Architecture Logicielle Distribuée et Adaptative

Dupin Victor
Jolly Solène
Lamoureux Adrien

26 janvier 2015

Table des matières

1. Architecture du projet

1.1 Gestionnaire de configuration

Nous avons utilisé Maven pour la gestion du projet. Ce dernier prend racine sur un méta-projet servant à capitaliser les dépendances redondantes à chaque sous-projet. Nous avons développé deux sous projets Maven. L'un correspondant au front-end de l'application (client) et l'autre au back-end (service).

1.2 Architecture service

Le back-end a été développé en J2EE à l'aide de Jersey et JPA. L'architecture du service est présentée dans la figure 1.1, page 4.

Le dossier /entities contient toutes les classes annotées JPA qui seront traduites dans le domaine relationnel. Le fichier MuseumRoot.java contient tous les points d'accès pour les méthodes REST dans le but d'interroger la base de données. L'interrogation passe par les fichiers contenus dans le dossier Data/ ayant accès à un EntityManager instanciable via des filtres (/filters).

Une phase de test a été intégrée au cycle de Maven avant la phase de déploiement. La phase de tests :

1. instancie une nouvelle base de données ;
2. établit une communication via un EntityManager ;
3. efface et remplit la base de données à l'aide d'un fichier de données XML ;
4. exécute une méthode de test ;
5. répète les étapes 3 et 4 jusqu'à ce que tous les tests soient terminés ;
6. envoi les résultats des tests dans des fichiers de sortie.

1.3 Architecture client

Le client a été développé en web 2.0 avec un design MVC à l'aide du framework AngularJS. L'architecture du service est présentée dans la figure 1.2, page 5.

La communication avec le service du musée passe par l'intermédiaire d'un serveur express. Ainsi, seul le serveur express dispose des chemins pour accéder et modifier les requêtes reçues / envoyées au serveur du musée.

