



# myerp

Mise en place d'un  
environnement de testing

Open Class Rooms : Projet 9

## I. Éléments de tests mis en place

- Tests unitaires
- Tests d'intégration

## II. Stratégie de test d'intégration

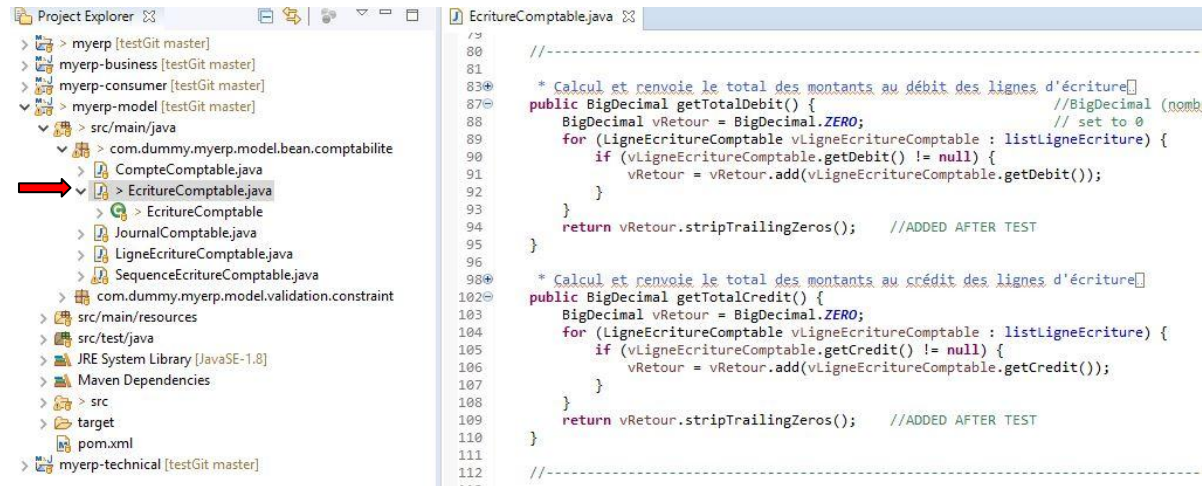


# Tests unitaires

Définition : Procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme.

- Mais quelles portions de code dans le projet ?

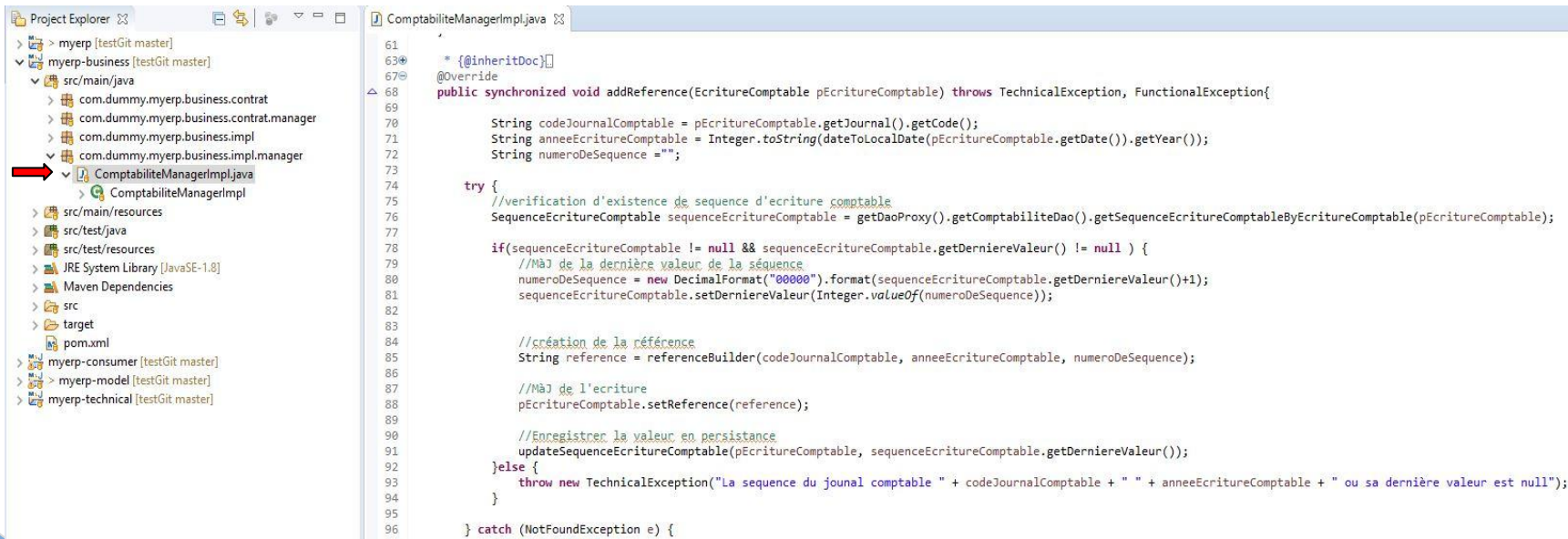
Model layer : EcritureComptable.class



```
//-----  
80  
81  
82  
83 * Calcul et renvoie le total des montants au débit des lignes d'écriture[]  
84  
85 public BigDecimal getTotalDebit() { //BigDecimal (nom)  
86     BigDecimal vRetour = BigDecimal.ZERO; // set to 0  
87     for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {  
88         if (vLigneEcritureComptable.getDebit() != null) {  
89             vRetour = vRetour.add(vLigneEcritureComptable.getDebit());  
90         }  
91     }  
92  
93     return vRetour.stripTrailingZeros(); //ADDED AFTER TEST  
94 }  
95  
96  
97  
98 * Calcul et renvoie le total des montants au crédit des lignes d'écriture[]  
99  
100 public BigDecimal getTotalCredit() {  
101     BigDecimal vRetour = BigDecimal.ZERO;  
102     for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {  
103         if (vLigneEcritureComptable.getCredit() != null) {  
104             vRetour = vRetour.add(vLigneEcritureComptable.getCredit());  
105         }  
106     }  
107  
108     return vRetour.stripTrailingZeros(); //ADDED AFTER TEST  
109 }  
110  
111  
112  
//-----
```

