Architecture du projet

I. Outils utilisés

Environnement de compilation : JDK 1.7
IDE : Netbeans (version > 7.0)
Outil de contrôle et partage de code source : GitHub
Outil d'intégration continue : BuildHive
Type de projet : Maven
Serveur WEB : Apache Tomcat (version > 7.0)
Serveur de base de données : WampServer
Type Base de données : MySQL (mysql-connector-java-5.1.13)
Driver Base de données : JDBC (jdbc-stdext-2.0)
Port Base de données : 3306
WebServices : Allociné
JPA : Hibernate (Hibernate-3.2.5-ga, Hibernate-entitymanager-3.3.2-ga)

II. Pattern mis en place

Le pattern action (=Command Pattern) :

- chaque action implémente l'interface Action définissant une méthode « execute() » pour chacune. C'est la servlet qui appellera cette méthode avant d'afficher la vue retournée par la méthode.

- les actions peuvent ainsi faire le lien entre les vues (jsp) et la couche métier afin de réaliser les traitements d'affichage et ceux sur le modèle nécessaires. Ce sont les actions qui récupéreront les paramètres passés aux requêtes via les formulaires des jsp (form) grâce à la méthode getParameter(). Elle récupéreront et modifieront également les attributs de la requete et de la session en conséquence via les méthodes setAttribute() et getAttribute().

III. Architecture du code et de la base

Classes du code :

- **Film** : id (string), titre, date_sortie (date), duree, realisateur (string), acteurs (string), note_utilisateurs (float), note_presse (float), url_affiche (string), url_bande_annonce (string), pays (liste <string>), genres (liste <string>), synopsis (string)
- **Cinema** : id(string), nom, adresse,code_postal, ville, nombre_salles (int), position (Geolocalisation), url_image , seances_films (liste<Seance_film>)
- **Geolocalisation** : latitude (float), longitude (float)
- **SeancesFilm** : film (film), seances(List<Seance>)
- **Seance** : langue, format, horaires (List<Horaire>)
- **Horaire** : jour (date), heures (liste <String>)
- Utilisateur : id (Long) pseudo, login, mdp, adresse, code_postal, ville, droit (Droit)
- CommentaireFilm : id (Long), utilisateur (Utilisateur), date (date), texte, id_film (string)
- CommentaireCinema : id (Long), utilisateur (Utilisateur), date (date), texte, id_cinema(string)
- FilmVu : id_utilisateur (Long), id_film (string)
- CinemaFrequente : id_utilisateur (Long), id_cinema (string)

Classes de la base : *Les classes du code en gras ne seront pas intégrées à la base car leurs informations seront récupérées via des WebServices.*

- Utilisateur : id, pseudo, login (=mail), mdp, adresse, code_postal, ville, #id_droit
- Droit : id, membre (booléen), moderateur (booléen), administrateur (booléen)
- Commentaire_film : id, #id_utilisateur, date, texte,#id_film
- Commentaire_cinema : id, #id_utilisateur, date, texte,#id_cinema
- Films_vus : #id_utilisateur, #id_film
- Cinemas_frequentes : #id_utilisateur, #id_cinema

