

Exercice : Portes Conteneurs

# Contexte



Le transport maritime mondial passe aujourd’hui par les [conteneurs](https://fr.wikipedia.org/wiki/Conteneur), grosses boite d’acier de 6 à 12m de long pour le modèle standard empilé sur des [navires](https://fr.wikipedia.org/wiki/Porte-conteneurs) qui desservent tout les ports du monde. La plus longue ligne maritime relie Boston à San-Francisco en passant par la Méditerranée, le canal de Suez, Singapour, Taïwan et le Japon en moins de 15 semaines.

L’organisation du transport maritime se résume à associer des conteneurs aux navires qui passent par les ports de destination et à empiler les conteneur dans le bon ordre. Rien de tel qu’une base de données relationnelle pour gérer tout ça !

# Enoncé

L'objectif de cet exercice est d'implémenter une librairie d’accès à une base de données relationnelle sous la forme de tests unitaires. Autrement dit : l'objectif de l'exercice est de faire passer en vert un jeu de tests.

Vous disposez pour cela d'une solution comprenant 2 projets :

* ITI.DataAccesLibrary.Tests, contient les tests unitaires
* ITI. DataAccesLibrary, une implémentation minimale du modèle (le code écrit ne permet rien de plus que de pouvoir compiler sans erreur)

Pour faire tourner les tests unitaires il vous suffit d’utiliser le *Test Explorer*.

Comme vous pouvez le constater, pour le moment, tous les tests sont rouges.

TODO CHANGE PICTURE

Pour les autres tests, à vous de faire en sorte qu'ils passent en vert. Pour cela, vous avez le droit de faire ce que bon vous semble dans le projet ITI.DataAccesLibrary (modifier l'implémentation des méthodes, des propriétés ou des constructeurs, ajouter des membres *internal* ou *private*, créer de nouveaux types etc...) En revanche, vous ne devez pas modifier le projet ITI.DataAccesLibrary.Tests.

-expliquer comment fonctionne la DAL et l'abstraction de Queries et des champs à disposition

-Préciser que toutes les requêtes sont réalisable en sql sans opération supplémentaire

-Dire qu'il est possible de charger tous en mémoire puis de la trier ou de juste écrire la requete sql

## Structure de la base

Pour faciliter l’embarquement d’une base de données, le choix s’est porté sur SQLite, une version allégé et portable de SQL. La base modélise 3 table liées entre elles. A chaque exécution des tests elle est regénéré de manière aléatoire et cohérente. Inutile donc de se baser sur son contenu. (Qui reste néanmoins accessible via une invite de commande)

Les ports (Harbor) sont modélisés simplement avec un id, un nom, un pays et une position (Latitude/Longitude)

Les navires (ContainerShip) sont modélisés par un id, un code ATIS (Guid), le nom, le port d’origine et de destination (relation un à plusieurs depuis les ports), la date de départ et d’arrivée (Au format supporté par SQLlite), du nombre de membre d’équipage, de la charge (en Tonnes) et la vitesse maximale du navire (en nœuds nautiques). Enfin 3 valeurs définissent le nombre de conteneur à bord. Par convention la largeur (width) comme axe X, la hauteur (height) comme axe Y et la longueur (length) comme axe Z.



Les conteneurs (Container) sont modélisés par un id, un code de référence (Guid), un contenu, le navire qui les transporte (relation un à plusieurs depuis les navires), l’origine et la destination (relation un à plusieurs depuis les ports), si le conteneur à un toit ouvert (il est donc forcément au sommet des piles), sa masse à vide, sa charge maximale et enfin sa position dans le navire (cf : convention plus haut).



## Fonctionnalités attendues

Les tests unitaires sont là pour spécifier de façon détaillée les fonctionnalités attendues. Cependant voici une courte description de ce qui est attendu :

* On peut créer une école. Les écoles sont indépendantes les unes des autres.
* On peut créer des élèves (Pupil) dans une classe.
  + Les élèves ont un nom et un prénom qui ne changent jamais.
  + On doit pouvoir retrouver un élève par son nom et son prénom
* On doit pouvoir créer/supprimer des classes (ClassRoom).
  + Chaque classe à un nom unique.
  + On doit retrouver une classe par son nom.
  + On doit pouvoir changer le nom d’une classe (uniquement si le nouveau nom ne rentre pas en conflit avec le nom d’une autre classe).
* On doit pouvoir créer des professeurs (Teacher).
  + Un professeur a un nom qui ne change pas.
* Un professeur s’occupe de 0 ou 1 classe et une classe ne peut être gérée que par un professeur au maximum : on dit qu’un professeur est assigné à une classe (Assignment).

Il y a 13 tests à passer au vert :

SCREEN TESTS

# Ressources

<https://www.sqlite.org>

<https://system.data.sqlite.org/index.html/doc/trunk/www/index.wiki>

Au boulot !