* Data Structures for Parallel Programming
* Async and await

**Day 6:**

**RDBMS and MS-SQL SERVER**

* SQL Server Overview
* Describe the main database objects
* Create tables
* Describe the data types that can be used when specifying column definition
* Alter table definitions
* Drop, rename, and truncate tables
* Stored procedures
* User-defined functions and in-built functions
* Triggers
* Indexing
* Views

**Day-7:**

**Introduction to ADO.NET and Entity Framework**

* Understanding Connected Approach
* Insight on Connection class, Command class and DataReader class
* Understanding ConnectionString Property
* Working with Parameters of Command class
* Understanding use of ExecuteNonQuery method, ExecuteReader method, ExecuteScalar method.
* Working with using clause
* Understanding Disconnected Approach
* Insight on DataSet class, DataTable class, DataColumn class and DataRow class
* Working with DataAdapter class
* What is EF?
* Architecture of EF: OR/M (Object Relational Mapping) and Services
* Advantages of EF
* Different approcahes in EF - Database first, Model first and Code First
* Creating a OR/M (EDMX file)
* Understanding the file structure in EF - CSDL, SSDL and MDL
* Types of Entity: POCO and Dynamic Proxy
* DbConext, DbSet<T> class
* Performing basic CRUD operations against database using ADO.NET Entity Framework
* Using stored procedures in EF

**Day 8:**

**Introduction to ASP.NET MVC**

* Goals of Modern Web Development
* ASP.NET MVC
* Model-View-Controller Design Pattern

**Routing**

* Configuration
* • RouteValueDictionary
* • Route Constraints
* • Catch-All Parameter
* Ignoring Routes
* Conventions
* Request Pipeline
* Controller Class Hierarchy
* ActionResult

**Views**

* Implementation
* View Engines
* Razor Syntax
* View Templates
* Layouts

**Helpers**

* HTML Helpers
* Html.ActionLink
* Url.Action
* Strongly-Typed Helpers
* Custom Helpers
* Inline Razor Helpers
* Hands-On Lab Exercise

**Strongly-Typed Views**

* Creating strongly typed views
* Passing Model Objects

**HTML Forms**

* Helpers
* Hands-On Lab Exercise
* Action Selectors
* Model Binding

**Day-9:**

**Partial Views and Child Actions and Layouts**

* + Partial Views
  + Child Actions
  + Layouts

**Models**

* + Object-Relational Mapping
  + using EF in MVC

**Data Validation**

* + Validation Attributes
  + ModelState
  + IValidatableObject

**Action Filters**

* OutputCache
* HandleError
* RequireHttps
* Authorization
* Global Filters
* Custom Filter

**Web API 2**

* ApiController
* Action Parameters
* Exceptions
* Configuration
* Routing
* Model Binding
* Formatters
* Status Codes

**Using jQuery in MVC application**

* Why to make use of jQuery?
* Using AJAX methods using jQuery.
*  $.get
*  $.post
*  $.getJSON
*  $.ajax

**Security**

* Authentication

**Day-10:**

**.NET Core Framework**

* Introduction .NET Core
* .NET Core, .NET Standard and .NET Framework
* Version history of .NET Core
* .NET Core runtime and Roslyn compiler
* Creating different types of application – console, web etc.
* .NET CLI (Command Line interface tool)
* Creating console application, compiling and executing
* Dependency injection in .NET Core

**ASP.NET MVC Core Applications**

* ASP.NET Core MVC applications
* Difference between ASP.NET MVC and ASP.NET MVC Core application
* Configuration of ASP.NET Core MVC application
* Using middleware in the Core MVC pipeline
* Creating ASP.NET Core MVC Web API application
* EF Core and its features
* Create an multi-tire application using ASP.NET Core Web API, EF Core and front-end UI (preferably ASP.NET Core MVC Webpage/ASP.NET MVC Web Page application)

**Miscellaneous:**

* Debugging
* Best Practices
* Unit Testing in VS
* Share code using Git