Architecture hexagonale pour les nuls (et les développeurs Spring)

Speakers : Youen Chéné (CTO Saagie)
Format : Conférence
Date : 19 avril 2018

Youen est un ancien développeur Spring, désormais Product Manager chez Saagie.
Rencontrant des difficulté à faire évoluer une application, Youen se demande si le Domain Driven Development (DDD) ne pourrait pas l'aider.
Il a assisté à beaucoup de conférences sur les DDD. Mais les snippets de code étaient toujours sans framework. Bref, des cas d'école à mille lieux des applications Java bourrées de framework.
Ce talk a pour objectif de créer un pont entre le monde DDD et les applis en couche basées sur Spring.

Youen présente une application dans laquelle une règle métier est répartie dans 3 services et 2 DAO. Cet éparpillement est à l'architecture en couche. Nécessite un refactoring pour la centraliser dans une seule méthode.

Du point de vue d'un CTO, tous ces mouvements se ressemblent Clean Architecture ⇔ Onion Architecture ⇔ Hexagonal Architecture

Modélisation du Domaine

Les 3 gèles à respecter lorsqu'on modélise le Domaine :

1. Règle 1 : dans un framework, **pas de dépendance vers un framework** ou un librairie (à l'exception de Guava)
2. Règle 2 : **le domaine n'appelle jamais directement l'extérieur**
3. Règle 3 : **pas de code d'infrastructure dans le domaine**

Pour accéder aux règles métiers du domaine, il faut créer des flux.
Pour cela, on va créer des **ports** primaires et secondaires :

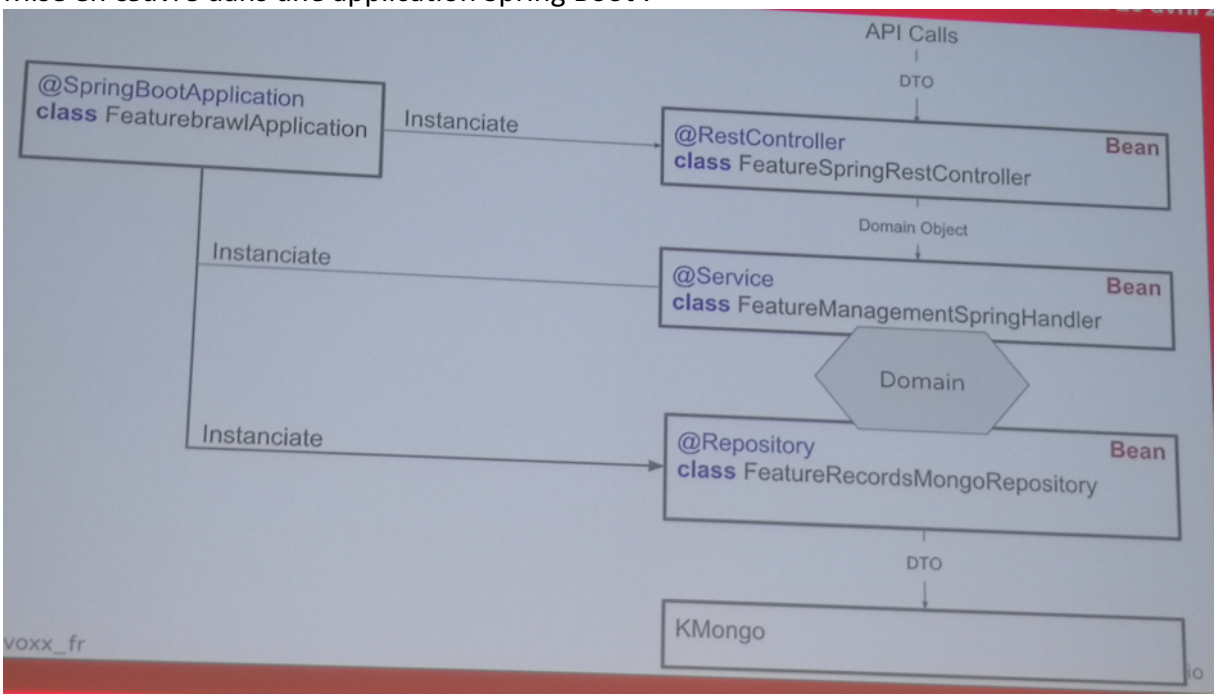
- **Port primaire** : ports d'entrée du domaine utilisés par la couche applicative
- **Port secondaire** : interface permettant au domaine d'appeler l'extérieur (repose sur de l'injection de dépendance)

Dans la littérature, on trouve également des Driving Port, Input Port et Output Port.

Les **Adapters** vont se brancher sur les ports. Toute la glue technique est placée dans les Adapters. Exemple : la gestion des exceptions techniques.
Le DDD nécessite des mappers entre les objets du domaine et les DTO.

Implémentation

Mise en œuvre dans une application Spring Boot :



Hack pour encapsuler le Domaine dans le @Service de Spring Boot : injecter le port secondaire dans le Domaine.

Mettre en place le DDD nécessite de former les développeurs avec des DOJO par exemple.

Conclusion