FIGURE 1.1 – Architecture service

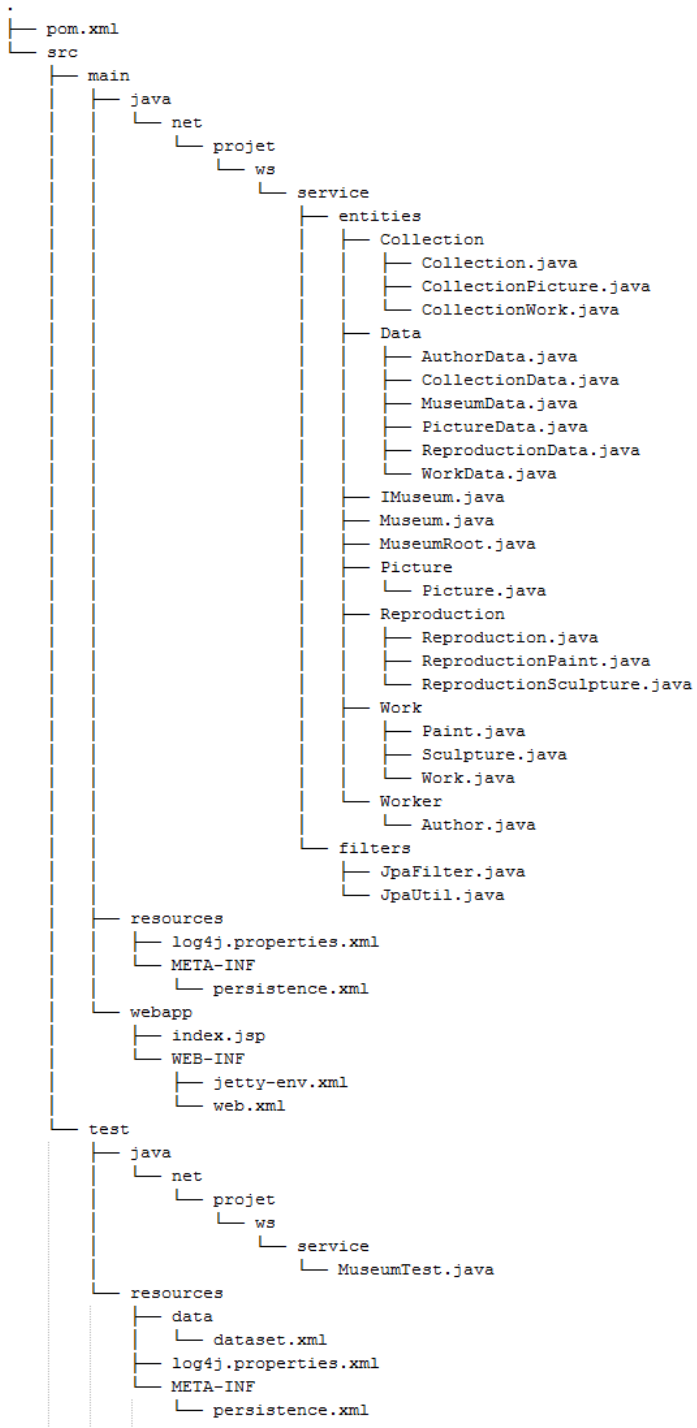
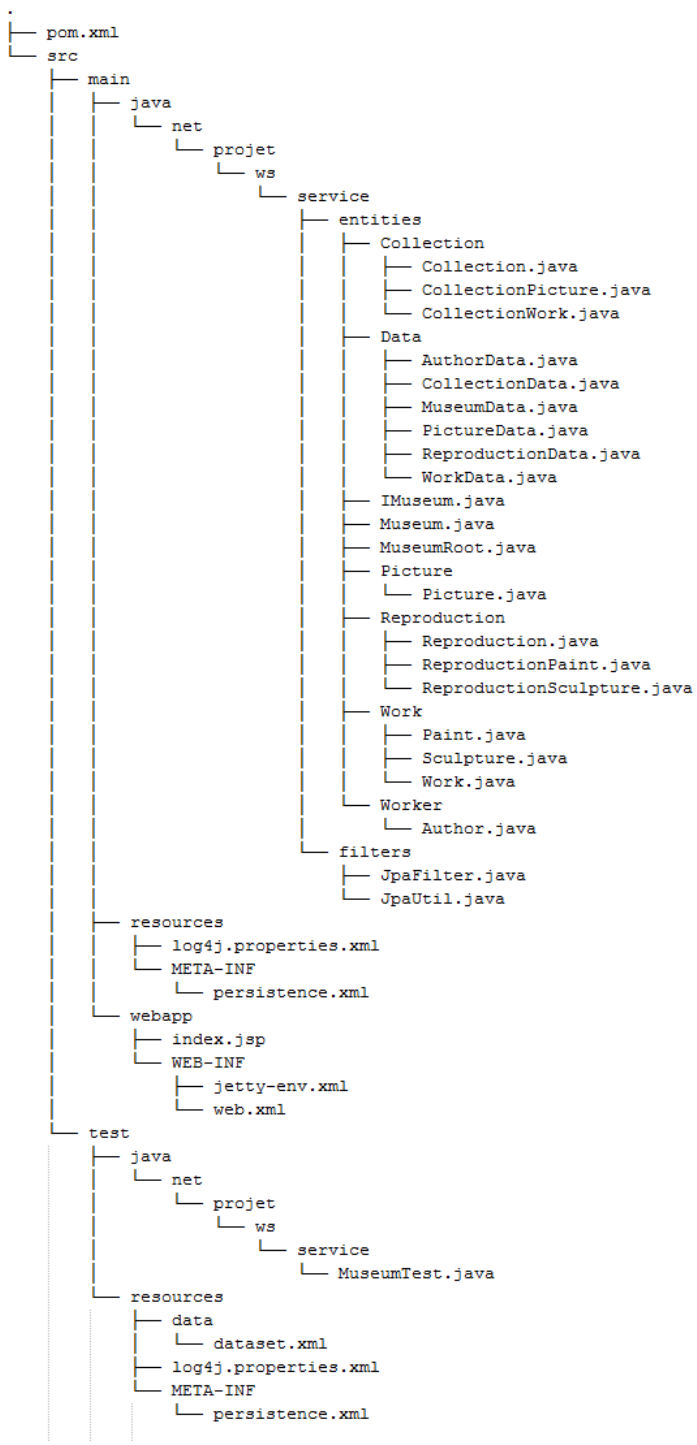


FIGURE 1.2 – Architecture client



2. Fonctionalités

Le gestionnaire de musée permet d'effectuer des requêtes CRUD sur des :
- musées ;
- œuvres ;
- artistes ;
- photos ;
- collections ;
- reproductions.

Chaque entité présente dans le musée (excepté l'artiste) :
- peut être commentée par le visiteur ;
- est renseignée par un tag ;
- dispose d'un titre et d'une description.

