## Testez vos développements Java

#### Méthodes

- Modification de l'environnement de test
- Écriture des tests
- Correction des bugs
- Implémentation des TODOs
- Refactoring et correction du code avec Sonar
- Intégration Continue Travis/Jenkins

# Modification de l'environnement de test

#### Séparation des environnements

- Base PostgreSQL pour le profile de production
- Base H2 en mémoire pour profiles de test

```
<id>production</id>
    properties>
        <myerp.database>MYERP</myerp.database>
    </properties>
    <dependencies>
        <dependency>
           <groupId>org.postgresql</groupId>
           <artifactId>postgresql</artifactId>
           <scope>runtime</scope>
       </dependency>
    </dependencies>
</profile>
ofile>
    <id>test-business</id>
    properties>
        <spring.profiles.active>test-business/spring.profiles.active>
       <myerp.database>TEST</myerp.database>
    </properties>
    <dependencies>
        <dependency>
           <groupId>com.h2database
           <artifactId>h2</artifactId>
           <scope>test</scope>
       </dependency>
    </dependencies>
</profile>
```

#### Passage à Junit5

- Utilisation de Junit5
- Ajout de AssertJ pour les assertions
- Mockito pour simuler les classes
- Mise à jour des différentes dépendances

```
<dependency>
   <groupId>org.junit.jupiter
   <artifactId>junit-jupiter-engine</artifactId>
   <version>${junit-jupiter.version}
   <scope>test</scope>
</dependency>
<dependency>
   <groupId>org.junit.vintage
   <artifactId>junit-vintage-engine</artifactId>
   <version>5.1.0
   <scope>test</scope>
</dependency>
<dependency>
   <groupId>org.assertj
   <artifactId>assertj-core</artifactId>
   <version>${assertj-core.version}
   <scope>test</scope>
</dependency>
<dependency>
   <groupId>org.mockito/groupId>
   <artifactId>mockito-junit-jupiter</artifactId>
   <version>${mockito-junit-jupiter.version}
   <scope>test</scope>
</dependency>
```

#### Outils d'analyse de code

- maven-project-info-reports
- maven-surefire-report
- maven-checkstyle
- cobertura-maven
- jacoco-maven
- sonar-maven

```
<sonar.projectKey>pedsf1968_myerp</sonar.projectKey>
<sonar.moduleKey>${project.artifactId}</sonar.moduleKey>
<sonar.organization>pedsf1968</sonar.organization>
<sonar.host.url>https://sonarcloud.io</sonar.host.url>
<sonar.login>9e78c9485e384e93f7bf7a2f68c024a6c15c6576
<sonar.language>java</sonar.language>
<sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
<sonar.dynamicAnalysis>reuseReports</sonar.dynamicAnalysis>
<sonar-maven-plugin.version>3.7.0.1746</sonar-maven-plugin.version>
<sonar.coverage.jacoco.xmlReportPaths>target/site/jacoco/jacoco.xml,../target/site/jaco
<sonar.coverage.exclusions>
   myerp-technical/src/main/java/com/dummy/myerp/technical/exception/*.java,
   myerp-technical/src/main/java/com/dummy/myerp/technical/util/spring/NullFactoryBean
   myerp-consumer/src/main/java/com/dummy/myerp/consumer/dao/impl/db/rowmapper/comptab
   myerp-consumer/src/main/java/com/dummy/myerp/consumer/dao/impl/db/DBN.java,
</sonar.coverage.exclusions>
```

# Écriture des tests

#### Écriture des tests unitaires

- Pour les MODEL dans myerp-model
- Pour les méthodes de bases dans myerp-consumer
- Pour les méthodes de bases dans myerp-busines

## Écriture des tests d'intégration

## Myerp-consumer sur les méthodes des classes

- CompteComptable
- JournalComptable
- EcritureComptable
- SequenceEcritureComptable

## Myerp-businness sur les méthodes

- De vérification d'EcritureComptable
- CRUD d'EcritureComptable

# Correction des bugs

### EcritureComptable.getTotalCredit()

Original

```
public BigDecimal getTotalCredit() {
    BigDecimal vRetour = BigDecimal.ZERO;
    for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
        // replace getDebit() call with getCredit
        if (vLigneEcritureComptable.getCredit() != null) {
            vRetour = vRetour.add(vLigneEcritureComptable.getDebit());
        }
    }
    return vRetour;
}
```

```
public BigDecimal getTotalCredit() {
    BigDecimal vRetour = BigDecimal.ZERO;
    for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
        // replace getDebit() call with getCredit
        if (vLigneEcritureComptable.getCredit() != null) {
            vRetour = vRetour.add(vLigneEcritureComptable.getCredit());
        }
    }
    return vRetour;
}
```

