SER - Laboratoire 2 – Une séance

1/ Sérialisation XML avec Java jdom

2/ Sérialisation Json avec Java &Google/Gson

ERIC LEFRANÇOIS - 24 Mars 2017

Table des matières

1. OBJECTIFS DU LABORATOIRE 2

2. RENDU 2

3. CONTRÔLE DU LABO 2

4. LA STRUCTURE DU PROGRAMME PLEXADMIN 3

5. INSTALLATION DE PLEXADMIN ET PRISE EN MAIN 5

1. Objectifs du laboratoire

L'objectif de ce deuxième laboratoire consiste en la génération des documents XML et JSON élaborés dans le cadre du 1er laboratoire.

Dans les deux cas (XML comme JSON), les fichiers obtenus obéiront à un « pretty format » (retours de ligne et indentation des structures).

En particulier, il s'agira de reprendre l'application existante « `Plex_Admin` » en complétant le code de deux contrôleurs :

Génération du fichier XML

➔ Contrôleur `ControleurXMLCreation`, avec comme point d'entrée, - *pour la création du fichier XML* -, la méthode `createXML()`.

La génération du document XML s'appuiera sur la librairie `Jdom2`.

Génération du fichier JSON

➔ Contrôleur `ControleurMedia`, avec comme point d'entrée, - *pour la création du fichier JSON* -, la méthode `sendJSONToMedia ()`.

La génération du document JSON s'appuiera sur la librairie `Google/Gson`.

2. Rendu

En début de séance no 4 (début du laboratoire no 3), rendre un rapport papier contenant :

- Petite introduction
- Les modifications éventuelles apportées à vos structures XML et JSON élaborées dans le 1er laboratoire. Décrire alors l'objectif de ces modifications.
- Présenter le code des contrôleurs `ControleurXMLCreation et ControleurMedia`

  Commentez votre solution (méthodes rajoutées, complétées, leur objectif, points remarquables).
- Présenter un extrait représentatif des deux fichiers que vous aurez générés.
- Bilan - Conclusion

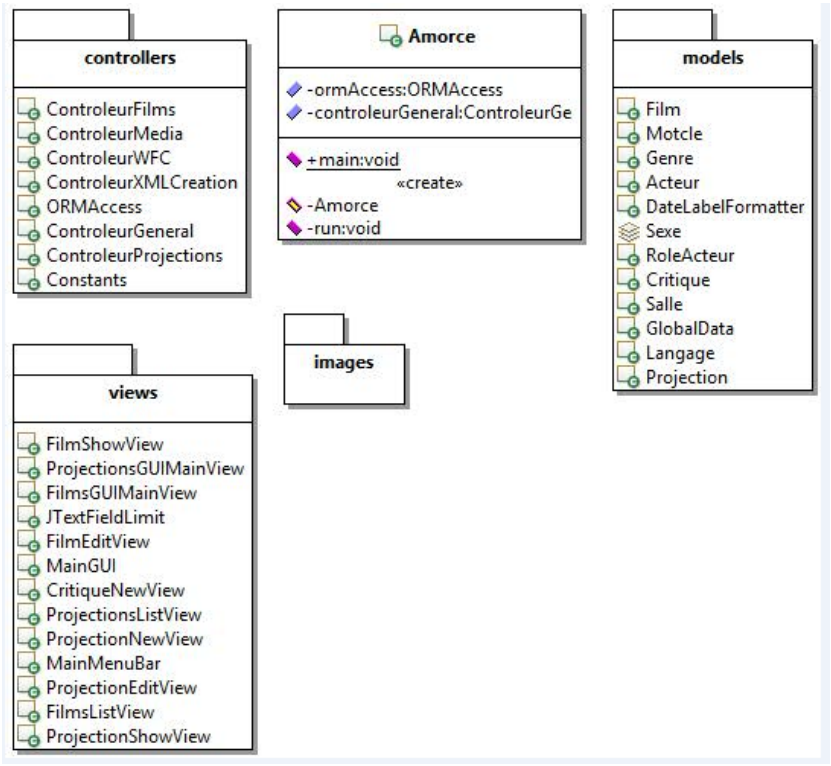
En plus du rapport au format papier, seront communiqués par email :

- Les fichiers source des contrôleurs `ControleurXMLCreation.java et ControleurMedia.java`
- Le fichier XML et le fichier JSON qui auront été générés par votre application PlexAdmin.

