Partie 2 – Gestion des relations, services et transactions

Nous allons étudier les différentes étapes du développement d'une application Spring Boot. Ces étapes étant dépendantes, il est indispensable que chaque partie soit entièrement réalisée avant de passer à la suivante.

1 – Gestion de relation de type to-one et to many entre Project et Enterprise

Chaque projet est proposé par une entreprise qui doit être spécifiée à la création d'un projet.

1. Modifiez le contenu du fichier ProjectTest.java pour qu'il soit identique au contenu disponible ici : https://gist.github.com/FranckSilvestre/70ca9d583d59cae9411522346c547482
2. Lancez les tests de la classe ProjectTest et constatez que certains tests ne passent plus.
3. Modifiez le contenu de la classe Project pour que les tests passent de nouveau.
   // GIT \\ fix #2.1.3 relation Project Enterprise
4. Modifiez le contenu du fichier EnterpriseTest.java pour qu'il soit identique au contenu disponible ici : https://gist.github.com/FranckSilvestre/ada676544b751a081f8e4f5f67a29e70
5. Lancez les tests de la classe EnterpriseTest et constatez que certains tests ne passent plus.
6. Modifiez le contenu de la classe Enterprise pour expliciter la relation one-to-many de Enterprise vers Project de telle sorte que les tests passent de nouveau.
   // GIT \\ fix #2.1.6 relation Enterprise Project
7. Modifiez le contenu du fichier EnterpriseProjectServiceIntegrationTest.java pour qu'il soit identique au contenu disponible ici : https://gist.github.com/FranckSilvestre/d6c39fa70b5ad64c6cf98022fc103193
8. Lancez les tests et constatez que certains tests ne passent plus.
9. Modifiez le contenu des classes Project, Enterprise et EnterpriseProjectService de telle sorte que les tests de la classe EnterpriseProjectServiceIntegrationTest passent de nouveau.
   // GIT \\ fix #2.1.9 service improvement
10. Lancez l'ensemble des tests et vérifiez que vous n'obtenez pas de régression. En cas de régression observée, modifiez la ou les classes nécessaires pour que l'ensemble des tests passent de nouveau.
    // GIT \\ fix #2.1.10 non regression OK

La suite de cette partie vise la mise en place un Web service exposant la liste des projets disponibles avec leur entreprise.

2 – Mise en place d'une méthode de service récupérant la liste des projets

Le web service doit s'appuyer sur le service EnterpriseProjectService ; ce dernier doit donc proposer une méthode permettant de récupérer la liste des projets stockées en base.

1. Modifiez le contenu du fichier EnterpriseProjectServiceIntegrationTest pour qu'il soit identique au fichier disponible ici : https://gist.github.com/FranckSilvestre/db3cae57740c9f2c011b1206b750a4cd
2. Lancez les tests et constatez que certains tests ne passent plus.
3. Effectuez les modification nécessaires dans votre projet pour que les tests passent de nouveau.
   // GIT \\ fix #2.2.3 fecthing projects with enterprises OK

Afin de disposer de quelques projets au lancement de l'application nous allons mettre en place la création de quelques entreprises et projets au démarrage de l'application.

4. Créez la classe Bootstrap et sa classe de test BoostrapTest associée. La classe Bootstrap permettra le déclenchement de la créations de projets au lancement de l'application en s'appuyant sur un service d'initialisation incarné par la classe InitializationService. Créez la classe InitializationService.
5. Modifiez le contenu des fichiers BootstrapTest.java et EnterpriseProjectServiceIntegrationTest.java pour qu'ils soient identiques aux fichiers disponibles ici :
   https://gist.github.com/FranckSilvestre/abf597bf3248674bd536ad7e85ef4eb2
   https://gist.github.com/FranckSilvestre/39bccd27aa7b198268ccf5d90134d346
6. Lancez les tests et constatez que certains tests ne passent plus.
7. Modifiez votre projet de telle sorte que les tests passent de nouveau.
   Indications : pour faire passer les tests vous aurez besoin des annotations « *javax.annotation.PostConstruct* » et « *javax.transaction.Transactional* ».
   // GIT \\ fix #2.2.7 initialisation application OK

3 - Création du RESTful contrôleur

1. Créez la classe ProjectController. Annotez la classe de l'annotation *org.springframework.web.bind.annotation.RestController.*
   Créez les deux classes de tests ProjectControllerTest et ProjectControllerIntegrationTest. Modifiez les contenus de ces deux classes pour qu'ils correspondent aux contenus disponibles en ligne :
   https://gist.github.com/FranckSilvestre/c6e0e7b23d85ca9633a89ec14c262d23
   https://gist.github.com/FranckSilvestre/3c9e08529970e1bfb8de5e399f5fbbb5
2. Lancez les tests et constatez que certains tests ne passent plus.
3. Modifiez votre projet de telle sorte que les tests passent de nouveau.
   Indications : pour faire passer les tests vous aurez besoin des annotations « *org.springframework.web.bind.annotation.RequestMapping* » et « *com.fasterxml.jackson.annotation.JsonIgnore* ».
   // GIT \\ fix #2.3.3 Rest Controller OK
4. Lancez l'application et rendez vous sur l'URL suivante :
   http://localhost:8080/projectsWithEnterprises

Vérifiez que le contenu Json obtenu présente bien les 3 projets et leur entreprise associée.

4 - Focus sur les transactions

Nous allons étudier l'effet de l'annotation « *@Transactional* » sur l'exécution du flot de requêtes permettant l'initialisation des projets au lancement de l'application.

1. Modifiez le fichier InitializationService.java de telle sorte que l'insertion du dernier projet échoue lors de l'appel de la méthode initProjects(). Par exemple, affectez au projet un titre vide ou « null ».
   Lancez (ou relancez) le test « *testFindAllProjectsFromInitialization* » de la classe EnterpriseProjectServiceIntegrationTest et vérifiez qu'aucun projet n'apparait dans la liste des projets au lieu des 3 attendus.
2. Après avoir remis la classe «InitialisationService» dans son état faisant passer les tests, expliquez sous forme d'un commentaire à l'intérieur de la méthode « initProjects » le phénomène observé.
   // GIT \\ fix #2.4.2 test erreur avec transaction

5 - Réflexions sur le SQL généré par Hibernate

Nous allons à présent observer le Sql généré par Hibernate lors de la récupération des projets ..

1. Dans le fichier application.properties (dans le dossier src/main/resources), ajoutez la ligne suivante :
   logging.level.org.hibernate.SQL=DEBUG
2. Relancez l'application et observez le contenu de la console lorsque vous accéder à l'URL http://localhost:8080/projectsWithEnterprises. La console affiche maintenant le Sql généré par Hibernate pour chaque requête.
   Vous pouvez faire un « clear » de la console à période régulière afin de pouvoir observer plus facilement le Sql lors du rechargement de la page.
   Combien de requêtes sont générées lors de l'affichage de tous les projets ? Des requêtes vous paraissent-elles « inutiles » ? Pouvez vous expliquer leur présence.
   // GIT \\ fix #2.5.2 generated SQL
3. Modifiez l'application de telle sorte qu'une seule requête soit exécutée lors de l'accès à l'URL http://localhost:8080/projectsWithEnterprises.
   // GIT \\ fix #2.5.3 optimization SQL