Les œuvres, artistes et photos sont liés de manière bidirectionnelle dans l'objectif de faciliter les associations entre ces entités. Les reproductions ne sont accessibles qu'à partir des œuvres et il n'existe pas de collections de reproductions. Les collections ne sont accessibles qu'à partir d'un musée.

2.1 Musées

Le gestionnaire de musées permet de gérer plusieurs musées de la façon qui suit :
- les entités sont fortement liées aux musées. Ainsi, la suppression d'un musée entraînera la suppression de l'ensemble des entités liées de manière bidirectionnelle au musée ;
- chaque musée est totalement indépendant et donc, il n'est pas possible de capitaliser de l'information entre les différents musées. Chaque musée dispose de caractéristiques (thème, adresse, ...).

2.2 Œuvres

Il existe deux types d'œuvres : peintures et sculptures. Le premier peut disposer d'une particularité technique et d'une particularité de support. Le

second dispose de plusieurs particularités de support. Les œuvres disposent aussi de divers champs (résumé, dimensions, ...). Une œuvre est liée à 0..n photo(s) et à un auteur.

2.3 Photos

Une photo renseigne l'œuvre à laquelle elle est liée. La photo dispose de caractéristiques propres mais n'offre pas la possibilité de choisir l'angle de prise de vue.

2.4 Artistes

Les artistes sont renseignés par un nom et une adresse. Ils sont liés à 0..n œuvre(s).

2.5 Collections

Il en existe deux types :
- les collections d'œuvres sont liées unidirectionnellement à des œuvres ;
- les collections de photos sont liées unidirectionnellement à des photos.

2.6 Reproductions

Les reproductions ne concernent que les œuvres. Il n'est possible d'accéder à une reproduction qu'à partir d'une œuvre. Les reproductions disposent de caractéristiques comme le nombre restant, le prix et les particularités de cette reproduction, ...

2.7 Fonctionnalités absentes en front-end

Les fonctionnalités décrient ci-dessus n'ont pas toutes été intégrées sur le front-end. Parmi ces dernières, il y a :
- l'accès aux reproductions ;
- l'ajout de nouveaux commentaires ;
- l'ajout de multiples tags ;
- l'upload et le download de photos.

3. Discussions

3.1 Domaine objet et relationnel

Nous avons consulté les annotations proposées par JPA 2.0 dans l'objectif de profiter au maximum des avantages du monde objet. Néanmoins, nous nous sommes heurtés à un certain nombre de problèmes de conception, notamment en ce qui concerne les relations d'héritage et de polymorphisme (conflits entre tables, etc). De ce fait, seules les classes de plus bas niveau ont été définies comme entités. Par exemple, les classes Work et IMuseum sont toutes les deux abstraites et sont renseignées par une annotation « @MappedSuperclass » signifiant que les champs sont factorisés pour les sous-classes entités. Ainsi, la modularité du modèle objet développé se voit réduite par la nécessité de travailler majoritairement avec des types concrets qui s'entre-référencent. Ceci n'est pas un problème car, dans le contexte métier actuel, les ajouts de catégories d'objets se feront rares.

3.2 Interactions client / service

La stratégie de développement a été de fournir des services (GET, POST, PUT et DELETE) pour chaque entité persistée. Des requêtes permettant l'accès direct à une entité complète ont été développées dans l'objectif de diminuer la taille des données à transiter.

En tenant compte du cadre d'utilisation de notre logiciel, nous pouvons prévoir que les modifications seront importantes mais à faible fréquence (modifications importantes avant exposition, ...). Le coût de développement que représentait la décomposition des services était trop élevé pour une manipulation occasionnelle. Les requêtes sont donc peu ciblées : le front-end doit nécessairement envoyer l'ensemble d'une entité même si un seul champ est renseigné.

Il est également possible d'effectuer des requêtes en envoyant à la fois l'ID d'un musée et l'ID d'une entité de ce même musée. Bien que le front-end

développé ici n'utilise pas ces requêtes (dû à la structure du site en front-end), nous avons tout de même conservé ces dernières pour offrir la possibilité de diminuer le nombre de requêtes à un autre client.

3.3 Extensions des fonctionnalités

Certaines extensions sont envisagées, grâce à l'apport du serveur express côté client :

- communiquer avec plusieurs services (par exemple, un pour les auteurs, un pour les œuvres, etc) ;
- déployer les clients à part d'express dans l'objectif de connecter plusieurs clients à express.

Les fonctionnalités présentes permettent également des ouvertures :

- partage du système d'information entre plusieurs musées ;
- recherche d'oeuvres / de photos / de musées en fonction de mots clés.