3. Contrôle du labo

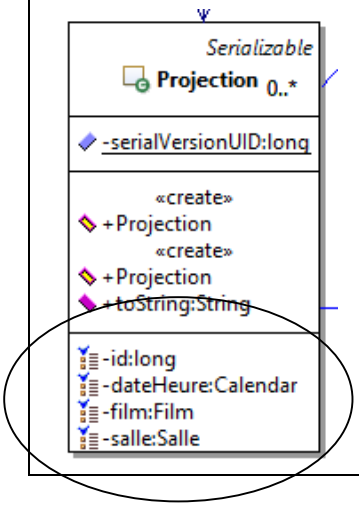
En plus du rendu décrit dans le paragraphe précédent, la génération des fichiers XML et JSON sera présentée au professeur ou à l'assistant (prévoir environ 5 mn de démonstration).

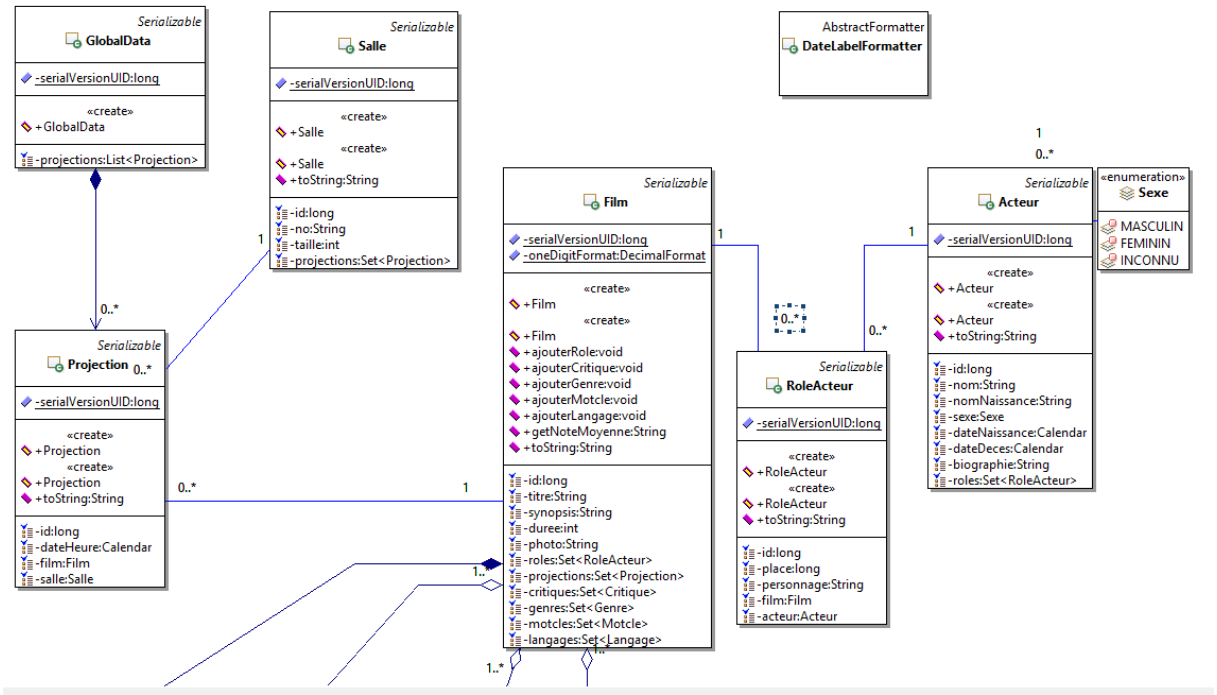
4. La structure du programme PlexAdmin

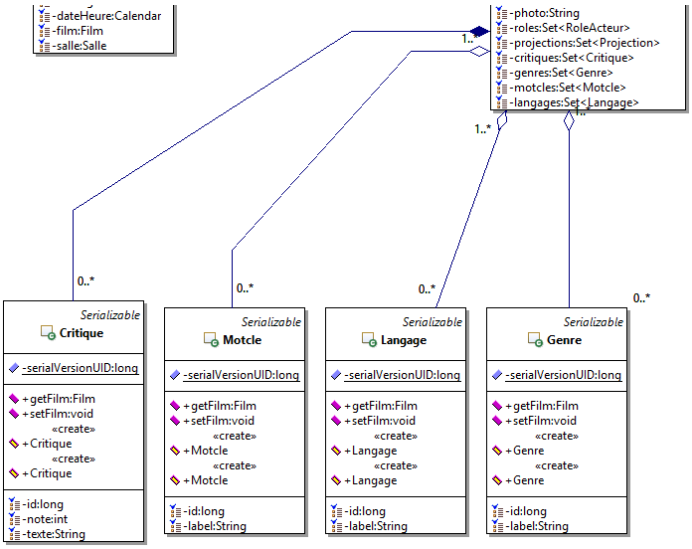


En particulier, dans les classes « `models` », la classe `GlobalData` est une structure qui permet d'obtenir la liste des projections actuellement planifiées par l'administrateur du complexe « Pathé-Flon ».

Voir le diagramme de classes présenté ci-après. Chacune des classes comporte une partie inférieure qui décrit la liste de ses « propriétés ». Ces propriétés sont accessibles par le biais d'accesseurs (getters et setters). Par exemple, la class `Projection` possèdent les getters suivants :

