C# 5.0 Programming in the .NET Framework

6 days Course

Course Description

This six-day instructor-led course provides students with the knowledge and skills to develop applications in the .NET Framework 4.5 using the C# 5.0 programming language. C# is one of the most popular programming languages in existence, and the C# 5.0 revision introduces new productivity, performance, and convenience features into the language. This course features an overview of all language-related features, as well as an introduction to general .NET Framework features such as garbage collection, assembly loading, Reflection, Language-Integrated Query (LINQ), Asynchronous prgramming and many others.

Intended Audience

This course is intended for developers with good knowledge of object-oriented principles and practical experience of at least 6 months with an object-oriented programming language (C++ preferred).

Prerequisites

- good knowledge of object-oriented principles and practical experience of at least 6 months with an object-oriented programming language (C++ preferred)
- For student who has experience with non-object oriented programming language must take the 1 day course introduction to object oriented, usually this day course preformed before this course in few days

Objectives

- Develop applications using the C# 5.0 language in the .NET Framework 4.5
- Use generic types and implement generic algorithms to improve application performance and reliability.
- Apply object-oriented architecture and design principles to .NET applications written in C#, and combine them with functional programming fundamentals.
- Use attributes and reflection for metadata-driven or aspect-oriented software development.
- Employ Language-Integrated Query (LINQ) syntax and classes to declaratively implement datadriven applications.
- Deploy, version, configure and register .NET assemblies and applications.

Reading

- New to C# Development, MSDN (http://msdn.microsoft.com/en-us/vcsharp/aa336768.aspx)
- Visual C# Developer Center, MSDN (http://msdn.microsoft.com/en-us/vcsharp/default.aspx)

Topics

- **Module 1: Introduction to the .NET Framework**
  - Introduction to the .NET Framework
  - Common Language Runtime Components – Garbage collector (GC), Common Type System (CTS), Just-in-Time compiler (JIT)
  - An Overview of Managed Languages
  - Microsoft Intermediate Language (IL)

- Native Image Generator (NGEN)
- An Overview of the Framework Class Library (FCL)
- .NET Version Evolution – from .NET 1.0 to .NET 4.5

• Module 2: Introduction to C# 5.0

- C# 5.0: Overview and Design Goals
- The Visual Studio Integrated Development Environment
- "Hello World" in C#
- Namespaces and References – Importing types, multi-targeting support, target platform
- Console Operations
- String Formatting
- Disassembling .NET – ILDASM, .NET Reflector
- Lab 1: Basic Operations
  - Simple console operations
  - String output formatting

• Module 3: The .NET Type System

- The Common Type System
- The Common Language Specification
- Primitives and Built-in Types
- Value Types and Reference Types
- Boxing and Unboxing
- System.Object Class Members
- Type Conversions
- Lab 2: Reviewing Reference Types and Value Types
  - Class exercise – comparing operations on value types and reference types
- Lab 3: Reviewing Object Equality
  - Class exercises – comparing equality operations on value types and reference types

• Module 4: C# Classes

- Class Members
- Access Modifiers
- Nested Types
- Fields
- Constructors and Static Constructors
- Constants and Readonly Fields
- Properties and Automatic Properties
- Object Initializer Syntax
- Methods and Static Methods
- Optional and Named Parameters
- Static Classes
- Extension Methods
- Partial Types and Partial Methods
- The new Operator
- Parameter Modifiers
- Variable Parameter Lists
- The Entry Point and its Parameters

- Destructors
- Lab 4: Basic Class
  - Rectangle class – methods, static methods, fields, properties
  - Linked list, partial methods and extension methods
  - Using optional and named parameters in a Microsoft Word interop scenario

- **Module 5: Garbage Collection**
  - Destructor and Finalization
  - Tracing Garbage Collection
  - Interacting with the Garbage Collector
  - Generations
  - Weak References

- **Module 6: XML Documentation**
  - XML Overview
  - XML Documentation in Comments
  - Auxiliary Tools – Sandcastle, DocumentX!

- **Module 7: Arrays and Strings**
  - Array Definition and Usage – Multi-dimensional, jagged, System.Array
  - Casting and Enumerating Arrays
  - String Class Members
  - String Immutability
  - StringBuilder
  - String Literals
  - Lab 5: Name Processing
    - Reading, sorting and writing strings and files

- **Module 8: Object Oriented Programming in C#**
  - Inheritance and Polymorphism
  - Up Casts and Down Casts
  - Inheritance and Overriding Subtleties
  - Lab 6: Shapes
    - Shape inheritance hierarchy
    - Extending the hierarchy – a compound shape (Composite design pattern)

- **Module 9: Structures and Enumerations**
  - User-Defined Value Types
  - Field Initialization
  - Nullable Types
  - Enumerations and Flags

