Faculté des Sciences
Aix*Marseille Université

Département Informatique et Interactions
Année 2016-2017

Master 2 FSIL

Fiabilité Logicielle 1

Test sur Base de Données

L'objectif de ce TP est de comprendre comment effectuer des tests unitaires sur des programmes Java utilisant une base de données. Il servira à voir comment effectuer ces tests en utilisant directement JUnit et la base de données dans un premier temps, puis à voir comment utiliser un outil *dbunit* pour effectuer des tests unitaires avec JUnit sur une image de la base (un fichier XML).

1 Travail à rendre

Ce TP est à effectuer de préférence en binôme. À la fin du TP (10/10 à 12h), vous enverrez par mail un zip contenant vos sources (java et fichiers xml utilisés lors des tests). Vous pouvez ajouter un rapport en pdf répondant aux questions du tp, incluant une description succincte de votre bdd, expliquant votre démarche et/ou des points techniques. Si vous n'avez pas eu le temps de terminer le TP, vous pourrez effectuer un deuxième rendu avant le lundi 17/10 9h.

- **Destinataire** : romaric.duvignau@lif.univ-mrs.fr; **Sujet** : [FIAB1] NOM prénom
- **Pièces jointes** "NOM prénom.zip", "NOM prénom.pdf", ...

2 Une base de données

Créer[1] une base de données permettant de gérer les soutenances de stage des étudiants sachant que :

1. Un étudiant a un nom, prénom, statut (fsi ou isl) et un numéro d'étudiant.
2. Une plage de soutenance a un jour (1 à 5), une plage horaire (1 à 8 pour 4 plages le matin de 8h à 12h, 4 plages l'après-midi de 14h à 18h) et un numéro de salle.
3. Une soutenance a un titre, un étudiant et une plage de soutenance.

3 Une application simple Java

La première partie du TP montre qu'on peut utiliser directement JUnit pour effectuer des tests d'un programme Java qui utilise une base de données.

Afin d'accélérer le cycle de développement, vous pourrez effectuer en parallèle le développement de l'application (points a) et l'écriture du jeu de test (points b). Si vous ne souhaitez ou ne pouvez utiliser qu'un poste de travail, vous pouvez utiliser la méthode du *ping-pong programming* (un développeur écrit un test qui échoue, puis l'autre écrit le code attendu et *répond* en écrivant le test suivant, etc) variante de la célèbre méthode agile de programmation par binôme.

[1] Le SGBD et l'interface utilisée sont laissés au libre choix des étudiants.

3.1 Interaction base de donnée

a) Écrire une classe *InteractBD* qui établit une connection avec votre base de données et contient les méthodes *connect(), deconnect(), isConnected()*. Voir la section 5 de rappel sur JDBC en page 3 pour avoir des rappels sur l'utilisation d'une base de donnée par Java.

b) Définir une suite de tests pour tester les aspects connection à la base et écrire la classe JUnit *InteractBDTest* correspondante. Vérifier que les tests passent.

3.2 Une application Java

La base de données est utilisée par une application Java. Celle-ci comporte les classes *Etudiant, Soutenance, Plage* et *Admin*. Les trois premières comportent les *getters* et *setters*, la dernière sert à administrer les soutenances et d'accéder à la base de données. Elle permet d'ajouter ou supprimer un étudiant, une soutenance ou une plage de soutenance. Ajout ou suppression lèvent une exception (à définir) si les objets existent déjà ou n'existent pas. L'administration a également la fonction *modifiePlageSoutenance(Soutenance sout, Plage oldPlage, Plage newPlage)* permettant de remplacer la plage horaire *oldPlage* d'une soutenance *sout* par une nouvelle plage horaire *newPlage*.

a) Écrire les classes de cette application qui utilise la base de données précédente.

b) Étendre la suite de tests avec les tests permettant de vérifier le bon fonctionnement des fonctions :

- de création d'une soutenance, étudiant, plage de soutenance,
- de suppression d'une soutenance, étudiant, plage de soutenance,
- de modification d'une soutenance.