	Liste des getteurs
	
	`getId()`
	
	`getDateHeure()`
	
	`getFilm()`
	
	`getSalle()`





Pour générer vos structures XML et Json, vous aurez à disposition une variable de type `GlobalData` dont la valeur aura été directement chargée depuis la base de données.

Vous pourrez ainsi, pour chacune des projections, obtenir la salle et le film. Pour chacun des films projetés, obtenir la liste des critiques, mots-clés, langages et genres, ainsi que la liste des rôles, associés chacun à un acteur.

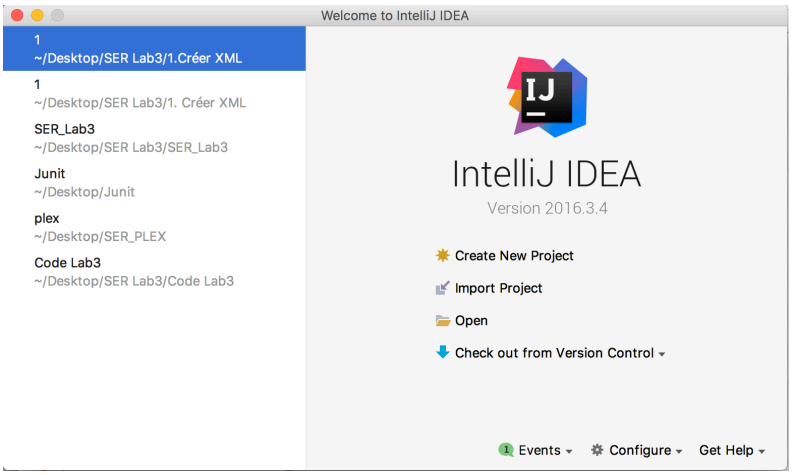
⚠ *Attention ! Seules les données relatives aux projections actuellement planifiées seront chargées dans la variable `GlobalData`. Par contre, toute la base de données n'est pas chargée dans cette structure.*