- **Module 10: Indexers**
  - Indexers
  - Consuming Indexers from Other .NET Languages
  - Lab 7: Receptionist Scheduling

- Indexer access to classes
- Multi-parameter indexers

- **Module 11: Exception Handling**
  - Error Reporting Alternatives
  - Throwing and Catching Exceptions
  - Exception Types and Objects
  - Inner Exceptions
  - User-Defined Exceptions
  - Resource Management
  - Checked and Unchecked Arithmetic
  - Exception Design Guidelines and Performance
  - Lab 8: Incorporating Exception Handling
    - Adding exception handling to Lab 4

- **Module 12: Interfaces**
  - Interface Declaration and Implementation
  - Explicit Interface Implementation
  - System Interfaces
  - Extending Interfaces using Extension Methods
  - Lab 8: Enumeration Capabilities
    - Providing enumeration via foreach to the class from Lab 7
    - Providing find (with a comparer) capabilities to the class from Lab 4

- **Module 13: Operator Overloading**
  - Overloading Operators
  - Operator Names in the CLS
  - User-Defined Conversions – Implicit and explicit, sequence of conversions

- **Module 14: Delegates and Events**
  - Delegate Definition and Usage
  - Delegate Implementation
  - Multi-cast Delegates
  - Anonymous Methods
  - Lambda Functions
  - Events
  - Event Design Patterns
  - Lab 10: Sorting with Delegates
    - Sort criteria implementation using delegates
  - Lab 11: Event-Based Chat System
    - Client and server event-based chat

- **Module 15: Preprocessor Directives**
  - Preprocessing Directives
  - Defining and Undefining Preprocessor Directives

- **Module 16: Improved C++**
  - Control Flow Statements
  - Switch Blocks

- **Module 17: Metadata and Reflection**
  - Metadata Tables
  - Reflection Types
  - System.Activator
  - Lab 12: Self-Registration with Interfaces
    - Self-registered singleton repository using a marker interface

- **Module 18: Attributes**
  - Attribute Class
  - Attribute Examples
  - Applying Attributes
  - User-Defined Attributes and Attribute Usage
  - Querying Attributes with Reflection
  - Lab 13: Logging with Attributes
    - Primitive object serialization for logging purposes
  - Lab 14: Self-Registration with Attributes
    - Self-registration (see Lab 12) with attributes instead of a marker interface

- **Module 19: Generics**
  - Motivation for Generics
  - Generic Constraints
  - Generic Interfaces, Methods and Delegates
  - .NET Generics vs. C++ Templates
  - Generics and Reflection

- **Module 20: Generic Collections**
  - Built-in Generic Collections
  - Generic System Interfaces
  - Collection Initializers
  - Lab 15: Implementing a Generic Collection
    - Implementing IList<T> on the collection from Lab 4

- **Module 21: Deployment, Versioning and Configuration**
  - Deployment and Versioning of .NET Assemblies
  - Private and Shared Assemblies – The Global Assembly Cache (GAC)
  - Application Configuration Files
  - Versioning Policies
  - Friend Assemblies
  - Multi-Module Assemblies
  - Lab 16: Creating and Registering Assemblies
    - Creating a privately deployed assembly
    - Using probing configuration to access an assembly at a sub-directory
    - Registering a shared assembly in the GAC

- Controlling versioning (binding) policy using application configuration

- **Module 22: Unsafe Code and Interoperability**
  - .NET Interoperability Options
  - Introduction to Platform Invoke (P/Invoke)
  - Unsafe Code – C# Pointers
  - Lab 17: Calling Exported C Functions from C#
    - Calling a custom exported C function from C#
    - Calling a Win32 API (requiring a reverse P/Invoke callback)

- **Module 23: Introduction to Language-Integrated Query (LINQ)**
  - Anonymous Types and Implicit Variables
  - Expression Trees
  - Query Operators and the Query Pattern
  - Language-Integrated Query Keywords and Query Translation
  - LINQ to Objects
  - Lab 18: Using LINQ
    - Implementing extension methods
    - Implementing custom query operators
    - Implementing the query pattern
    - Writing declarative LINQ queries against object models

- **Module 24: Covariance and Contravariance**
  - Introduction to Covariance and Covariance
  - Evolution of Covariance and Contravariance—from C# 1.0 to C# 5.0
  - Covariant and Contravariant Delegates and Interfaces in C# 5.0

- **Module 25: Dynamic**
  - Static and Dynamic Languages
  - Dynamic Method Invocation
  - Circumventing Generic Constraints
  - Introduction to Dynamic Language Runtime
  - Extending Class Definitions with DynamicObject
  - Lab 19 - Dynamic
    - Sum an array of an arbitrary type

- **Module 26 - Async and Await**
  - History of Asynchronous Programming
  - Tasks
  - Tasks vs. APM
  - async/await syntax
  - Exceptions flow
  - Limitation