# Tests unitaires

## Business layer : ComptabiliteManagerImpl.class



```
Project Explorer
> myerp [testGit master]
  > myerp-business [testGit master]
    > src/main/java
      > com.dummy.myerp.business.contrat
      > com.dummy.myerp.business.contrat.manager
      > com.dummy.myerp.business.impl
      > com.dummy.myerp.business.impl.manager
      > ComptabiliteManagerImpl.java
      > ComptabiliteManagerImpl
    > src/main/resources
    > src/test/java
    > src/test/resources
    > JRE System Library [JavaSE-1.8]
    > Maven Dependencies
    > src
    > target
    > pom.xml
  > myerp-consumer [testGit master]
  > myerp-model [testGit master]
  > myerp-technical [testGit master]
```

```
ComptabiliteManagerImpl.java
61
62
63 * {@inheritDoc}
64
65 @Override
66
67 public synchronized void addReference(EcritureComptable pEcritureComptable) throws TechnicalException, FunctionalException{
68
69
70     String codeJournalComptable = pEcritureComptable.getJournal().getCode();
71     String anneeEcritureComptable = Integer.toString(dateToLocalDate(pEcritureComptable.getDate()).getYear());
72     String numeroDeSequence = "";
73
74
75     try {
76         //verification d'existence de sequence d'écriture comptable
77         SequenceEcritureComptable sequenceEcritureComptable = getDaoProxy().getComptabiliteDao().getSequenceEcritureComptableByEcritureComptable(pEcritureComptable);
78
79         if(sequenceEcritureComptable != null && sequenceEcritureComptable.getDerniereValeur() != null ) {
80             //MàJ de la dernière valeur de la séquence
81             numeroDeSequence = new DecimalFormat("00000").format(sequenceEcritureComptable.getDerniereValeur()+1);
82             sequenceEcritureComptable.setDerniereValeur(Integer.valueOf(numeroDeSequence));
83
84
85             //création de la référence
86             String reference = referenceBuilder(codeJournalComptable, anneeEcritureComptable, numeroDeSequence);
87
88             //MàJ de l'écriture
89             pEcritureComptable.setReference(reference);
90
91             //Enregistrer la valeur en persistance
92             updateSequenceEcritureComptable(pEcritureComptable, sequenceEcritureComptable.getDerniereValeur());
93         }else {
94             throw new TechnicalException("La sequence du journal comptable " + codeJournalComptable + " " + anneeEcritureComptable + " ou sa dernière valeur est null");
95         }
96     } catch (NotFoundException e) {
```

# Tests unitaires

- Comment tester en isolation des autres éléments de l'application ?
  - Mockito : Mock (simule) les éléments nécessaires au fonctionnement de l'élément testé.

```
ComptabiliteManagerImplUnitTest.java x
1 package com.dummy.myerp.business.impl.manager;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32 @RunWith(MockitoJUnitRunner.class)
33 public class ComptabiliteManagerImplUnitTest {
34
35
36     @InjectMocks
37     private ComptabiliteManagerImpl manager;
38     @InjectMocks
39     private DaoProxyImpl daoProxy;
40     @Mock
41     private ComptabiliteDaoImpl comptabiliteDaoImpl;
42     @Mock
43     private SequenceEcritureComptable sec;
44     @Mock
45     private BusinessProxyImpl businessProxyImpl;
46     @Mock
47     private TransactionManager trM;
48
49     private static SimpleDateFormat sdf;
50     private static Date date;
51     private EcritureComptable vEcritureComptable;
52     private String expectedRef;
53     private Integer derniereValeurSequence;
54 }
```

