Programmation C#

Durée: 4 jours
Public cible: Professionnel TI

À qui s'adresse cette formation ?

Développeurs

Objectifs de la formation

Ce cours vise à comprendre pourquoi et quand utiliser chaque caractéristique du langage C#, au-delà du simple "comment".

Description sommaire

Ce cours de C# va au-delà de l'explication des nombreuses caractéristiques offertes par le langage. La majorité des livres, articles et formations traitent principalement du « comment » avec bien des exemples. C'est bien, mais on oublie le plus important : préciser « pourquoi » et « quand » l'on doit utiliser une caractéristique. Ces explications sont fondamentales pour bien comprendre « pourquoi » cette caractéristique a été intégrée dans le langage (sa raison d'être) et pourquoi elle contribue à résoudre telle problématique, et pour discerner « quand » cette dernière est utile pour régler et faciliter la solution à un problème.

Contenu du plan de cours

- Bref historique de C#
- Meilleures pratiques d'utilisation avec Visual Studio pour les solutions et les projets
- Standardisation, soutien de plateformes, relations entre le C# et le CLR
- Survol du Framework .NET
- Concepts objet importants en C#
- Classes, objets et espaces de nommage : utilisation des modificateurs d'accès et unités de compilation
- Membres de classe et réutilisation : champs, méthodes, paramètres, utilisation de méthodes publiques, protégées, internes et privées
-

Déclaration de constructeurs, de champs en lecture seulement, de propriété, d'indexeur et de destructeur

- Système de type unifié : référence, valeur, littéraux, conversions, emboîtage (Boxing) et désemboîtage (Unboxing)
- Opérateurs et expressions : priorité et associativité
- Instructions : bloc, sélection et itération
- Chaînes, Tableaux, variables et paramètres
- Passage d'arguments par valeur, par référence, avec nombre variable et avec arguments nommés
- Variables locales implicitement typées et surcharge de méthodes
- Collections et génériques
- Réutilisation de classes : agrégation et héritage
- Comparaison de l'agrégation et de l'héritage
- Types avancés : Délégués (Delegates) et événements (Events)
- Utilisation des inférences de délégué
- Utilisation des délégués pour les rappels (callback), les événements et les méthodes anonymes
- Exceptions et gestion des erreurs : try-catch et finally blocs et lancement throw
- Importance de la classe racine : pour la comparaison et le clonage d'objets
- Classes abstraites et interfaces
- Polymorphisme et accesseurs : surcharge (overload) versus redéfinition (override) de méthodes
- Création de types valeur : structure (struct) et énumération (enum)
- Expressions lambda et méthodes d'extensions
- Programmation de fils d'exécution (threads)
- Les états et les priorités d'un thread
- Création, démarrage, réordonnancement et synchronisation des threads
- Utilisation de la classe Monitor
- Utilisation de classes d'entrées ou sorties, fichiers, flux (Streams), attributs et sérialisation
- Projet complet de A à Z réutilisant des composants communs avec interfaces console et graphique

Approche et méthode pédagogique

- Exposés
- Démonstrations
- Exercices dirigés et individuels

La répartition du contenu est approximativement : matériel 35% et laboratoires 65%

Prérequis

Cours "Introduction au développement orienté-objet" ou la connaissance des concepts objet

Recommandations

Une expérience de C ou C++, Java ou Visual Basic est un atout