Formation Les Design Patterns

Code : DEV-ALGO-200 Durée : 3 jours (21 heures)

Public visé

Architectes, ingénieurs concepteurs et développeurs orientés objet (Java, .NET, PHP, Python, JavaScript).

{{< formation-prerequis >}}

Objectifs pédagogiques

À l'issue de cette formation, vous serez capable de :

- Expliquer la philosophie des Design Patterns
- Catégoriser les patterns et reconnaître les principaux
- Transformer les patterns en code
- Implémenter les Design Patterns dans une architecture Web
- Utiliser les outils d'IA pour intégrer des Design Patterns dans un projet

Programme

Jour 1 - Matin

Introduction aux Design Patterns

- Présentation générale et fondamentaux
- Lien avec UML : principaux diagrammes et domaines d'application
- Formalisation des patterns
- Les familles de patterns : GoF et GRASP

Patterns créationnels

- Singleton : garantir une instance unique
- Factory : déléguer la création d'objets
- Builder : construire des objets complexes étape par étape
- Prototype : cloner des objets existants

Jour 1 - Après-midi

Patterns créationnels - Suite

- Abstract Factory : familles d'objets liés
- Mise en pratique des patterns créationnels

Travaux pratiques : Écriture et intégration des patterns créationnels dans un projet Java

Jour 2 - Matin

Anti-patterns

- Identifier et éviter les anti-patterns courants
- Refactoring vers les bonnes pratiques

Patterns structurels

- Adapter : rendre compatibles des interfaces incompatibles
- Bridge : séparer abstraction et implémentation
- Composite : structures arborescentes uniformes
- Decorator : ajouter dynamiquement des responsabilités
- Facade : simplifier l'accès à un sous-système
- Flyweight : partager des objets pour économiser la mémoire
- Proxy : contrôler l'accès à un objet

Jour 2 - Après-midi

Patterns comportementaux

- Chain of Responsibility : chaîne de traitement
- Command : encapsuler des requêtes
- Interpreter : évaluer des expressions
- Iterator : parcourir des collections
- Mediator : centraliser les communications
- Memento : capturer et restaurer un état
- Observer : notification de changements
- State : modifier le comportement selon l'état
- Strategy : interchanger des algorithmes
- Visitor : ajouter des opérations sans modifier les classes

Travaux pratiques : Implémentation de patterns structurels et comportementaux

Jour 3 - Matin

Patterns d'architecture

- MVC, MVP et MVVM : séparation des préoccupations
- DAO : abstraction de l'accès aux données
- DTO : transfert de données entre couches
- Injection de dépendances

Jour 3 - Après-midi

Méthodologie et mise en pratique

- Méthodologie de sélection du bon pattern
- Refactoring de code existant vers les Design Patterns
- Utilisation de l'IA générative pour créer et intégrer des patterns

Travaux pratiques : Refactoring complet d'un projet en appliquant les patterns étudiés

Modalités d'évaluation des acquis

En cours de formation, par des études de cas ou des travaux pratiques. En fin de formation, par un questionnaire d'auto-évaluation.

{{< formation-require >}}

{{< formation-seealso >}}