Par exemple, si vous planifiez une seule projection, la projection du film « Ocean Eleven », l'acteur Brad Pitt sera présent dans la structure de données. Si vous envoyez le message `getRoles()` *à cet acteur, la méthodes vous retournera la liste des rôles de Brad Pitt pour d'autres films enregistrés dans la base de données qui ne font pas partie de vos projections actuelles. L'accès à ces différents rôles – non présents dans la structure* `GlobalData` *-, retournera une erreur de type « Hibernate – Proxy Error – Lazy Access ».*

5. Installation de PlexAdmin et prise en main

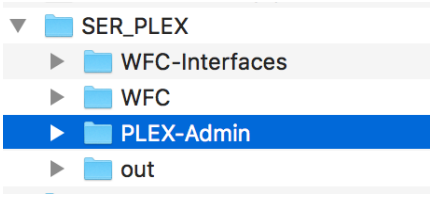
Déposer le dossier `SER_PLEX` confié par le professeur dans un de vos dossiers.

Ouvrir IntelliJ



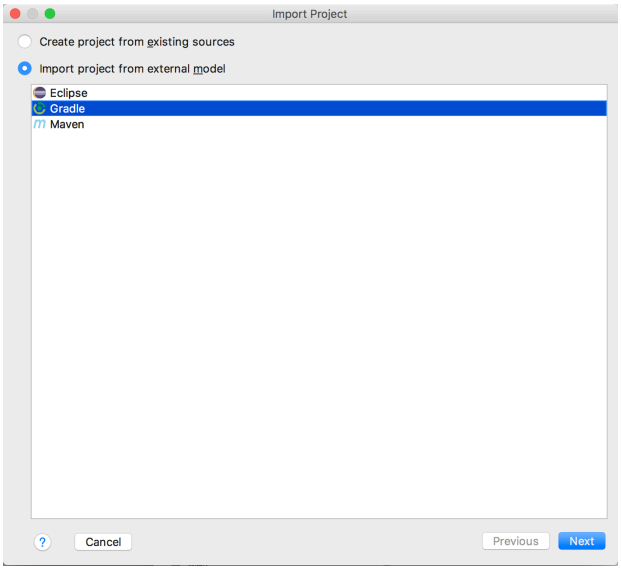
Import Project

⇒ Sélectionner `SER_PLEX/PLEX-Admin`

