Génie logiciel avancé - 2016
TD n°7
Système de planification des vols

Les objectifs de ce TD sont :

- De poursuivre l'implémentation du projet
- De réaliser des tests sur vos différents composants, et vos différentes chaines fonctionnelles.

A partir des problématiques que vous avez rencontré durant l'implémentation de chaque composant, vous devez développer un ou plusieurs tests permettant de contrôler le bon fonctionnement de vos composants.

La méthodologie générale est :

1. Identifier un ou plusieurs problèmes récurrents liés à votre composant ;
2. Réfléchir à une stratégie pour tester la présence d'un tel problème le plus automatiquement possible ;
3. Implémenter vos tests ;
4. Vérifier leur bon fonctionnement !

Afin de vous guider dans la réalisation de vos tests, voici quelques exemples liés à chaque composant.

Question 1 – Tests de la couche de persistance

Concernant la couche de persistance, une problématique récurrente est le statut des objets en sortie de votre API. En particulier, si les objets ne sont pas détachés correctement, cela peut amener à des exceptions ultérieures du type « JDODetachedFieldAccessException ».

Pour éviter cela, chacune de vos méthodes du DAO doit appeler la fonction « detachCopy » du gestionnaire de persistance sur les éléments récupérés depuis la base de données.

Une stratégie de test peut par exemple consister, via le système de réflexion en Java, à trouver automatiquement toutes les méthodes débutant par « get » et les appeler de manière automatique.

Pour chaque valeur de retour, il suffira de tester son statut via la méthode « isDetached ».

Vous devez développer un tel test. Voici quelques liens utiles pour cela :

- JDOHelper : http://db.apache.org/jdo/jdohelper.html
- DataNucleus – Cycle de vie des objets : http://www.datanucleus.org/products/datanucleus/jdo/object_lifecycle.html

Question 2 – Tests des web-services

Dans certains cas, vos web-services renvoient un code d'erreur « 500 – Internal server error ». Il peut avoir plusieurs raisons : problème d'implémentation, paramètre de la requête invalides, etc.

Dans tous les cas, cela doit être évité :

- S'il existe un problème d'implémentation, il doit être corrigé ;
- Si un ou plusieurs paramètres de la requête sont invalides, un code retour de la famille 400 est plus approprié.

Pour vérifier le bon comportement de votre serveur, vous pouvez appeler de manière automatique l'ensemble de vos web-services et vérifier que le code de retour 500 n'est jamais renvoyé.

L'implémentation peut être faite via l'extension « Client » de Jersey. La dépendance Maven vous est donnée ci-dessous :

```xml
<dependency>
        <groupId>org.glassfish.jersey.core</groupId>
        <artifactId>jersey-client</artifactId>
        <version>2.3</version>
</dependency>
```

Voici un exemple d'appel via cet API :

```java
Client client = ClientBuilder.newClient(new ClientConfig().register(JacksonFeature.class));
WebTarget target = client.target("webserviceUrl");
Response resp = target.request().get();
System.out.println(resp.getStatus());
```

Vous devez implémenter un tel test.

Question 3 – Test de l'interface graphique

A la différence des autres composants, vous pouvez tester l'ensemble d'une fonction utilisateur via l'interface graphique.

Tel que vu en cours, et en vous appuyant sur le framework « Selenium », vous devez implémenter le test suivant :

- Authentification dans l'interface CCO,
- Ajout d'un vol,
- Ajout d'un équipage sur ce vol
- Déconnexion de l'interface CCO,
- Authentification dans l'interface équipage,
- Visualisation de ce vol.

Une fois réalisé, vous pouvez l'exporter vers JUnit pour qu'il soit lancer automatiquement dans Jenkins.