Exercices sur les bases de données : ExBdd

- **Exercice 1 : [Répertorier vos requêtes dans un script]**
  - **1.1** Générer votre base de données à l'aide du script Shop.sql
  - **1.2** Effectuer les requêtes permettant d'afficher toutes les tables en base
  - **1.3** Trouver un moyen pour décrire une table
  - **1.4** Ajouter à la table des articles des occurrences de votre choix
  - **1.5** Modifier un article avant de vérifier si c'est pris en compte
  - **1.6** Supprimer un article puis vérifier
  - **1.7** Sélectionner tous les articles dont le prix est supérieur à 100
  - **1.8** Sélectionner les articles dont le prix est compris entre 50 et 150
  - **1.9** Afficher les articles dans l'ordre croissant des prix
  - **1.10** Afficher uniquement la description des articles
  - **1.11** Choisissez une requête particulièrement intéressante à présenter aux autres
  - **1.12** Ajouter la table des catégories à votre base de données et insérez-en
  - **1.13** Trouver la requête qui permet d'obtenir le résultat suivant :

IdArticle	Description	brand	UnitaryPrice	CatName
12	WebCam	Logitoch	0	Materiel info
11	Casque Audio	Syno	105	Materiel info
16	Office	Microsoft	150	Logiciel
14	S10	Samsung	2 000	Smartphone
13	Macbook	Apple	2 000	PC
15	Iphone50	Apple	20 000	Smartphone

- **Exercice 2 :** Il s'agit maintenant d'afficher le contenu de votre table des articles sous Eclipse aussi reprenez l'exemple vu en cours sans omettre d'ajouter le driver de mariaDB* puis exécuter votre programme. La classe Article contient les mêmes données que la table des articles en base, à savoir : identifiant, description, brand, price

  (*) : Comment ajouter une librairie à un projet dans Eclipse une fois le bon JAR téléchargé, contenant donc les pilotes jdbc appropriés (copier-coller dans un rep lib par ex) :
  → Sur la perspective « project explorer », clic droit sur le jar « add to build path »

- **Exercice 3 :** Nous souhaitons maintenant, toujours sous Eclipse, réaliser une requête d'insertion d'un article, une autre de mise à jour, une autre qui supprime un article et une

dernière qui nous renvoi toutes les infos d'un article. Vous devez afficher tous vos articles sous Eclipse ou vérifier en ligne de commande que cela a fonctionné en base.

- **Exercice 4 :** Nous souhaitons dorénavant utiliser un fichier de configuration unique contenant toutes les informations de connexion. En effet, afin de rendre notre application ouverte aux évolutions notamment lorsqu'il s'agit de migrer d'un SGBD à un autre sans avoir à toucher au code source, un fichier de configuration contiendra toutes les informations nécessaires comme ici :

```
db.driver.class = org.mariadb.jdbc.Driver
db.url = jdbc:mariadb://localhost:3306/Shop
db.login = le votre
db.password = le vôtre
```

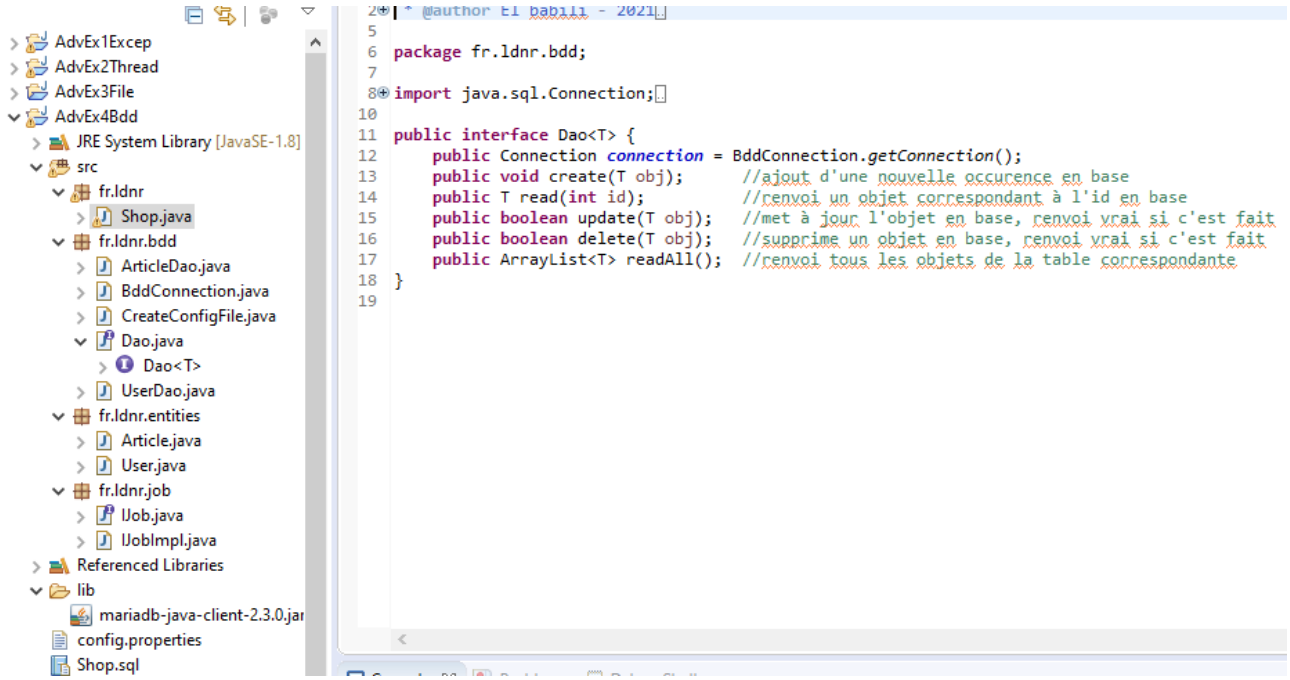
  Dans un premier temps, il faudra le créer/générer à l'aide d'une classe CreateConfigFile puis dans un second temps l'utiliser pour exploiter votre base de données, vérifier que le tout fonctionne parfaitement. Pour vous aider, voir l'exemple ici