Cette approche permet de faire des tests unitaires sur des données persistantes mais elle présente plusieurs inconvénients majeurs. Lesquels ? Un spécialiste du test considère que tester une application qui utilise une base de données demande 4 bases différentes (si les tests sont effectués par une équipe). Pouvez-vous identifier chacune de ces bases de données et définir comment elle sont utilisées ?

4 Ecriture de tests avec DbUnit

DbUnit est un logiciel permettant d'automatiser d'effectuer des tests unitaires sur les bases de données en utilisant Junit et en utilisant un format interne à l'outil pour représenter les données d'une base de données. Les sources (prendre la version 2.4.9[2] ainsi que la documentation, notamment la javadoc se trouvent sur le site et sont à installer dans un répertoire

[2] la version 2.5.* demande à utiliser maven

personnel. Les .jar seront à ajouter aux bibliothèques du projet (il peut être nécessaire d'installer d'autres .jar: par exemple `slf4j-simple-1.7.21.jar` et `slf4j-api-1.7.21.jar` de SLF4J). Les données récupérées par dbunit sont dans des fichier de format *flat XML file* qui peuvent être lus ou écrits (à la main ou par l'outil à partir d'une base de données). Cela va permettre d'initialiser les bases de données systématiquement, puis de pouvoir comparer les résultats obtenus aux résultats souhaités. Un test lancera l'opération à tester sur une base de données initialisée avec les bonnes données et les résultats obtenus se comparera avec ce qui est attendu (résultats et valeurs attendues étant dans le même format de données IDataSet). Un élément de ce type s'obtient soit via une connection à la base, soit par lecture d'un fichier flat XML.

1. Lire la documentation *dbunit* qui explique le fonctionnement et donne des exemples de classes de test. Avant de commencer à écrire la moindre ligne de code, il est fortement conseillé de lire les exemples et la partie Core Components.

2. Réécrire les tests précédents en utilisant *dbunit*. Quelques informations utiles (cf documentation).

    - L'interface *IDatabaseConnection* représente une connection avec une base de données.
    - L'interface *IDataSet* représente une suite de tables. Un objet implémentant ce type se crée à partir de la lecture d'un fichier xml avec la méthode *build* de *FlatXmlDataSetBuilder()*.
    - La classe abstraite *DataBaseOperation* représente une opération effectuée avant et après chaque test.
    - L'initialisation (SetUp) et le nettoyage (TearDown) pour chaque test peuvent s'effectuer avec *getTearDownOperation* et *getSetUpOperation*.

Remarque

Vous pouvez reprendre l'architecture du TP pour qu'elle soit plus conforme à une application Java trois-tiers (dans ce cas expliquer en détail l'architecture et les fonctions des différents packages).

5 Rappel: utilisation de base de données via JDBC

Les classes java peuvent dialoguer avec une base de données grâce aux classes de JDBC (Java Data Base Connectivity) et exécuter des commandes SQL (importer java.sql.*), se reporter au cours J2E pour plus de détails. Il faut charger le pilote correspondant au SGBD qui devra être sur les chemins de recherche (via CLASSPATH ou lib/ext de l'installation de la machine virtuelle java); dans eclipse, on clique sur *Build Path* → *Add External Archives....* L'instruction *Class.forname("nom_du_driver");* initialise le Driver manager. Un objet de type *Connection* est créé avec l'appel *getConnection(url,user,passwd)* sur l'objet *DriverManager* avec *url* la chaine donnant l'adresse de la base décrite par
jdbc:type_de_la_base://adresse_IP/nom_de_la_base
en local, prendre *localhost* pour *adresse_IP*, *user* est le nom d'utilisateur, *passwd* son mot de passe. Les méthodes utiles sur l'objet de type *Connection* sont *close(), createStatement()*. Les exceptions déclenchées sont usuellement de type *SQLException*.

Exemple : MySQL

La page Fobec suivante `http://www.fobec.com/java/943/connecter-une-base-mysql-avec-driver-jdbc.html` donne un exemple complet de connexion à une base de donnée mysql. Pour télécharger le driver mysql, allez sur `http://dev.mysql.com/downloads/connector/j/5.1.html`. Pensez à inclure la bibliothèque à votre projet eclipse. Pour supprimez les warning SSL, vous pouvez désactivez complétement SSL via l'url jcdb en ajoutant en fin `?useSSL=false`.