# Tests unitaires

- Mockito : donne la possibilité de faire renvoyer aux éléments “mockés” des résultats prédéfinis par l'utilisateur. Celui-ci pourra ainsi vérifier qu'avec des paramètres connus en entrée, l'élément testé renvoie le résultat attendu en sortie.

```
ComptabiliteManagerImplUnitTest.java
78
79 // ----- METHODE addReference -----
80 //Methode addReference Test : si la référence de l'écriture comptable suit la séquence du journal comptable auquel cette dernière appartient
81 @Test
82 public void addReference() throws Exception {
83     derniereValeurSequence = 2;
84     expectedRef = "AC-2019/00003";
85
86     //Mock the consumer layer
87     {
88         when(sec.getDerniereValeur()).thenReturn(derniereValeurSequence);
89         when(daoProxy.getComptabiliteDao().getSequenceEcritureComptableByEcritureComptable(vEcritureComptable)).thenReturn(sec);
90     }
91     //method to test
92     manager.addReference(vEcritureComptable);
93     assertEquals(expectedRef, vEcritureComptable.getReference());
94 }
```



# Tests unitaires

- JUnit : Permet l'exécution des tests. Met à disposition de l'utilisateur un ensemble d'outils permettant de vérifier que le résultat obtenu correspond au résultat attendu.

```
ComptabiliteManagerImplUnitTest.java
78
79 // ----- METHODE addReference -----
80 //Methode addReference Test : si la référence de l'écriture comptable suit la séquence du journal comptable auquel cette dernière appartient
81 @Test
82 public void addReference() throws Exception {
83     derniereValeurSequence = 2;
84     expectedRef = "AC-2019/00003";
85
86     //Mock the consumer layer
87     when(sec.getDerniereValeur()).thenReturn(derniereValeurSequence);
88     when(daoProxy.getComptabiliteDao().getSequenceEcritureComptableByEcritureComptable(vEcritureComptable)).thenReturn(sec);
89
90     //method to test
91     manager.addReference(vEcritureComptable);
92
93     { assertEquals(expectedRef, vEcritureComptable.getReference()); }
94 }
```

# Tests unitaires

- JUnit: Tandis que `@BeforeClass` permet l'exécution de code avant le début de l'exécution des méthodes de tests, `@Before` permet l'exécution de code avant l'exécution de chaque méthode.

```
{ @BeforeClass }  
public static void executeBeforeAll() throws ParseException, FunctionalException {  
    sdf = new SimpleDateFormat("dd/MM/yyyy");  
    date = sdf.parse("21/12/2019");  
}  
  
{ @Before }  
public void executeBeforeEach() {  
    //config with MOCK  
    AbstractBusinessManager.configure(businessProxyImpl, daoProxy, trM);  
  
    //STUB  
    vEcritureComptable = new EcritureComptable();  
    vEcritureComptable.setDate(date);  
    vEcritureComptable.setJournal(new JournalComptable("AC", "Achat"));  
    vEcritureComptable.setLibelle("Libelle");  
    vEcritureComptable.setReference("AC-2019/00004");  
    vEcritureComptable.getListLigneEcriture().add(new LigneEcritureComptable(new CompteComptable(1), null, new BigDecimal(123), null));  
    vEcritureComptable.getListLigneEcriture().add(new LigneEcritureComptable(new CompteComptable(2), null, null, new BigDecimal(123)));  
}
```



# Tests unitaires

→ JUnit : Le résultat attendu peut-être une exception.

ComptabiliteManagerImplUnitTest.java

```
121 //Methode addReference Test : si l'écriture comptable appartient à un journal comptable qui possède une séquence mais dont la VALEUR est null, technical exception
122 { @Test(expected = TechnicalException.class) }
123 public void addReferenceSequenceJournalComptableExistButLastValueIsNull() throws Exception {
124     derniereValeurSequence = null;
125
126     //Mock the consumer layer : sequence found but last value is null
127     when(sec.getDerniereValeur()).thenReturn(derniereValeurSequence);
128     when(daoProxy.getComptabiliteDao().getSequenceEcritureComptableByEcritureComptable(vEcritureComptable)).thenReturn(sec);
129
130     //method to test
131     manager.addReference(vEcritureComptable);
132 }
133
```

# Tests d'intégration

Définition : Procédure permettant de vérifier que l'élément testé s'intègre bien parmi les autres éléments constituant l'application.