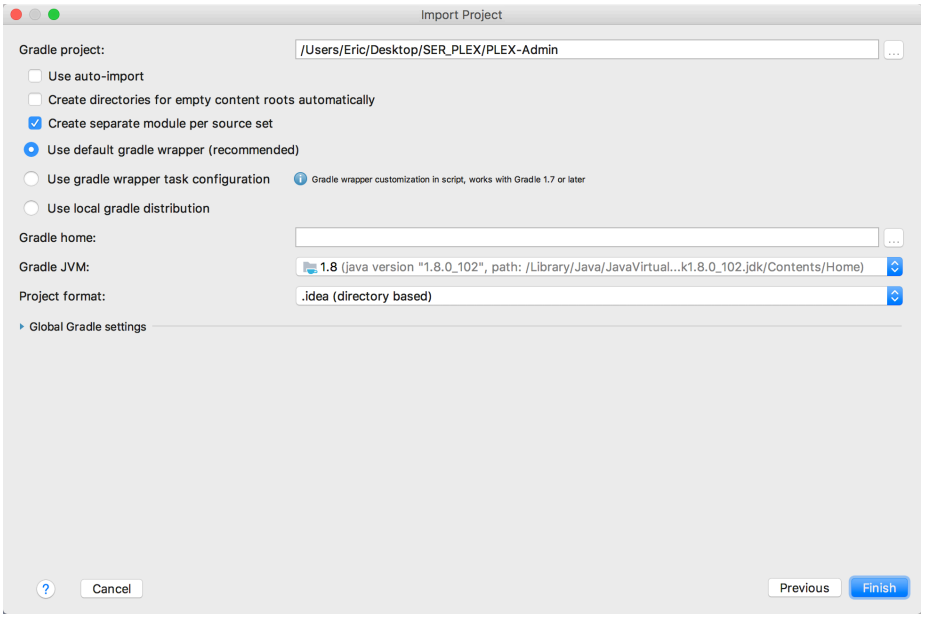


Le projet à importer est un projet Gradle

Sélectionner le type de projet Gradle

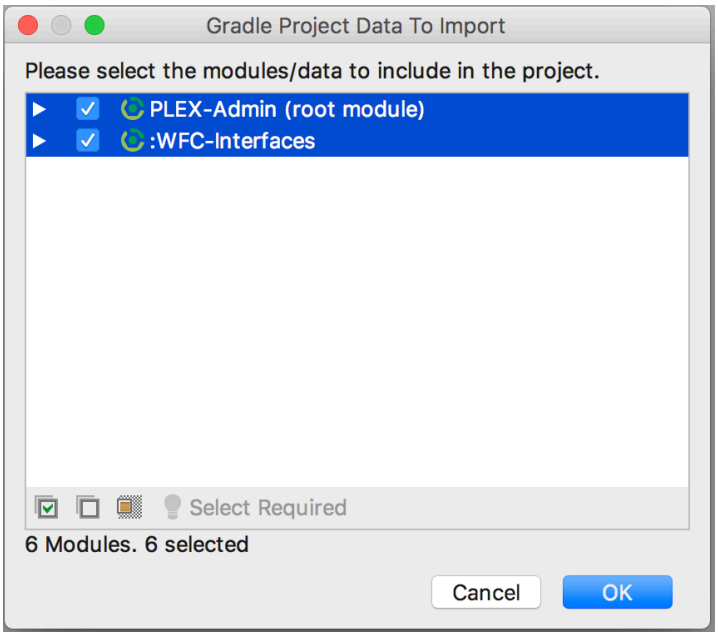


Appuyez sur **Next**



Appuyez sur **Finish**

Puis, comme le projet `PLEX-Admin`, - *qui contient le code que vous allez compléter* -, référence une interface décrite dans le projet `WFC-Interfaces`, le projet `WFC-Interface` doit lui-même être importé. Sélectionnez les deux projets dans la fenêtre que vous propose IntellIJ :



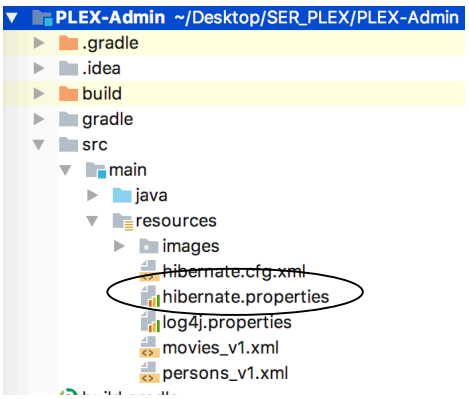
Appuyez sur **OK**

Créer une base de données nommée « `plex_admin` » dans MySQL

Créer cette base avec un type d'encodage UTF-8 (Collation : UTF-8)

Configurer l'application `Plex_Admin` pour qu'elle puisse se connecter à votre base de données.

A cette fin, ouvrez le fichier `hibernate.properties` :

