Rapport de tests

21/03/2012

Anthony Morales, Chen-Yang Gao, Irvin genieys, Simon Maby

Table des matières

I Tests unitaire

Nous avons choisi de réaliser des tests unitaires pour tester le retour de certaines méthodes, pour cela nous avons utilisé le framework JUnit qui permet de réaliser des tests unitaires.

A. Test de connexion

Nous avons réalisé un fichier de tests pour vérifier le type de retour de la fonction connexion, en effet cette fonction doit renvoyer une valeur différente selon si l'enseignant qui se connecte est un responsable ou pas.

TestLogin.java :

```
public class TestLogin {

	public void testLoginEnseignant() {

		AbstractFactory.setInstance(new FactoryBD());

		Facade f = new Facade();

		int res = f.connexion("Sala", "michel");

		assertTrue(res == 1);
	}

public void testLoginResponsable() {

		AbstractFactory.setInstance(new FactoryBD());

		Facade f = new Facade();

		int res = f.connexion("Stratulat", "tiberiu");

		assertTrue(res == 2);
	}

}
```

B. Test de chargement des créneaux

Nous avons aussi choisi de tester le bon déroulement du chargement des créneaux. Pour cela nous avons créer le fichier de test suivant.

TestCreneaux.java :

```
public class TestCreneaux {

public void testChargementCreneaux() {

		AbstractFactory.setInstance(new FactoryBD());

		Facade f = new Facade();

		int res = f.connexion("Sala", "michel");

		ArrayList<Creneau> creneaux = f.getCreneaux();
		int nbCreneaux = -1;
		ResultSet resRequete;

		resRequete = GestionnaireBD.getInstance().executeRequete("select
count(*) from crenau");

		try {
			while(resRequete.next()) {
				nbCreneaux = Integer.parseInt(resRequete.getString(1));

			}
		} catch (NumberFormatException e) {

			e.printStackTrace();
		} catch (SQLException e) {

			e.printStackTrace();
		}

		assertTrue(nbCreneaux == creneaux.size());
	}

}
```

C. Test du chargement des caractéristiques

Nous avons enfin décidé de vérifier le chargement des caractéristiques.

TestCaracteristiques.java :

```
public class TestCaracteristiques {

public void testChargementCaracteristiques() {

		AbstractFactory.setInstance(new FactoryBD());

		Facade f = new Facade();

		int res = f.connexion("Sala", "michel");

		ArrayList<Caracteristique> caracteristiques =
f.getCaracteristiques();
		int nbCaracteristiques = -1;
		ResultSet resRequete;

		resRequete = GestionnaireBD.getInstance().executeRequete("select
count(*) from caracteristique");

		try {
			while(resRequete.next()) {
				nbCaracteristiques =
Integer.parseInt(resRequete.getString(1));

			}
		} catch (NumberFormatException e) {

			e.printStackTrace();
		} catch (SQLException e) {

			e.printStackTrace();
		}

		assertTrue(nbCaracteristiques == caracteristiques.size());
	}
}
```

II Tests fonctionnels

Tout au long du développement de notre projet nous avons réalisé des tests pour vérifier le bon fonctionnement de notre application. En effet comme cela est décrit dans le rapport de conception, nous avons fait en sorte que notre business logic soit la plus indépendante possible de l'interface utilisateur. Pour cela nous avons effectué certains contrôles dans les classes métiers. Nous avons donc vérifié que ces contrôles s'opéraient bien en simulant par des tests une interface utilisateur qui fournit des données incohérentes. Ces tests se trouvent dans les fichiers MainTest.java et Test_tmp.java.