.NET Course

Course description

This fast-paced instructor-led online five-day course covers application development for the .NET framework.

Web and desktop applications are assembled, using Entity Framework Core to implement the data layer with ASP.NET Core MVC, Web API, React and WPF MVVM for the presentation layers. Other topics include the collections classes, SQL, LINQ, test driven development, dependency injection, concurrent programming and design patterns.

While we've currently suspended classroom training, we're using video conferencing and remote desktop software, enabling the instructor to view your code and assist with exercises remotely. Course code can be viewed on a Git repository.

Customised Training

We can provide customised training for a team or for individuals, starting at £400+VAT per day. We have been training Developers Since 1999. Call us today to discuss how we can train your staff to build and maintain .NET applications.

What you will learn

- Programming in C#
- Test Driven Development (view a 20 minute tutorial video)
- Collections and LINQ (view a 10 minute tutorial video)
- SQL and Entity Framework Core
- Web API and React
- ASP.NET Core MVC
- Windows Presentation Foundation
- Design Principles and Patterns

Contact

To discuss training options or to make a booking, please telephone 0118 966 4994 or email mailbox@javaconsult.co.uk

Course instructor

Originally a Civil Engineer with a BSc from London University the course instructor, Simon Dineen, has 20 years' experience in .NET training and development.

Course outline

1. Programming in C#

Review of C# fundamentals. This is intended as a recap for developers who are already familiar with object-oriented programming but may not be up to date with best practices and new additions to the language.

- Overview of the .NET Core Framework including the CLR, CTS and CLS
- Syntax including data types, operators and statements
- Class design including UML, encapsulation, expression-bodied members, auto-implemented properties, overloaded constructors, inheritance and overriding, abstract classes, structs
- Handling and throwing exceptions; recursive methods
- Passing parameters by reference with the in, out and ref keywords
- Unit testing with xUnit, including parameterised tests and substituting dependencies for mock objects with the Moq library.

2. Collections

Comparing classes and interfaces in the collections framework

- The collections interface hierarchy with IList and IDictionary implementations. Polymorphism and numeric formatting.
- Using FIFO and LIFO collections including Stack, Queue and LinkedList
- ISet implementations including HashSet and SortedSet. Overriding Equals and GetHashCode, implementing IComparer
- Relative speeds of collection methods and Big O notation
- Writing a generic class that implements ICollection<T>, using the Array class. Using the yield keyword to get an Enumerator

3. LINQ

Using Language-Integrated Query to interrogate collections

- Using LINQ Query syntax
- Understanding LINQ method syntax, including extension methods, generic delegates, lambda expressions, covariance and contravariance
- Filtering, selecting and aggregation operations
- Building a repository class, using a HashSet persisted to a file with JsonSerializer. Unit testing the repository with XUnit and Moq.

4. SQL

Communicating with a SQL Server database

- Transact-SQL types. Creating tables, insert, update, delete and select statements. Primary and foreign keys, table joins
- Getting a connection, IDisposable types and using statements. Transactions; pessimistic and optimistic concurrency control
- Building repository classes and writing integration tests

5. Entity Framework

Object-relational mapping with Entity Framework Core

- Configuring a DbContext, data seeding. Test-driven development
- Entity classes, generated values, foreign key properties, data annotations, optimistic concurrency control
- Managing database schemas, using migrations to sync the data model with the database; reverse engineering
- Querying with LINQ, eager and lazy loading of related data
- Saving data; entity state; adding, modifying and removing tracked entities
- Asynchronous methods, the Task class, using the await keyword, starting a background thread with Task.Run
- Building a Service Layer for separation of business logic (services) and data-access logic (repositories)

6. Web API

Creating RESTful services with ASP.NET Core

- Controller classes; HTTP methods and status codes; route mapping and constraints
- Configuring dependency injection; CORS; debugging with Postman
- Unit testing the controller with xUnit and Moq
- Generating a SHA256 hashed password
- Claims-based authentication, symmetric encryption, generating a JSON Web Token (JWT), authenticating the JWT on the server
- Writing controller methods that call the service layer, including the JWT in the HTTP header
- Creating a release build and deploying the service

7. React

A JavaScript library for building user interfaces

- Understanding toolchains, packages and modules. Using Visual Studio Code with Node Package Manager to create a single page app
- Javascript syntax review including data types, operators, variable declaration and scope, control structures, objects, constructors, arrays, inline and arrow functions
- JSX. Components and props. Handling events
- State and lifecycle with state and effect hooks; the virtual DOM; debugging with React developer tools
- Conditional rendering, extracting components into a hierarchy, lifting shared state; HTML forms and controlled components
- Generating HTML lists and tables; array functions including map and filter; for loops
- Connecting to a REST service; promise objects and async functions
- Authentication; obtaining a JSON Web Token and including the token in HTTP requests
- Creating a production build and deploying the application

8. ASP.NET Core MVC

A cross-platform open-source framework for building cloud-based applications

- Using the MVC design pattern to decouple the user-interface, data and application logic
- Controllers, views and Razor syntax. HTTP methods and status codes
- Configuring dependency injection and routing
- Scaffolding views; tag helpers; layout pages; Bootstrap
- Razor pages and model binding
- Claims-based authentication with ASP.NET Core Identity. Scaffolding Register and Login pages
- Writing controller methods that call the service layer. Data annotations and model state. client-side validation with jQuery. Preventing CSRF attacks
- Unit testing the controller with xUnit and Moq
- Creating a release build and deploying the application

9. WPF

Windows Presentation Foundation is a framework for building desktop applications on Windows

- XAML controls and layout panels; delegates and events; styles and resources
- Data binding; dependency properties; type converters; the DataGrid control
- Using the Model-View-ViewModel (MVVM) pattern to separate presentation and business logic from the user interface. Configuring dependency injection
- Calling the Web API asynchronously; serializing and deserializing JSON
- Binding control properties to the ViewModel and sending PropertyChanged events
- Authentication; obtaining a JSON Web Token and including the token in HTTP requests to the Web API
- Unit testing the controller with xUnit and Moq

10. Design Principles and patterns

Design principles are intended to make software more understandable, flexible and maintainable. Design patterns are reusable solutions to commonly occurring problems. The following are discussed in the context of the applications built in the previous modules

- SOLID principles (Single responsibility principle; open-closed, Liskov Substitution Principle; Interface Segregation principle; Dependency inversion)
- Don't repeat yourself; encapsulate what changes; favour composition over inheritance; program to an interface, not to an implementation
- Creational design patterns: abstract factory; builder; factory method; singleton
- Structural design patterns: adapter, decorator, façade
- Behavioural design patterns: observer; strategy; template method