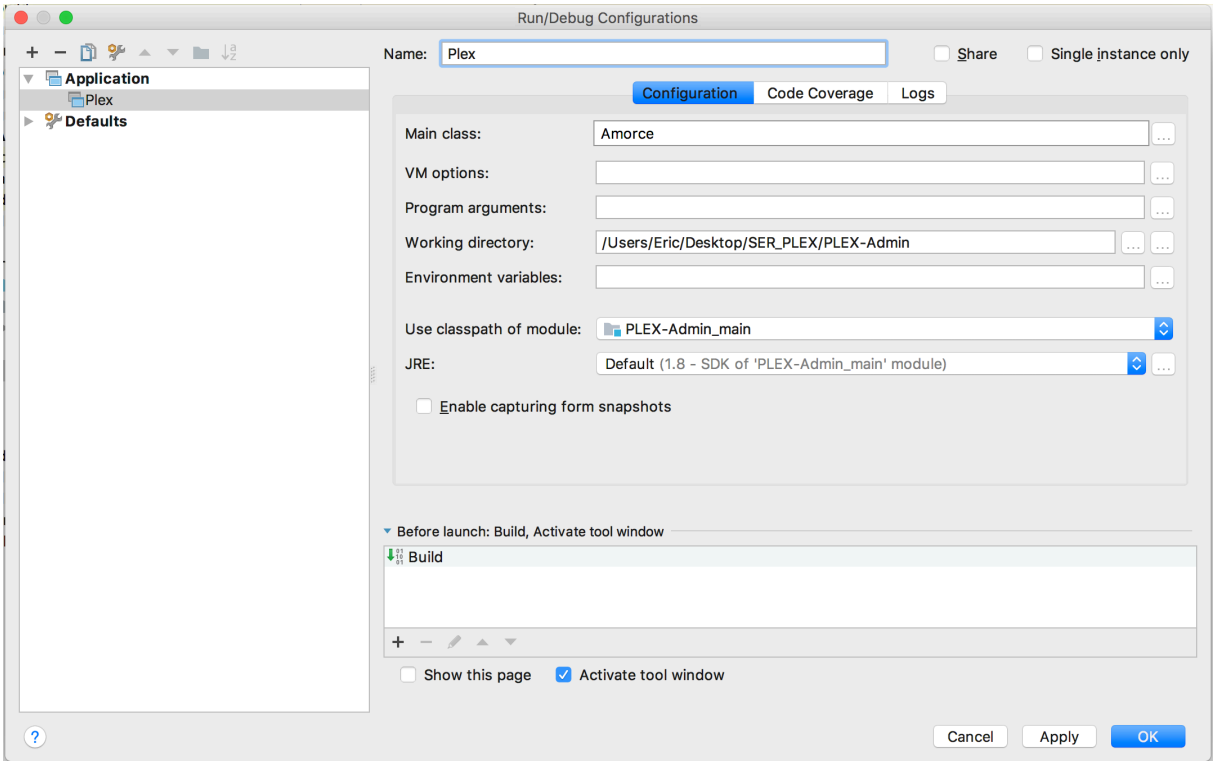


Et configurez les valeurs des champs `hibernate.connection.username` & `hibernate.connection.password` avec le `username` et le `password` de votre connexion MySQL :

```
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url=jdbc:mysql://localhost:3306/plex_admin?
hibernate.connection.username=votre_user_name
hibernate.connection.password= votre_password
hibernate.connection.pool_size=1
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.show_sql=false
hibernate.hbm2ddl.auto=update
hibernate.connection.isolation=2
```

Pour compiler et exécuter le projet, créer une configuration d'exécution en choisissant « Application »

Puis remplir les champs en sélectionnant le projet `PLEX-Admin`, le module `PLEX-Admin_main`, avec la classe `Amorce` comme point d'entrée (fonction main).



