Découpage en Couches des Applications

M1 MIAGE Université Paris 1 Panthéon-Sorbonne

2020-2021

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Rôle de l'architecte

- L'architecte s'assure que tous les développeurs aient la même vision du découpage du système.
- Il peut être également amené à proposer un tel découpage, dans le cadre de son leadeurship technique.
- Il peut concevoir des diagrammes documentant le découpage, mais c'est un moyen et non une fin. L'objectif central est d'obtenir une vision commune par l'équipe.

Dans cette section, nous allons étudier comment concevoir une architecture multi-tiers.

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Approche de la conception par couche

- Le découpage par couche permet un découpage logique des préoccupations du système
- Chacune des couches fournit du support et les fonctionnalités de base utilisées par les autres couches
- L'architecture par couche n'est pas un concept nouveau, il est utilisé depuis des décennies dans Systèmes d'exploitation et les architectures réseaux..

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Approche de la conception par couche

Les applications métiers utilisent extensivement le découpage par couche

- par exemple, l'accès au donné bénéficie du fait d'être partagée entre les différents composants métier
- il permet également de changer l'implémentation d'une couche sans impacter les autres couches (p. ex. changement de base de données)
- il est également possible de réutiliser les couches pour ajouter de nouvelles fonctionnalités (p. ex. passage d'un site web à une API web)

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

La séparation des préoccupations (separation of concerns)

- Le principe de *separation of concerns* stipule que chaque composant ne devrait avoir qu'une seule responsabilité au sein du système
- Par ailleurs, la responsabilité de chaque composant ne devrait pas être partagée par un autre composant.
- Cette approche favorise la réutilisation du code

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Couches d'une application multi-tier

Layer (en)	Couche (fr)	Role
Data Access	Accès aux données	gère la lecture, l'écriture, la mise à jour et la suppression des données stockées. Il peut se rapporter aux bases de données ou autres sources de données (XML, CSV. . .)
Business logic	Logique Métier	gère les règles de gestion et la logique métier
Entity Object	Entité	Objets légers directement rattachés à une base de données
Value Object	Value Object	Objets légers et immutables utilisés pour transmettre des données synthétisées entre les couches
Deployment	Déploiement	Encapsule et rend disponible des règles métier
Présentation	Présentation	gère le contenu affiché à l'utilisateur
Architectural component	Composant d'architecture	Composants réutilisables à l'échelle de l'entreprise

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Couches d'une application multi-tier

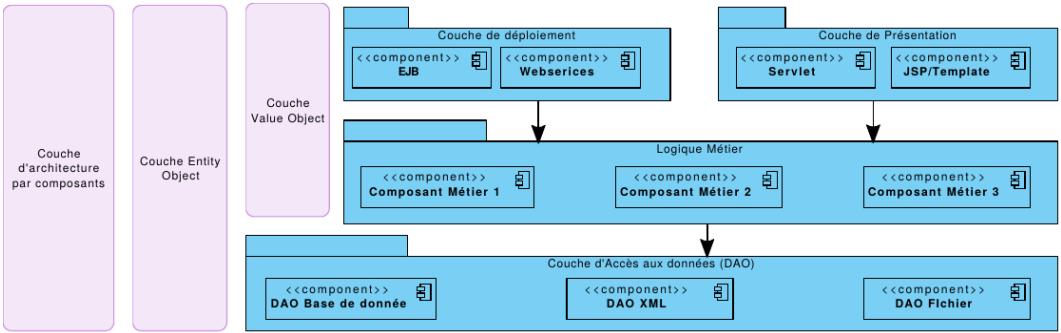

FIGURE 1 – Composition des couches d'une application métier

Packaging des couches: cas des packages Java

Il est commun de séparer les classes appartenant aux différentes couches dans des packages java différents:

package	couche
fr.miage.app.dao	Data access object (DAO)
fr.miage.app	Business logic layer
fr.miage.app.model	Entity objects
fr.miage.app.vo	Value Objects
fr.miage.app.ui	Presentation layer
fr.miage.app.services	Deployement layer/webservices

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Data Access Object Layer (DAO)

La couche DAO gère l'accès au stockage persistant: - Bases de données relationnelles - Bases de données NoSQL - Fichiers, documents XML...

- L'intérêt principal de cette couche est la possibilité de changer de méthode de persistance (p. ex. passer de MySQL à Oracle ou à MongoDB)
- La couche peut être mutualisée par plusieurs couches métier et différentes applications.

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Data Access Object Layer (DAO)

La couche DAO gère l'accès au stockage persistant: - Bases de données relationnelles - Bases de données NoSQL - Fichiers, documents XML...

- L'intérêt principal de cette couche est la possibilité de changer de méthode de persistance (p. ex. passer de MySQL à Oracle ou à MongoDB)
- La couche peut être mutualisée par plusieurs couches métier et différentes applications.

