C# programming

Duration: 4 days
Audience: Professionnel TI

Who is this training for?

Developers

Training objectives

This course aims to understand why and when to use each feature of the C# language, beyond just the "how".

Summary

This C# course goes beyond explaining the many characteristics offered by the language. The majority of books, articles and training courses mainly deal with the "how" with many examples. That's good, but we forget the most important thing: specifying "why" and "when" we should use a characteristic. These explanations are fundamental to understanding "why" this characteristic has been integrated into the language (its raison d'être) and why it contributes to solving a particular problem, and to discern "when" the latter is useful to solve and facilitate the solution to a problem.

Course outline

- A brief history of C#
- Best practices for using Visual Studio for solutions and projects
- Standardization, platform support, C#-CLR relationships
- An overview of the .NET Framework
- Important object concepts in C#
- Classes, objects, and namespaces: using access modifiers and compilers
- Class members and reuse: fields, methods, parameters, use of public, protected, internal, and private methods
- Declaring constructors, read-only fields, property, indexer, and destructor
- Unified type system: reference, value, literals, conversions, boxing, and unboxing

- Operators and expressions: precedence and associativity
- Statements: block, select, and iteration
- Strings, arrays, variables, and parameters
- Passing arguments by value, by reference, with variable number, and with named arguments
- Implicitly typed local variables and method overload
- Collections and generics
- Class reuse: aggregation and inheritance
- Comparison of aggregation and inheritance
- Advanced types: Delegates and Events
- Using delegate inferences
- Using delegates for callbacks, events, and anonymous methods
- Exceptions and error handling: try-catch and finally blocks and throw launch
- Importance of the root class: for comparing and cloning objects
- Abstract classes and interfaces
- Polymorphism and accessors: overloading versus override methods
- Creating value types: struct and enumeration
- Lambda expressions and extension methods
- Programming threads
- Threading states and priorities
- Creating, start, reorder and synchronize threads
- Use the Monitor class
- Use of input/output classes, files, streams, attributes and serialization
- Complete project from A to Z reusing common components with console and graphical interfaces

Approach and methodology

- Lectures
- Demonstrations
-

Guided and individual exercises

The distribution of the content is approximately: equipment 35% and laboratories 65%

Prerequisites

Course "Introduction to Object-Oriented Development" or Knowledge of Object Concepts

Recommendations

Experience with C or C++, Java or Visual Basic is an asset