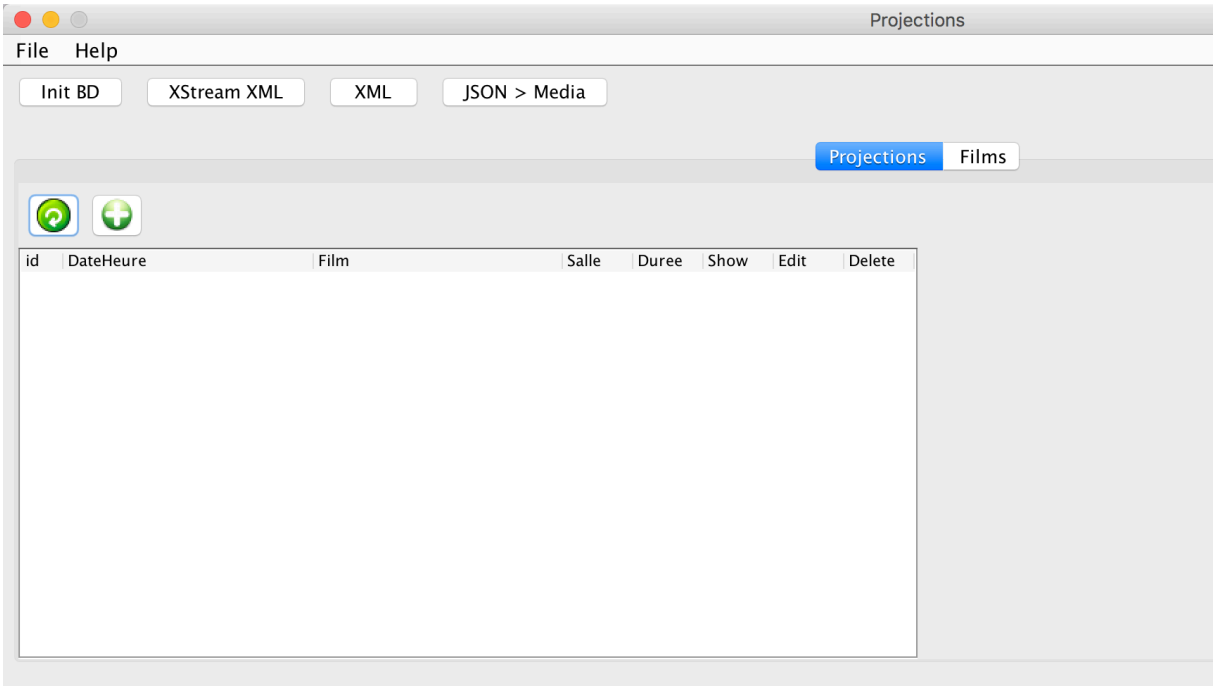
Exécuter le programme.

Viendrons alors s'afficher en console quantité de messages du style :

