Les **tests unitaires** valident les **briques de base**

Rédacteurs :	
Kévin LEBRETON	**Thomas RAMBALDI**

	REVISIONS	
Date	Nature de la modification	Version
24/10/16		1.1

Table des matières

1 Test unitaires "JdbcTools"

1.1 Méthodes newConnection

1.1.1 Description

Le but de ce test est de garantir une connexion fonctionelle à la base de donnée. Nous disposons d'une base de donnée locale (MySQL). Des tests de connexion valide et invalide seront réalisés.

1.1.2 Contrainte

Les personnes doivent installer MySQL. L'utilisation d'une base de donnée locale nous garanti une connexion stable mais si une personne du projet modifie la base de donnée, elle doit avertir les autres membres du groupe pour qu'ils effectuent eux aussi les modifications.

	Valide	Invalide
public Connection newConnection()	Url, user et password sous forme de chaine de caractère	Autre qu'une chaîne de caractère

1.1.3 Dépendances

Le test à mener au préalable est : **public void** InitTest() ainsi que les tests sur les setters

1.1.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void newConnectionTest()	Url, user et password sous forme de chaine de caractère	Connexion à la BDD établie	Les données d'entrées sont valides
Public void connectionFalseTest()	Url, user et password sous forme de chaine de caractère	Envoie une erreur SQLException	Les données d'entrées sont invalides

1.2 Méthodes QuietClose

1.2.1 Description

Le but de ce test est de tester la déconnexion à la base de données.
Une connexion "null" est une connexion déjà fermée.

1.2.2 Contrainte

	Valide	Invalide
public void quietClose(Connection c)	Avoir une connection valide	Autre qu'une connection

1.2.3 Dépendances

Tester auparavant une connexion.

1.2.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public void quietCloseTest()	Une connexion	Déconnexion de la BDD	Une connexion construite avec des données valides

1.3 Méthodes executeUpdate

1.3.1 Description

C'est une fonction qui excute une requête SQL.

1.3.2 Contrainte

Avoir une base de donnée fonctionnelle.

Avoir les tables qui sont en relation avec les requêtes déjà créee.

	Valide	Invalide
public int executeUpdate(String query, Object ... parameters)	• Une chaîne de caractère, • 0 ou plusieurs paramètres Object	Autre q'une chaîne de caractère

1.3.3 Dépendances

Tester les cas de connexion

1.3.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public int executeUpdate(String query, Object ... parameters)	• Une chaîne de caractère, • 0 ou plusieurs paramètres Object	Exécution de la requête	Retourne toutes les informations relatives à la requête
public int executeUpdateSQLException(String query, Object ... parameters)	• Une chaîne de caractère, • 0 ou plusieurs paramètres Object	SQLException	Une requête SQL invalide

1.4 Méthodes isConnect

1.4.1 Description

Permet de savoir si la connexion est établie.

1.4.2 Contrainte

Avoir une URL valide qui nous dirige vers une base de donnée.

	Valide	Invalide
public boolean isConnect()	Être connecté ou non connecté	Url ne doit pas être vide

1.4.3 Dépendances

Aucune dépendance pour ce test

1.4.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public void isConnectTest()	Une connexion	True si la connexion est établie False sinon	-----
public void isConnectFalseTest()		SQLException	Avoir une url null

2 Test unitaires "Personnes"

2.1 Méthodes findAllPersons

2.1.1 Description

Tester si la fonction renvoie toutes les personnes se trouvant dans un groupe.
Création d'une collection contenant des personnes pour la comparer aux resultats retournés par la fonction.

2.1.2 Contrainte

Avoir une base de donnée fonctionnelle.
Il faut insérer au préalable des données dans la base pour avoir des résultats.

	Valide	Invalide
Collection<Person> findAllPersons(**long** groupId);	Entier positif	Différent d'un entier positif

2.1.3 Dépendances

Nous devons sauvegarder des personnes donc la fonction savePerson doit être testée auparavant

2.1.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public void findAllPersonsTest()	Entier positif représentant l'identifiant du groupe	Toutes les personnes au groupe en spécifié	Retourne toutes les personnes qui se trouve dans un groupe
public void findAllPersonsInGroupDontExistTest()	Entier positif représentant l'identifiant du groupe	Une collection vide	L'id groupe n'existe pas

2.2 Méthodes findPerson

2.2.1 Description

Tester si la fonction renvoie la personne p.
Création d'une personne p' pour la comparer aux resultats retourner par la fonction.

2.2.2 Contrainte

Avoir une base de donnée fonctionnelle.
Il faut insérer au préalable des données dans la base pour tester les résultats.

	Valide	Invalide
Person findPerson(**long** id);	Entier positif	Différent d'un entier positif

2.2.3 Dépendances

Nous devons sauvegarder des personnes donc la fonction savePerson doit être testée auparavant

2.2.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void findPersonTest()	Entier positif représentant l'identifiant d'une personne	La personne p est retournée	La personne se trouve dans la BDD et est renvoyée
Public void findPersonDontExistTest( )	Entier positif représentant l'identifiant d'une personne	Renvoie null	La personne n'existe pas dans la BDD

2.3 Méthodes SavePerson

2.3.1 Description

Test si la fonction insère dans la base de donnée la personne avec les informations données.

2.3.2 Contrainte

Entrer les informations de la personne (par exemple email, naissance, ...) dans un format défini.

	Valide	Invalide
void savePerson(Person p);	Une personne	Différent d'une personne

2.3.3 Dépendances

Aucun dépendances.

