

# DDD : Domain-Driven Design

## Acquérir les pratiques d'une conception logicielle orientée métier

### DESCRIPTION

Le développement logiciel est employé généralement pour automatiser des processus existants ou pour fournir des solutions à des problèmes métier. Le Domain-Driven Design repose sur une idée simple : pour créer un bon logiciel, il est indispensable qu'il reflète le domaine métier pour lequel il est conçu, qu'il en incorpore les concepts, les process, les éléments et qu'il saisisse avec précision, leurs relations. A titre d'exemple, vous ne pouvez pas créer un système logiciel bancaire à moins d'avoir une bonne compréhension du secteur bancaire et de ses métiers.

Décrit par Eric Evans dans son ouvrage « Domain-Driven Design : Tackling complexity in the Heart of software », le Domain-Driven Design fournit un cadre solide et un ensemble de techniques décrivant comment modéliser le domaine du logiciel et définir une vision et un langage partagés par toutes les personnes impliquées dans la construction d'une application, afin de mieux en appréhender la complexité.

Tout au long de cette formation, vous serez à même de comprendre pourquoi le Domain-Driven Design permet ainsi de maintenir un alignement constant, pas toujours évident, entre les experts métier, les développeurs et le code afin de s'assurer que le logiciel réponde bien à ses objectifs.

### OBJECTIFS PÉDAGOGIQUES

Maîtriser les concepts et principes clés de l'approche Domain-Driven Design  
Mettre en œuvre les principes de conception du DDD et savoir appliquer les patterns

Utiliser un langage commun à tous les acteurs liés par le développement du logiciel

Avoir une expérience concrète d'implémentation de l'approche DDD

### PUBLIC CIBLE

Développeur  
Architecte  
Chef de projet  
Tech lead  
Scrum master

### PRÉ-REQUIS

- Pratique de la programmation orientée objet (JAVA, C#).
- Au cours de la formation, vous pourrez vous servir de votre ordinateur et du langage de programmation que vous utilisez habituellement.

### MÉTHODE PÉDAGOGIQUE

Formation pratique, visant à l'acquisition d'un savoir-faire, basée sur un cas pratique ainsi que des échanges et retours d'expérience pratique du formateur.

### Stage pratique en présentiel QUALITÉ DU LOGICIEL

Code :  
**DDD01**

Durée :  
**2 jours (14 heures)**

Exposés :  
**30%**

Cas pratiques :  
**70%**

### Sessions à venir :

9 - 10 déc. 2019  
Paris / 1 630 eur

Tarif & dates intra :  
**Sur demande**

Une session d'Event Storming sera proposée aux participants pour s'approprier les problématiques métier, et suivre une implémentation (en binômes ou tous ensemble).

## **PROFILS DES INTERVENANTS**

Toutes nos formations sont animées par des consultants-formateurs expérimentés et reconnus par leurs pairs.

## **MODALITÉS D'ÉVALUATION ET FORMALISATION À L'ISSUE DE LA FORMATION**

L'évaluation des acquis se fait tout au long de la session au travers des ateliers et des mises en pratique. Une évaluation à chaud sur la satisfaction des stagiaires est réalisée systématiquement en fin de session et une attestation de formation est délivrée aux participants mentionnant les objectifs de la formation, la nature, le programme et la durée de l'action de formation ainsi que la formalisation des acquis.

## **POUR ALLER PLUS LOIN :**

- Ouvrage « Domain-Driven Design: Tackling Complexity in the Heart of Software » - d'Eric Evans
- Article « Domain Driven Design : des armes pour affronter la complexité »
- Formation « Clean Code : concevoir et écrire un code propre ou améliorer un code existant » (TDD03)
- Formation « Les architectures réactives » (SREAC)

## Programme pédagogique détaillé par journée

### Jour 1

#### Introduction au DDD

##### CONCEPTS CLÉS DE LA DÉMARCHE

- Explorer un domaine métier via l'Event Storming
- Modéliser avec Entities, Value objects et Repositories
- Modéliser des agrégats (Aggregates)
- Factories
- Domain events
- Domain services
- Application services

##### CADRE D'UTILISATION

- Etude de cas
  - Session d'Event Storming pour comprendre et visualiser efficacement le cas métier
  - Réflexion/discussions sur les bounded contexts, aggregates et l'ubiquitous language

### Jour 2

##### DÉFINITION D'UN MODEL

- Échanges sur des exemples rencontrés
- Pièges à éviter

##### IMPLÉMENTATION TECHNIQUE EN BINÔMES OU TOUS ENSEMBLE

- Itération 1 : implémentation des premiers aggregates, entities et value objects
- Itération 2 : quand un value object doit devenir une entity
- Itération 3 : émettre un domain event
- Itération 4 : réagir à un domain event
- Itération 5 : refactoring pour avoir un meilleur regroupement des classes (bounded context)
- Itération 6 : interagir avec des objets du domaine via un application de services
- Itération 7 : stocker les domain events et des aggregates via des repositories

##### SYNTHÈSE ET RAPPEL DES POINTS CLÉS