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_102.jdk/Contents/Home/bin/java ...
mars 27, 2017 7:32:45 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {4.0.5.Final}
mars 27, 2017 7:32:45 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {4.3.6.Final}
mars 27, 2017 7:32:45 PM org.hibernate.cfg.Environment <clinit>
INFO: HHH000205: Loaded properties from resource hibernate.properties: {hibernate.connection.driver_class=com.mysql.
mars 27, 2017 7:32:45 PM org.hibernate.cfg.Environment buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
mars 27, 2017 7:32:45 PM org.hibernate.cfg.Configuration configure
INFO: HHH000043: Configuring from resource: hibernate.cfg.xml
mars 27, 2017 7:32:45 PM org.hibernate.cfg.Configuration getConfigurationInputStream
INFO: HHH000040: Configuration resource: hibernate.cfg.xml
mars 27, 2017 7:32:45 PM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Configured SessionFactory: null
```

Ne pas s'inquiéter ! Le programme cherche des tables dans la base de données qui n'existent pas encore. Il les crée alors automatiquement.

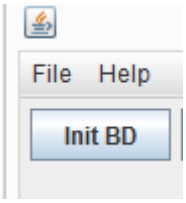
Vous devez obtenir dès lors l'interface suivante :



En l'état, vous pouvez constater que les tables ont été créées dans la base de données (salles, projections, films, etc..).

Ces tables sont vides.

Pour les remplir avec les données initiales, appuyez sur le bouton « Init BD »



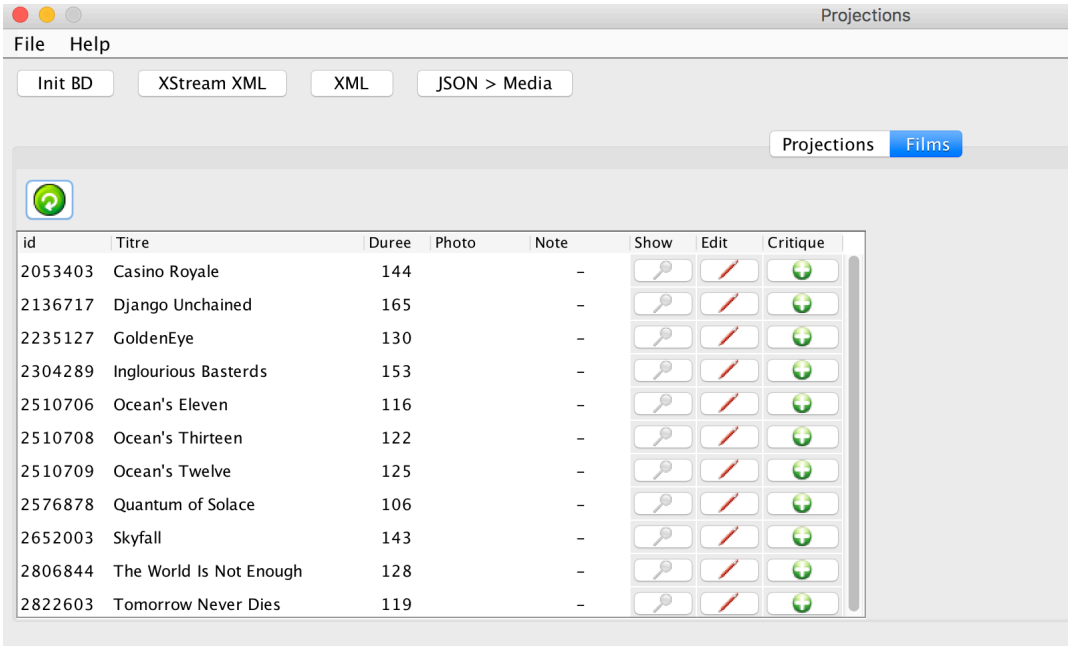
Cette action a pour effet d'effacer tout le contenu des tables et de les peupler avec le contenu de deux fichiers XML qui se trouvent dans le répertoire « `resources` » de votre projet : Les fichiers `movies_v1.xml` et `persons_v1.xml`

En console, vous obtenez l'affichage suivant :

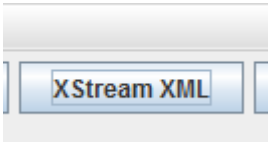
```
INFO: HHH000232: Schema update complete
Chargement du contenu XML...
Chargement du contenu XML: OK
Enregistrement des films ...
Enregistrement des films: OK
Enregistrement des acteurs ...
Enregistrement des acteurs: OK
Enregistrement des roles ...
Enregistrement des roles: OK
```

Vous pouvez répéter cette opération(`Init BD`) autant de fois que le désirez !

En allant dans l'onglet « `Films` », vous voyez alors apparaitre une dizaine de films :



Appuyez maintenant sur le bouton XStream XML :



Est alors créé un fichier `Global_data.xml` dans la racine de votre projet qui contient la liste des projections actuelles.

Ce fichier est obtenu en opérant la sérialisation d'un objet Java qui contient la liste actuelle des projections. Cette sérialisation est obtenue en utilisant la librairie XStream (le pendant de la librairie Google/Gson pour XML).

Voir le code de la méthode `createXStreamXML` `du contrôleur` `ControleurXMLCreation.`

Amusez-vous avec l'application, ajoutez quelques commentaires critiques sur vos films et créez une ou deux projections ..

Puis appuyez à nouveau sur le bouton XStream XML et consultez le fichier `Global_data.xml` qui aura été généré.

Vous pouvez alors passer à la réalisation de votre laboratoire.

La pression des boutons `XML` et `JSON > Media` devront avoir pour effet, respectivement, de générer le fichier XML et le fichier Json.

La pression du bouton XML invoque la méthode `createXML()` du contrôleur `ControleurXMLCreation.`

La pression du bouton JSON > Media invoque la méthode `sendJSONToMedia ()` du contrôleur `ControleurMedia.`