Côté implémentation

L'implémentation en Java est facilitée par la classe *EntityManager*. Retrouvez plus d'info dans le chapitre JPA

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Entity Object Layer

- Les entités correspondent directement à une table d'une base de données (le plus soutent relationnelle)
- Un Table *Product* correspondra à un objet *fr.miage.app.model.Product*
- Les classes DAO peuvent être comprises comme des *factory* d'entités, permettant également leur persistance.
- Ces classes possèdent uniquement des setter/getter
- Elles peuvent être générées automatiquement

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Entity Object Layer

- Les entités correspondent directement à une table d'une base de données (le plus soutent relationnelle)
- Un Table *Product* correspondra à un objet *fr.miage.app.model.Product*
- Les classes DAO peuvent être comprises comme des *factory* d'entités, permettant également leur persistance.
- Ces classes possèdent uniquement des setter/getter
- Elles peuvent être générées automatiquement

Côté implémentation

La correspondance des entités avec les tables est réalisée avec des annotations JPA, retrouver plus d'info sur le chapitre JPA

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Value Object Layer

- En plus des entités, les applications ont besoin d'une série d'objets transitoires (*transient*)
- Ces objets sont là pour faciliter l'échange de données entre les couches, mais ne sont jamais modifiés ou sauvegardés en base.
  - e.g. une application d'analyse de données peut produire des statistiques agrégées sur le mois, calculées à partir des données brutes.
- Ils ne contiennent en général que des getters et leurs attributs sont immutables (*final*)

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Business Logic Layer

- Les objets de la couche métier contiennent des données, des règles métier, des contraintes et des activités.
- Ils se servent d'objets de la couche DAO, d'entités, de Value Objects.
- Ils utilisent et coordonnent fréquemment plusieurs DAO
- Ils devraient pouvoir être utilisés dans plusieurs contextes différents:
  - dans des traitements par lot (*batch*)
  - par des web services
  - par des clients lourds
- Ils ne font donc pas d'hypothèse sur le contexte d'utilisation

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Business Logic Layer Patterns: *Transaction Script* 1

Organise la logique métier en procédures où chaque procédure est responsable d'un résultat utilisé par la couche présentation/déploiement

- Pourquoi?
  - La plupart des applications métiers peuvent être vues comme des séries de transactions.
  - Les transactions peuvent être en lecture et/ou en écriture
  - *Transaction Script* organise la logique de chacune de transaction dans une procédure.
- Comment ça marche?
  - La logique métier est découpée dans une procédure
  - Par exemple, on veut créer un compte bancaire: on doit récupérer l'utilisateur dans la DB ou le créer s'il n'existe pas, vérifier qu'il n'est pas interdit bancaire auprès de l'autorité administrative puis créer le compte et le rattacher à l'utilisateur
  - On peut avoir 1 ou plusieurs procédures par classe, en fonction de la cohésion de celles-ci

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Business Logic Layer Patterns: *Transaction Script* 2 (UML Diagram)

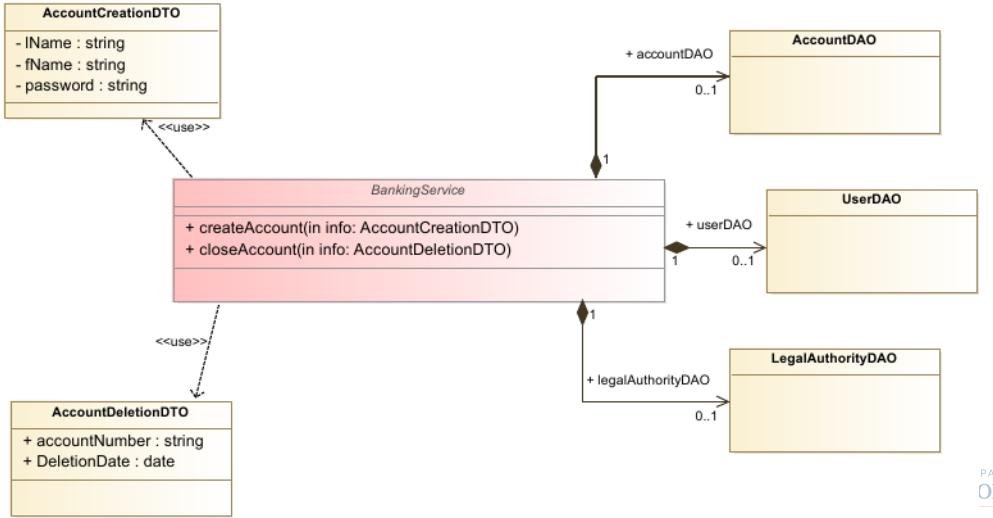

FIGURE 2 – Transaction Script Pattern

Business Logic Layer Patterns: *Transaction Script* 3

- Quand l'utiliser?
  - Hypersimple d'utilisation
  - difficile de réutiliser le code métier entre plusieurs transactions

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE