Atelier
Persistance avec JPA

Présentation de la solution à mettre en place

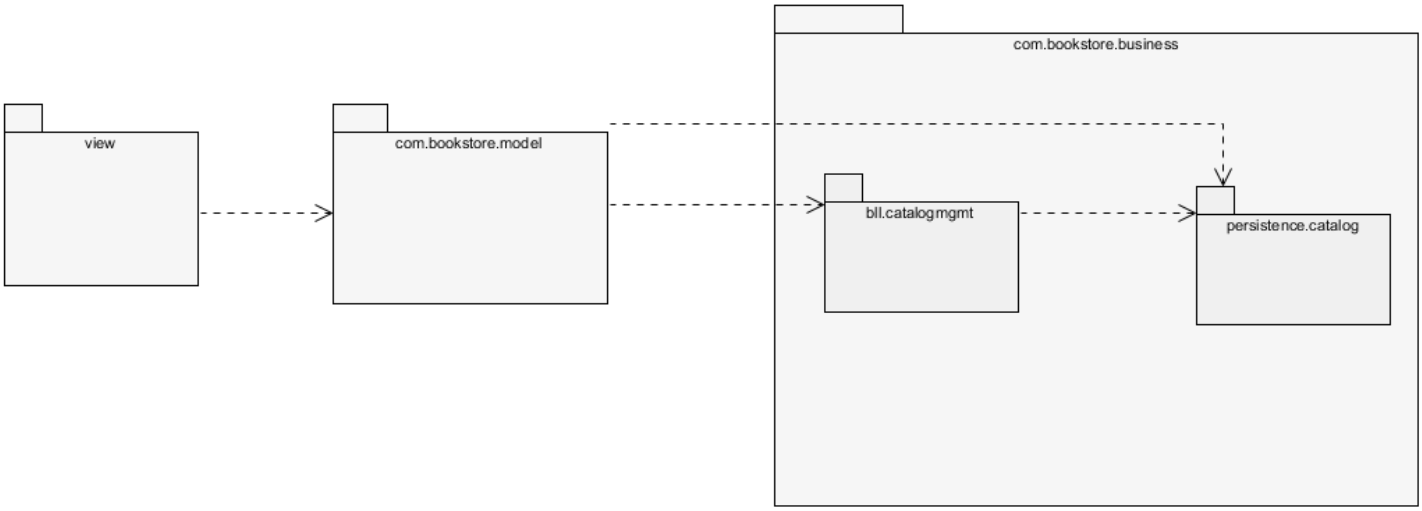
Nous nous situons dans le cadre d'une « application » devant gérer le catalogue d'une librairie.
Dans le dossier ***images***, vous trouverez les différents diagrammes présentés dans cet énoncé.
Pour débuter voici un diagramme de cas d'utilisation présentant les différentes fonctionnalités présentent dans bookStore (bs_use.jpg) :

Les fonctionnalités n'ont pour objectif que de vous faire travailler sur JPA.
Elles sont présentées dans l'ordre où elles sont listées dans la page d'accueil de l'application.
La notion d'inclusion signifie que la modification d'un livre n'est accessible qu'au travers de la fonctionnalité Associer un livre à un éditeur.

A noter que *lister les catégories* signifie lister des catégories racines (sans parent) ou des catégories enfants d'une catégorie (plus d'informations sur le domaine ultérieurement).

Atelier
Persistance avec JPA

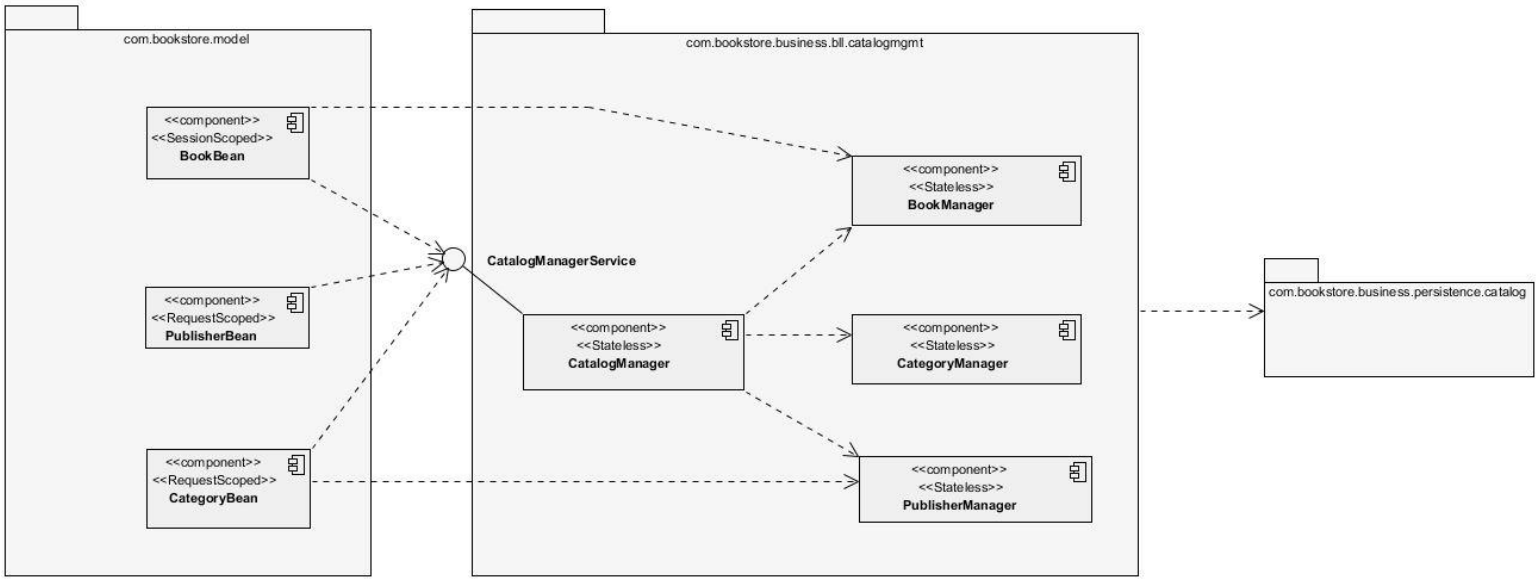
Voici un diagramme de package (bs_pack.jpg) qui présente la structure des différents paquetages et leurs interactions.



Le paquetage view n'est pas un paquetage au sens java. C'est le regroupement des Facelets JSF fournissant l'IHM web de gestion du catalogue. Les vues sont liées aux beans CDI du paquetage *com.bookstore.model* grâce au langage d'expression unifié (#{expression}). Ce paquetage contient les beans managés CDI qui vont lier les vues à la logique métier. Ceux sont ces beans qui permettent au client (aux vues) d'interagir avec l'application. Ce paquetage est ainsi nommé car il correspond au point d'entrée de la partie *Modèle* au sens MVC.
com.bookstore.business modélise la couche métier déployée sur le serveur.
com.bookstore.business.bll.catalogmgmt représente la subdivision logique métier de la gestion du catalogue. Les stateless session beans (SLSB) implémentant l'essentiel de logique métier (bll = business logic layer). C'est ici que les opérations de persistance (d'accès aux données) seront gérées.
com.bookstore.business.persistence.catalog contient les entités modélisant le domaine du catalogue. Elles sont manipulées par les session beans. A noter que les entités de *com.bookstore.persistence.catalog* sont accessibles dans les vues. Ceux sont des éléments du modèle au sens MVC.

Ci-dessous le diagramme des composants représentant l'interaction entre les beans CDI et les session beans stateless gérant le catalogue (bs_comp.jpg) :

Atelier
Persistance avec JPA

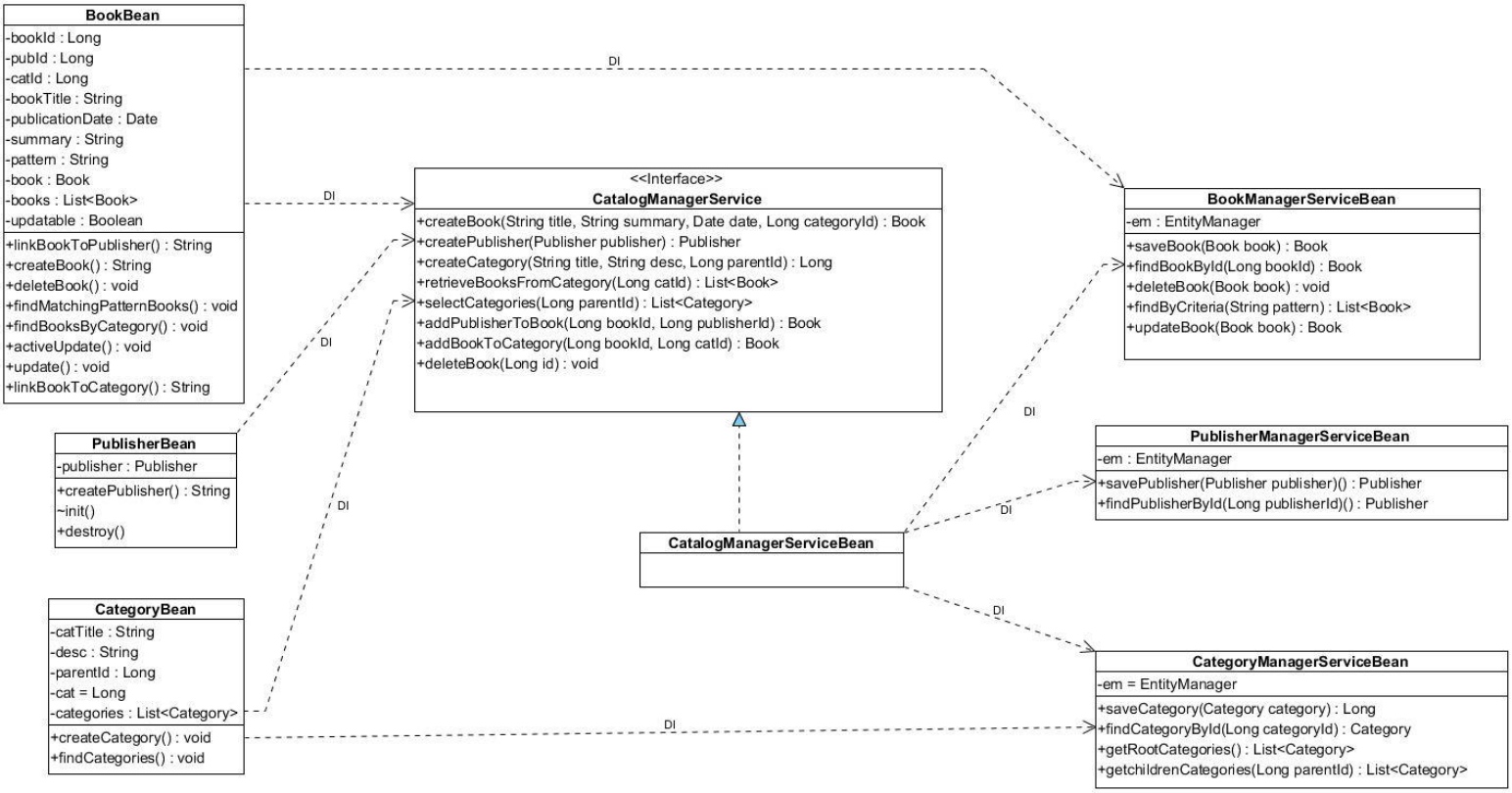


Les noms de session beans sont spécifiés dans le code via *@Stateless.name*.
CatalogManager décrit une façade à forte granularité qui centralise la gestion du catalogue de la librairie. Il s'exécute dans un environnement transactionnel. Soit il joint une transaction active cliente soit il en démarre une nouvelle (attribut REQUIRED). Dans ce workshop les beans CDI ne démarrent pas de transaction, une transaction sera donc démarrée par le conteneur EJB pour chaque invocation de méthode de CatalogManager. CatalogManager expose une vue locale de type interface annotée avec @Local (CatalogManagerService).
CatalogManager s'inspire du pattern Service Facade. Pour réaliser la logique métier, il s'appuie sur les services BookManager (gestion des livres), CategoryManager (gestion des catégories) et PublisherManager (gestion des éditeurs). Ces services sont essentiellement chargés de mettre en œuvre les opérations de persistance CRUD (Create, Retrieve, Update, Delete) sur les entités adéquates. A noter que vous implémenterez uniquement les comportements nécessaires pour cet atelier. Ces 3 services exposent une vue locale sans interface. Ils n'implémentent aucune interface métier. Ces composants ont bien sûr une granularité moins forte que la façade de gestion du catalogue. Ces 3 services interagissent avec la couche de persistance.
Ils doivent être généralement invoqués au sein d'une transaction créée par un client (ici CatalogManager). L'attribut de transaction spécifié sur les classes de ces composants est donc MANDATORY.
A noter que l'attribut transactionnel SUPPORTS est spécifié sur *BookManagerServiceBean.findByCriteria(String pattern)* et *CategoryServiceBean. findCategoryById(Long catId)*. Le comportement transactionnel de ces méthodes est donc redéfini. Cela permet d'invoquer ces méthodes depuis les beans CDI non transactionnels sans passer par la façade. SUPPORTS permet à une méthode d'être enrôlée dans la transaction l'appelant. Si l'appelant n'est pas associé à une transaction, il n'y aura pas de transaction démarrée pour la méthode annotée SUPPORTS. Ces 2 méthodes implémentent uniquement des opérations de lecture, c'est pour ça qu'elles n'ont pas besoin de s'exécuter obligatoirement dans une transaction.
Dans un même ordre d'idée l'attribut REQUIRED est spécifié sur la méthode *BookManagerServiceBean.updateBook(Book book)* pour que cette méthode s'exécute au sein d'une transaction. En effet toute opération de modification de la base telle que la mise à jour

nécessite d'être associée à une transaction. Ainsi la méthode pourra être invoquée directement depuis un bean CDI.

Remarque :
Pour rappel depuis Java EE 7, les beans CDI peuvent être annotés avec @Transactional pour permettre l'invocation des méthodes au sein de transactions.

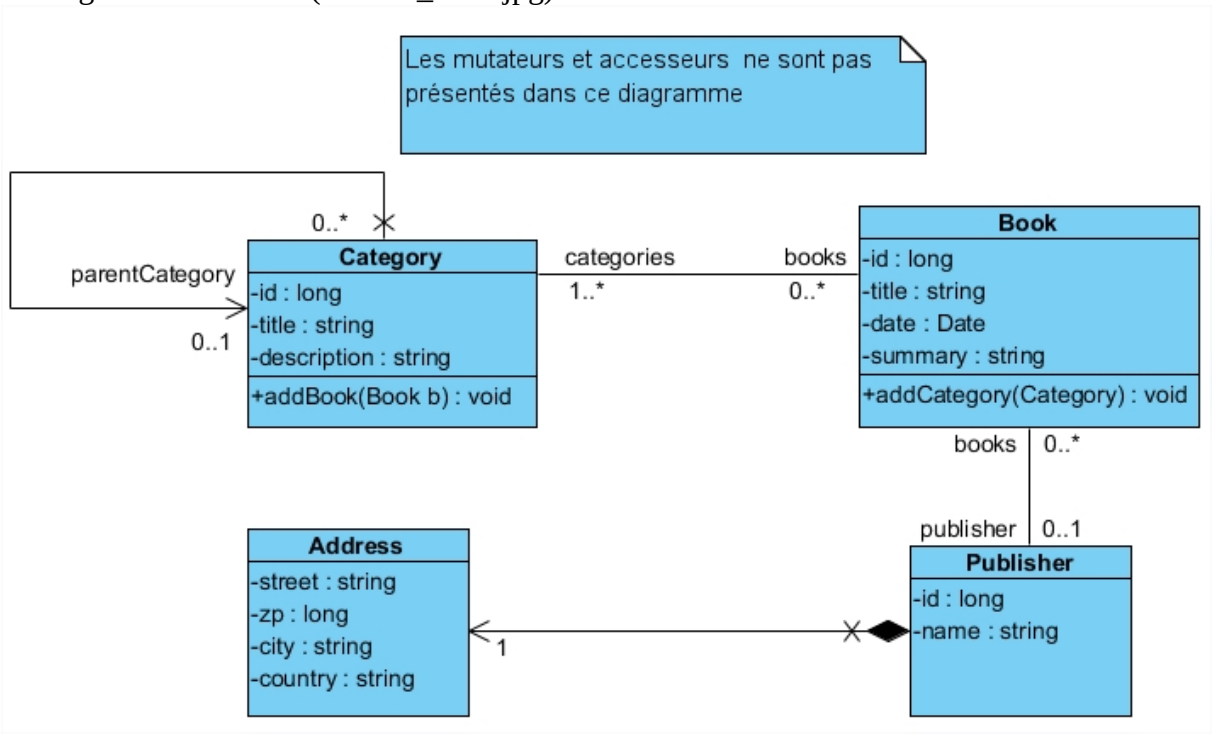
Passons au diagramme présentant plus précisément les composants managés chargés du traitement et donc entre autre de la manipulation des entités JPA (bs_class.jpg) :



Seules les méthodes utilisées dans le workshop sont définies.
DI signifie Dependency Injection.
Les getters et setters des propriétés des beans CDI (BookBean, publisherBean, CategoryBean) ne sont pas représentés. Les méthodes d'action de ces beans qui retournent String ou void initient les différents processus implémentés au sein de la logique métier.
Les chaînes (String) retournées par les méthodes d'action permettent de naviguer dans les différentes vues JSF. A la différence de goodcesi, la navigation est configurée au sein de WEB-INF/faces-config.xml. Cela permet de séparer les règles de navigation de l'implémentation. Ici les chaines retournées ne sont pas des noms de vues (comme c'était le cas dans goodcesi) mais des sorties utilisées par le gestionnaire de navigation qui examinera les règles dans faces-config pour déterminer vers quelle page naviguée.
Les méthodes retournant « void » n'entrainent pas de navigation vers une nouvelle vue.
Les méthodes qui seront à implémenter / compléter pour mettre en œuvre la gestion de la persistance JPA sont les méthodes définies dans les session beans Stateless BookManagerServiceBean, PublisherManagerServiceBean, CategoryManagerServiceBean.

Pour plus de précisions sur le rôle des méthodes, reportez-vous à la javadoc (dossier **javadoc**). La page d'accueil est index.html.

Voyons maintenant ce qui est peut-être le plus important, c'est-à-dire le modèle du domaine catalogue de bookStore (domain_class.jpg) :

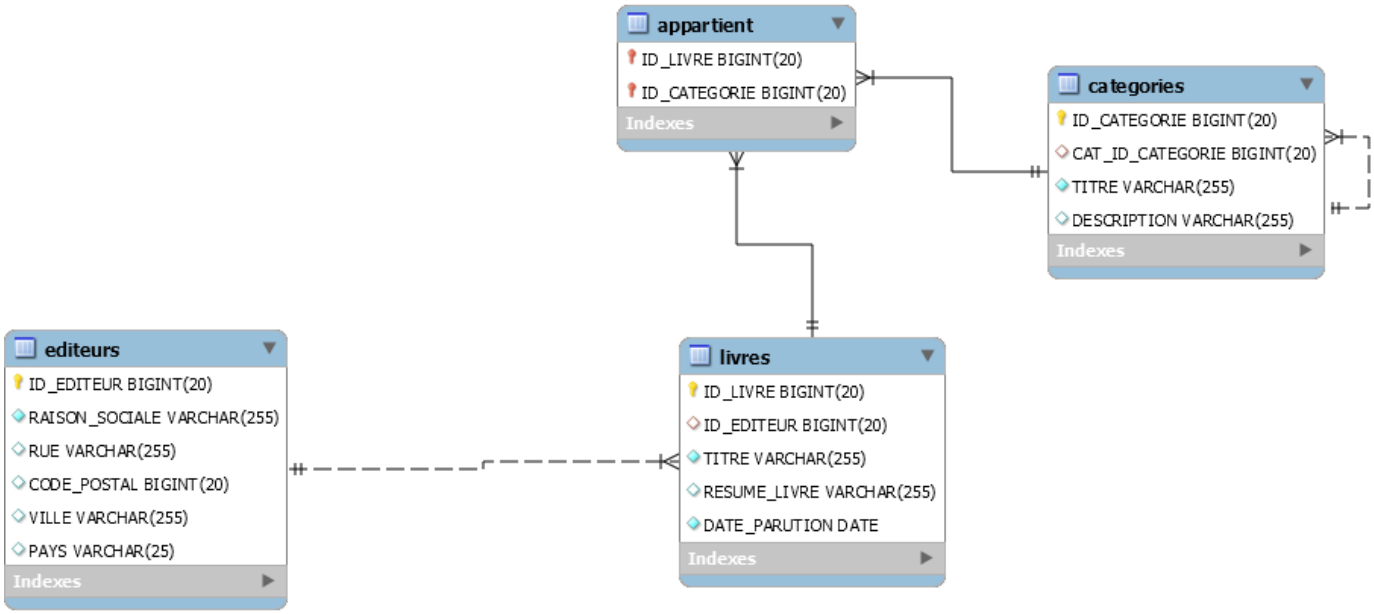


Un livre appartient obligatoirement à une catégorie.
Un livre peut appartenir à plusieurs catégories
Une catégorie contient 0 ou plusieurs livres.
Une catégorie a 0 ou une et seule catégorie parent
Un livre peut être associé à un éditeur.
Un éditeur a 0 ou plusieurs livres qui lui sont associés.
Un éditeur a une seule adresse.

Appréhendez-bien la multiplicité des relations basée sur les exigences présentées ci-dessus ainsi que la navigabilité de celles-ci.

Prenez enfin quelques minutes pour comparer le modèle objet ci-dessus avec le MCD (MCD.pdf) et le modèle Entités-Relations étendu (ERModel.jpg) fournis en ressource dans le dossier bdd.
Le modèle Entité-Relations ci-dessous a été généré via l'option Reverse Engineer du menu Database de MySQL workbench. Dans le dossier bdd, vous trouverez aussi le modèle en version lisible avec MySQL Workbench, EERModel.mwb. Ce Modèle utilise la notation crow's foot (pate de corbeau) pour la représentation des cardinalités.

Atelier
Persistance avec JPA



La relation plusieurs-plusieurs entre *categories* et *livres* est mise en place grâce à une table de jointure *appartient* contenant une clé primaire composée des 2 clés étrangères référençant respectivement les clés primaires des tables *categories* et *livres*.
La relation entre *livres* et *editeurs* est mise en œuvre via la clé étrangère *livres.ID_EDITEUR* référençant la clé primaire d'*editeurs*.
Les informations relatives à l'adresse de l'éditeur sont stockées dans la table *editeurs*.
Enfin, la relation parent / enfant dans catégorie est modélisée par la clé étrangère *categories.CAT_ID_CATEGORIE* pointant vers la clé primaire de la même table. Les champs de cette clé étrangère contiennent la valeur de la clé primaire de la catégorie parent.
Dans la base relationnelle le champ *titre* des tables sous-jacentes *livres* et *categories* ne peut être NULL. De même, la date de publication d'un ouvrage ne peut être NULL.