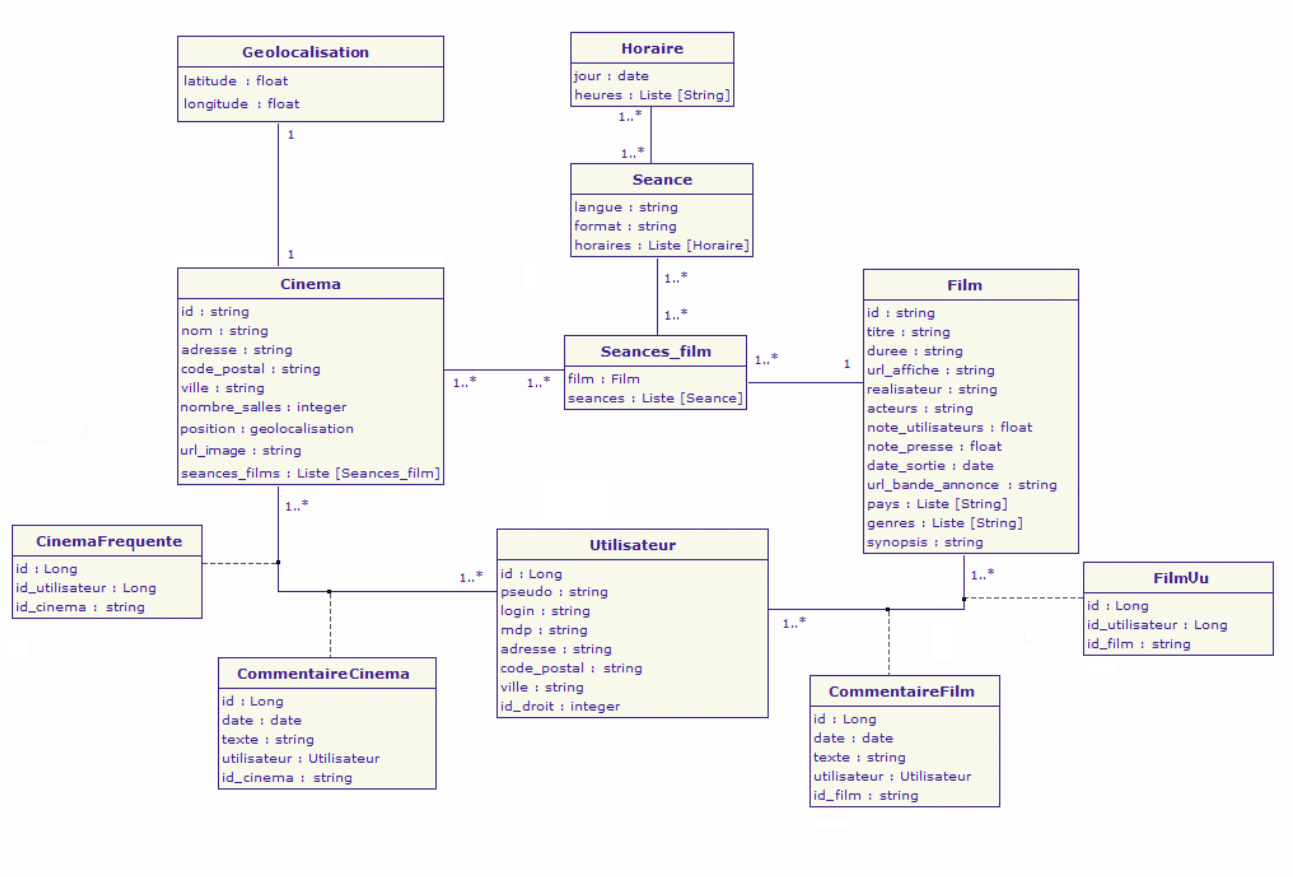
Commentaire [1]: j.dusseaut:
ça me semble prématuré de rentrer autant dans le détail. On bon design n'est pas créé à l'avance, il apparaît par affinements successifs.

bnoiraud:
c'est pour ça que cette partie là évolue toujours, pour le moment c'est juste une ébauche et seule deux classes ont véritablement été codées. Une fois qu'elles le sont nous modifions ce fichier en conséquence, et non l'inverse ...

j.dusseaut:
dans ce cas, mettre à jour le fichier est une perte de temps et une duplication d'informations.

ichraibi:
Le travail de groupe a distance nécessite de l'organisation : c'est donc afin de réaliser une tracabilité et d'avoir tout le temps un point de référence à disposition que nous avons ce fichier

Diagramme de classes du code :



IV. Accès aux données de la base SQL

On a choisi d'implémenter JPA (implémentation via Hibernate) qui définit des entity manager pour les entités à mapper en base et pour la manipulation de ces objets avec la base.

La configuration de la base de donnée se trouve dans le fichier src\main\resources\ META-INF\persistence.xml qui définit ainsi l'unité de persistence :

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="JPAPersistence" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>com.cineteam.cinebook.model.utilisateur.Utilisateur</class>
    <class>com.cineteam.cinebook.model.commentaire.CommentaireCinema</class>
    <class>com.cineteam.cinebook.model.commentaire.CommentaireFilm</class>
    <class>com.cineteam.cinebook.model.film.FilmVu</class>
    <class>com.cineteam.cinebook.model.cinema.CinemaFrequente</class>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="hibernate.connection.username" value="bdcinebook"/>
      <property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver"/>
      <property name="hibernate.connection.password" value="bdcinebook"/>
      <property name="hibernate.connection.url" value="jdbc:mysql://localhost:3306/BDCineBook"/>
      <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
    </properties>
  </persistence-unit>
</persistence>
```

Pour pouvoir effectuer des tests sur le mapping des objets en base, on reproduit cette configuration mais en configurant une base fictive via un autre fichier de persistence, définit dans le fichier src\test\resources\META-INF\persistence.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="JPAPersistence" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <class>com.cineteam.cinebook.model.utilisateur.Utilisateur</class>
        <class>com.cineteam.cinebook.model.commentaire.CommentaireCinema</class>
        <class>com.cineteam.cinebook.model.commentaire.CommentaireFilm</class>
        <class>com.cineteam.cinebook.model.film.FilmVu</class>
        <class>com.cineteam.cinebook.model.cinema.CinemaFrequente</class>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="hibernate.connection.driver_class" value="org.hsqldb.jdbcDriver"/>
            <property name="hibernate.connection.username" value="sa"/>
            <property name="hibernate.connection.password" value=""/>
            <property name="hibernate.connection.url" value="jdbc:hsqldb:mem:testdb"/>
            <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
            <property name="hibernate.hbm2ddl.auto" value="create-drop" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
        </properties>
    </persistence-unit>
</persistence>
```

A l'exécution des tests sur le mapping, ils n'appelleront pas le fichier persistence des sources mais celui des tests.