- **Exercice 5 :** LE CODE DEVIENT ILLISIBLE, N'EST-CE PAS ?

Reprenons donc nos bonnes habitudes et découpons notre appli à l'aide des packages :

5.1 Ajouter le package fr.fms.entities et ajoutez-y la classe Article

5.2 Ajouter le package fr.fms.dao qui va faire le lien entre notre application et la base de données, ajoutez-y l'interface générique Dao (ci-dessous) contenant les méthodes dites CRUD pour Create, Read, Update et Delete, puis ajouter la classe ArticleDao qui implémente l'interface générique Dao : `public class ArticleDao implements Dao<Article> {...}`



5.3 *Capturer et relayer les exceptions avec des messages explicites (Privilégier le try with ressources)*. Puis dans votre programme principal tester votre implémentation ArticleDao.

- **Exercice 6 :** Allons plus loin en permettant uniquement aux personnes autorisées d'avoir accès à notre boutique. Pour ce faire, vous devez ajouter une classe Utilisateur (autorisées) et son homologue en base dont voici le contenu :

```
CREATE TABLE T_Users (
    IdUser          int(4)          PRIMARY KEY AUTO_INCREMENT,
    Login           varchar(20)     NOT NULL,
    Password        varchar(20)     NOT NULL,
) ENGINE = InnoDB;
```

- **Exercice 7 :** A vous le soin de l'ajouter en base et d'effectuer des insertions d'utilisateurs, vous pouvez modifier le script Shop.sql par exemple et le régénérer mais attention aux doublons !

- **Exercice 8 :** « Plus haut », côté Eclipse, tout comme pour la classe Article, vous devez ajouter une classe User qui contient les attributs correspondants. Puis dans le package dao, rajouter la classe UserDao qui implémente aussi Dao, tester les méthodes Create, Read, Update, Delete.

- **Exercice 9 :** Sans retirer les éléments de configuration liés à la connexion, vous devez mettre en œuvre le pattern singleton afin d'ouvrir une seule connexion pour tous, il faudra étudier comment fonctionne ce pattern avant de le mettre en œuvre ici.

- **Exercice 10 :** Trouver un moyen pour permettre dorénavant aux seuls utilisateurs en base de consulter la liste des articles. En effet, votre application propose de saisir login et password pour vérifier si vous avez accès, si c'est le cas, l'utilisateur pourra visualiser la liste des articles. Ajouter vos noms et pwd à la table puis vérifier si ça marche aussi pour vous.

- **Exercice 11 :** Vous étiez root jusqu'ici donc administrateur système aussi faites en sorte de permettre l'accès à un utilisateur uniquement sur la base de données Shop.

- **Exercice 12 :** Enfin, d'après vous, quelles seraient les méthodes d'une interface représentant la couche métier de notre appli de E-commerce ici ?

- **Exercice 13 :** Implémenter votre interface puis réaliser une application permettant l'achat d'articles dans une boutique en remplissant un panier avec la possibilité de passer commande à tout instant…