

Programmation Java : Les bonnes pratiques - 4 jours

formation 516

- Vous apprendrez à**
- Appliquer les bonnes pratiques Java pour accroître la productivité et créer des applications performantes, sécurisées et fiables
 - Automatiser le déploiement, les tests et la détection de bugs dans les applications logicielles
 - Résoudre les problèmes relatifs à l'architecture grâce à des design patterns éprouvés et à des fonctionnalités avancées du langage et améliorer la fiabilité des applications multitâches
 - Coder en Java de façon sécurisée et authentifier avec les plates-formes de développement actuelles
- À qui s'adresse cette formation**
- Développeurs, architectes et toute personne impliquée dans des projets Java et souhaitant étendre ses compétences en programmation Java. Des connaissances de Java du niveau de la formation 471, "Programmation Java : Les fondamentaux", sont supposées acquises.
- Travaux pratiques**
- Amélioration de la testabilité en créant le test unitaire d'une classe en même temps que celle-ci
 - Implémentation de design patterns orientés objet pour améliorer l'extensibilité et la maintenabilité
 - Optimisation des performances logicielles en réorganisant les boucles et en diminuant le nombre d'appels vers les bases de données

Programmation Java : Les bonnes pratiques - 4 jours *formation 516*

Programmation efficace en Java

- Objectifs des bonnes pratiques
- Identifier les caractéristiques clés d'un logiciel de haute qualité

Optimisation du développement de logiciels grâce à des techniques éprouvées

Simplifier la génération et le déploiement des projets

- Automatisation du processus de génération en utilisant Ant
- Contrôle et configuration de la journalisation

Mise en place du développement guidé par les tests

- Constituer et maintenir les tests JUnit
- Automatisation des tests sur l'intégralité du projet
- Validation des résultats des applications avec les tests fonctionnels
- Tests de composants encapsulés tels que les servlets

Meilleure conception pour une qualité du code améliorée

Recommandations des experts

- Équilibrer extensibilité et maintenabilité
- Limiter les problèmes de chargement de classe
- Bonnes pratiques pour la gestion des exceptions

Contrôle des types

- Élimination des erreurs d'exécution grâce aux types génériques
- Limitation des valeurs de paramètre avec la canonicalisation

Mise en place de l'encapsulation

- Fournir des macros méthodes avec le design pattern Memento et simplifier l'adaptation aux interfaces

Créer des frameworks flexibles

- Élargir l'applicabilité avec l'introspection
- Simplifier l'introspection avec les JavaBeans et les annotations

Refactorisation et design patterns

- Simplification du code source avec la refactorisation
- Conception d'interfaces pour une meilleure flexibilité logicielle
- Design patterns orientés objet clés
- Patron de méthode
- Stratégie

- Singleton
- Composite
- Factory
- Inversion de contrôle

Automatisation des contrôles qualité du code

- Normes applicables à l'intégralité du projet
- Suppression des erreurs de codage courantes
- Identification précoce des erreurs de conception

Réglages pour un maximum de performances

- Outils d'analyse des performance, évaluation des temps de réponse, réalisation des tests de charge et stress
- Identification de goulots d'étranglement
- Techniques de gestion des problèmes de performances courants de Java
- Utilisation du ramasse-miettes
- Choix des paramètres adaptés pour la JVM et le container
- Évaluation des besoins de NIO et JNI
- Réorganiser les boucles pour améliorer les temps de réponse
- Traitement des données en flux continu pour diminuer les dépassements de mémoire

Utilisation efficace de l'API Collections

- Éviter les fuites de mémoire grâce aux références faibles
- Choix des meilleures classes collection

Tirer pleinement profit des processus légers

Parallélisation pour un meilleur temps de réponse

- Écriture de code fiable et réentrant
- Éviter les pièges du multitâche: recouvrement et interblocages

Sécurisation d'une application multitâche

- Synchronisation des processus légers
- Techniques de partage de données entre les threads
- Conséquences de la synchronisation sur les performances

Mise en place de contraintes de sécurité

- Codage sécurisé en Java
- Restrictions d'accès aux ressources protégées
- Établissement de règles de sécurité

- Application de la sécurité basée sur les rôles
- Authentification des utilisateurs dans des applications web

Étendre les fonctionnalités d'une application

- Limiter l'impact des modifications avec Proxy Adapter
- Inversion de contrôle (IoC) par les Beans Factories
- Injection de comportement avec les aspects
- Doter une application de capacités de scripting