2.3.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void savePersonTest()	Une personne	Insère une personne dans la BDD	La personne p se trouve dans la base de donnée
Public void savePersonAlreadyExistTest()	Une personne	MySQLConstraintViolationException	La personne qu'on veut insérer possède un id existant

2.4 Méthodes DeletePerson

2.4.1 Description

Test si la fonction deletePerson supprime une personne p de la base de donnée.

2.4.2 Contrainte

Il faut insérer au préalable des données dans la base.

	Valide	Invalide
void deletePerson(Person p);	Une personne	Autre qu'une personne

2.4.3 Dépendances

Nous devons afficher la personne supprimée pour vérifier qu'elle n'existe plus dans la base de donnée personne avec la méthode findPerson

2.4.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void deletePerson()	Une personne	La personne p est supprimé de la BDD	La personne p existe et n'existe plus après l'exécution de la méthode
Public void deletePersonDontExistTest()	Une personne	Exécute la requête sans modification de la BDD	La personne que l'on veut supprimer n'existe pas dans la BDD

2.5 Méthodes UpdatePerson

2.5.1 Description

Test de la fonction updatePerson qui permet de mettre à jour les informations d'une personne.

2.5.2 Contrainte

Il faut insérer au préalable des données dans la base.

	Valide	Invalide
void updatePerson(Person p);	Une personne	Autre q'une personne

2.5.3 Dépendances

Nous devons sauvegarder des personnes donc la fonction savePerson doit être testée auparavant, idem en ce qui concerne l'affichage d'une personne avec la méthode findPerson

2.5.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void updatePerson()	Une personne	Mise à jour des informations de la personne.	Met à jour les informations d'une personne
Public void updatePersonAlreadyExist()	Une personne	Renvoie une erreur MySQLIntegrityConstraintViolationException on change l'id par un id existant	Les personne p et p' existent dans la BDD avec des id différents au préalable
Public void updatePersonDontExist()	Une personne	Exécute la requête sans modification de la BDD	La personne p n'existe pas dans la BDD

3 Test unitaires "Groupe"

3.1 Méthodes FindAllGroup

3.1.1 Description

Tester si la fonction renvoie tous les groupes se trouvant dans la base de donnée.
Création d'une collection contenant les groupes pour la comparer aux resultats retournés par la fonction.

3.1.2 Contrainte

Il faut insérer au préalable des données dans la base.

	Valide	Invalide
Collection<Group> findAllGroups();	-----	-----

3.1.3 Dépendances

Nous devons sauvegarder des personnes donc la fonction saveGroup doit être testée auparavant

3.1.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public void findAllGroupTest()	-----	Tous les groupes se trouvant dans la BDD	Les groupes se trouvent dans la BDD

3.2 Méthodes FindGroup

3.2.1 Description

Tester si la fonction renvoie le group g.
Création d'une personne p' pour la comparer aux resultats retourner par la fonction.

3.2.2 Contrainte

Il faut insérer au préalable des données dans la base.

	Valide	Invalide
Group findGroup(**long** id);	Entier positif	Différent d'un entier positif

3.2.3 Dépendances

Nous devons sauvegarder des personnes donc la fonction savePerson doit être testée auparavant

3.2.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void findGroupTest()	Entier positif représentant l'identifiant du groupe	Donne le groupe ayant pour identifiant celui du paramètre	Retourne le groupe ayant comme identifiant l'id passé en paramètre
Public void findGroupDontExistTest( )	Entier positif représentant l'identifiant du groupe	Aucun groupe retourner (null)	Le groupe n'existe pas dans la BDD

3.3 Méthodes SaveGroup

3.3.1 Description

Tester si la fonction insère un groupe dans la base de donnée.

3.3.2 Contrainte

Entrer les informations du groupe (par exemple le nom du groupe) dans un format défini.

	Valide	Invalide
void saveGroup(Group g);	Un groupe	Autre qu'un groupe

3.3.3 Dépendances

3.3.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public void saveGroupTest()	Un groupe	Insère une groupe dans la BDD.	Ajout un nouveau group dans la BDD
public void saveGroupAlreadyExistTest	Un groupe	Renvoie une MySQLIntegrityConstraintViolationException si l'id du groupe existe dejà	Le groupe qu'on veut insère possède un id existant

3.4 Méthodes DeleteGoup

3.4.1 Description

Test si la fonction deleteGroup supprime un groupe g de la base de donnée.

3.4.2 Contrainte

Il faut insérer au préalable des données dans la base.

	Valide	Invalide
void deleteGroup(Group g);	Un groupe	Autre qu'un groupe

3.4.3 Dépendances

Aucune dépendance requise.

3.4.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
Public void deleteGroupTest()	Un groupe	Suppression de la personne de la BDD	Le groupe g existe et n'existe plus après l'exécution de la méthode
Public void deleteGroupDontExistTest()	Un groupe	Exécute la requête sans modification de la BDD	Le groupe que l'on veut supprimer n'existe pas dans la BDD

3.5 Méthodes UpdateGroup

3.5.1 Description

Test de la fonction updateGroup qui permet de mettre à jour les informations d'un groupe.

3.5.2 Contrainte

Il faut insérer au préalable des données dans la base.

	Valide	Invalide
void updateGroup(Group g);	Un groupe	Autre qu'un groupe

3.5.3 Dépendances

Aucune dépendance requise.

3.5.4 Procédure de test

	Données en entrée	Résultats attendus	Critère de validation
public void updateGroupTest()	Une personne	Mise à jour des informations deu groupe	Met à jour les informations d'un groupe
Public void updateGroupAlreadyExist()	Une personne	Renvoie une erreur MySQLIntegrityConstraintViolationException on change l'id par un id existant	Les groupes g et g' existent dans la BDD avec des id différents
Public void updateGroupDontExist()	Une personne	Exécute la requête sans modification de la BDD	Le groupe g n'existe pas dans la BDD