- Quels éléments dans le projet ?
  - Consumer layer : Tests portés sur les classes dao, vérification du bon fonctionnement du processus et des résultats des requêtes envoyées à la base de données.
  - Business layer :
    - Test de l'initialisation du context Spring
    - Tests du bon fonctionnement du processus et vérification de la conformité des résultats des méthodes des classes de la couche.

# Tests d'intégration

- Les *profiles* Maven :

- Profile test-consumer : avant la phase de lancement des tests

- Initialisation du context Spring de la couche consumer.
- Création des tests d'intégration de la couche consumer ( via build-helper-maven-plugin).

Sont lancés pendant la phase test :

- Les tests unitaires
- Les tests d'intégration de la couche consumer

- Profile test-business : avant la phase de lancement des tests

- Initialisation du context Spring de la couche business et consumer.
- Création des tests d'intégration de la couche business ( via build-helper-maven-plugin).

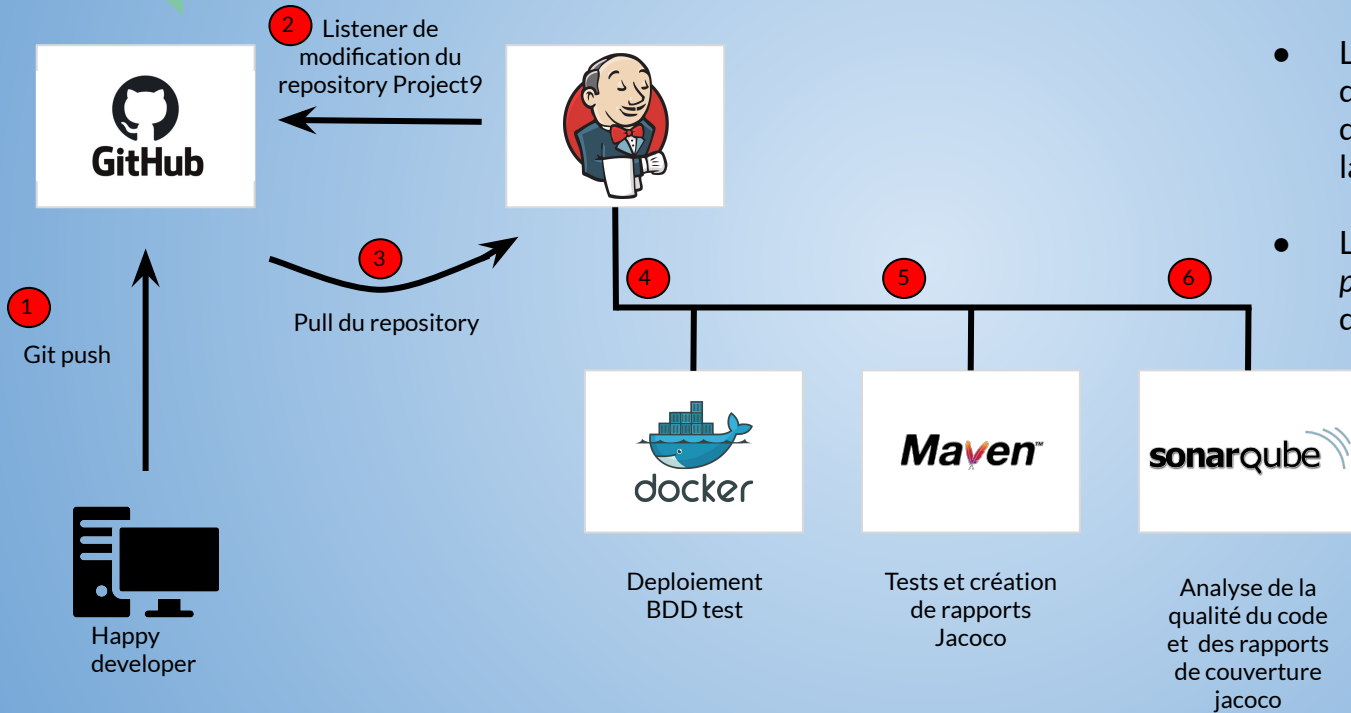
Sont lancés pendant la phase test :

- Les tests unitaires
- Les tests d'intégration de la couche business

## II. Stratégie de test d'intégration



# Pipeline d'intégration continue



## Configuration du serveur jenkins

- Le job est triggered lors d'un push sur le github repository Project 9.
- La base de données test est déployée dans un container docker (réinitialisée à chaque lancement).
- Les tests sont lancés avec les 2 *profiles* afin de générer les tests d'intégration des couches consumer et business.
- Jacoco génère, lors de la phase test, les rapports de couverture de code.
- Le server Sonarqube déployé dans un container docker analyse les rapports et la qualité du code.



# Merci

Open Class Rooms : Projet 9