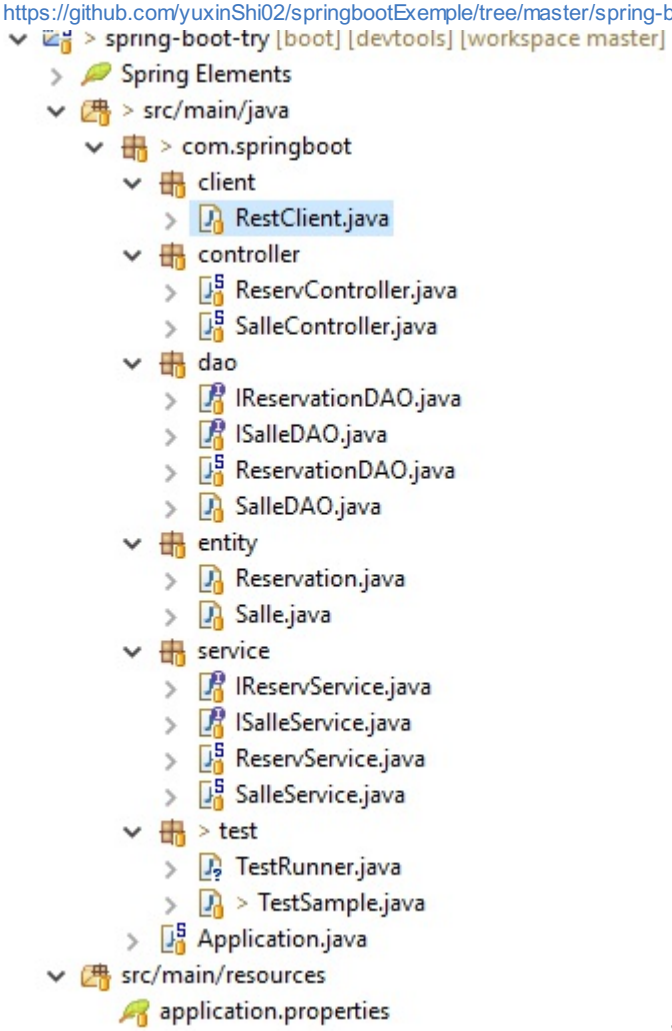
Editor: Yuxin SHI
SI 2 FISE

Compte rendue : Outils pour cycle de vie Logiciel

TP3

1. Implémenter une deuxième fonctionnalité de votre choix pour l'utilisateur enseignant, en se basant sur le principe de multi-module vue dans le cours (vous êtes libre dans le choix de l'implémentation de cette fonctionnalité, vous pouvez utiliser ce que vous voulez comme librairie). Le projet est sur github: https://github.com/yuxinShi02/springbootExemple/tree/master/spring-boot-try Ceci la structure du projet:

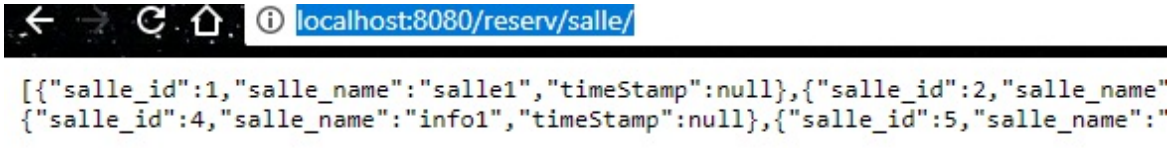


J'ai implementé une nouvelle fonctionnalité : consulter les salles, ainsi sa disponibilité. Par exemple:

- Si on veut savoir que la salle id 1 est disponible ou pas à 2017/10/24: 2017/10/24:
  http://localhost:8080/reserv/salle/1/2017-10-24
  Le résultat:

no

- Si on veut consulter tous les salles:
  http://localhost:8080/reserv/salle/
  Le résultat:



2. Centraliser les informations, factoriser les dépendances de votre projet (utiliser le principe des propriétés).
   `jai pas compris`
3. Configurer maven afin de pouvoir utiliser ce référentiel central : http://repo.maven.apache.org/maven2/.
   Dans le fichier config de maven: `setting.xml`, on change les parametres:

```
<repositories>
     <repository>
       <id>central</id>
       <url>https://repo.maven.org/maven2</url>
       <releases>
         <enabled>true</enabled>
       </releases>
     </repository>
   </repositories>
   <pluginRepositories>
     <pluginRepository>
       <id>central</id>
       <url>https://repo.maven.org/maven2</url>
       <releases>
         <enabled>true</enabled>
       </releases>
     </pluginRepository>
   </pluginRepositories>
```

4. Installer Archiva sur votre machine (idéalement sur une VM accessible en réseau depuis votre machine), configurer maven pour utiliser ce référentiel distant.
   `je n'ai pas compris`

5. Implémenter des tests unitaires couvrant certaines méthodes proposées par l'API JUnit et Jmockit.
   Dans mon projet, j'ai ajouté un package: test, ainsi les deux classes qui permet d'effectuer les tests unitaires.
   Test de reservationById:

```
/**
 * Compare the result of web and the result of database
 * if they are equal, test is true
 */
@Test
public void testReservById(){
  System.out.println("test ReservByID");
  RestClient restClient = new RestClient();
  String stringByWeb = restClient.getByIdReserv(1);

  ReservationDAO reservationDAO = new ReservationDAO();
  Reservation rsvByDao = reservationDAO.getById(1);
  assertEquals(stringByWeb, rsvByDao.toString());
}
```

Ceci testrunner:

```
public class TestRunner {
     public static void main(String[] args) {
          Result result = JUnitCore.runClasses(TestSample.class);
          for (Failure failure : result.getFailures()) {
             System.out.println(failure.toString());
          }
          System.out.println(result.wasSuccessful());
```

```
        }
}
```

6. Exécuter en mode commande les tests unitaires et vérifier que ça passe sans échec.
   Malheusement:

```
<terminated> TestRunner [Java Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (5 nov. 2017 à 21:08:06)
Exception in thread "main" java.lang.NoClassDefFoundError: org/junit/runner/JUnitCore
        at com.springboot.test.TestRunner.main(TestRunner.java:9)
Caused by: java.lang.ClassNotFoundException: org.junit.runner.JUnitCore
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        ... 1 more
```

   J'ai bien ajouté jUnit.jar dans maven projet, et le jar est bien dans le build path mais il existe tourjours ce probleme, je n'ai pas pu résodre ce probleme.