### EcritureComptable.isEquilibree()

Original

```
public boolean isEquilibree() {
   boolean vRetour = this.getTotalDebit().equals(getTotalCredit());
   return vRetour;
}
```

```
public boolean isEquilibree() {
    // equal compare string and not the value
    // compareTo return 0 if it's the same value
    return this.getTotalDebit().compareTo(getTotalCredit()) == 0;
}
```

#### ResultSetHelper.getDate()

Original

```
public static Date getDate(ResultSet pRS, String pColName) throws SQLException {
    Date yDate = pRS.getDate(pColName);
    if (yDate != null) {
        vDate = DateUtils.truncate(vDate, Calendar.DATE);
    }
    return yDate;
}
```

```
public static Date getDate(ResultSet pRS, String pColName) throws SQLException {
   if (pRS==null || pColName==null) {
      throw new SQLException(ERROR_MESSAGE);
   }
   return pRS.getDate(pColName);
}
```

#### sqlContext.xml

Original

#### ComptabiliteManagerImpl.updateEcritureComptable()

Original

```
public void updateEcritureComptable(EcritureComptable pEcritureComptable) {
    TransactionStatus vTS = getTransactionManager().beginTransactionMyERP();
    try {
        getDaoProxy().getComptabiliteDao().updateEcritureComptable(pEcritureComptable);
        getTransactionManager().commitMyERP(vTS);
        vTS = null;
    } finally {
        getTransactionManager().rollbackMyERP(vTS);
    }
}
```

```
public void updateEcritureComptable(EcritureComptable pEcritureComptable) throws FunctionalException {
    this.checkEcritureComptable(pEcritureComptable);
    TransactionStatus vTS = getTransactionManager().beginTransactionMyERP();
    try {
        getDaoProxy().getComptabiliteDao().updateEcritureComptable(pEcritureComptable);
        getTransactionManager().commitMyERP(vTS);
        vTS = null;
    } finally {
        getTransactionManager().rollbackMyERP(vTS);
    }
}
```

# Implémentation des TODOs

#### ComptabiliteManagerImpl.addReference()

```
public synchronized void {\sf addReference}(EcritureComptable <code>pEcritureComptable</code>) {\sf throws} <code>FunctionalException</code> {\sf fine}
   int numeroSequence = 1; // initialise with 1 to not increment
   SequenceEcritureComptable vSEC = new SequenceEcritureComptable();
   Integer annee = Integer.parseInt(new SimpleDateFormat( pattern: "yyyy").format(pEcritureComptable.getDate()));
   SequenceEcritureComptable vLastSEC = getDaoProxy().getComptabiliteDao().getLastSeqOfTheYear(annee, pEcritureComptable.getJo
   numeroSequence += (vLastSEC==null?0:vLastSEC.getDerniereValeur());
   vSEC.setAnnee(annee);
   vSEC.setDerniereValeur(numeroSequence);
   String reference = String.format("%s-%s/%05d",
         pEcritureComptable.getJournal().getCode(),
         annee,
         numeroSequence);
   pEcritureComptable.setReference(reference);
   getTransactionManager().beginTransactionMyERP();
   this.updateEcritureComptable(pEcritureComptable);
   if (numeroSequence == 1) {
       this.insertEcritureComptable(pEcritureComptable);
       this.updateEcritureComptable(pEcritureComptable);
```

#### ComptabiliteManagerImpl.checkEcritureComptableRG5()

```
protected void checkEcritureComptableRG5(EcritureComptable pEcritureComptable) throws FunctionalException {
   if (pEcritureComptable.getJournal()!=null && pEcritureComptable.getReference()!=null) {
       String code = pEcritureComptable.getReference().split( regex: "-")[0];
       if (!code.equals(pEcritureComptable.getJournal().getCode())) {
           LOGGER.error("Pas le même Code de Journal Comptable : Code {} et Code dans Reference {}",
                 pEcritureComptable.getJournal().getCode(),code);
           throw new FunctionalException(
       String year = pEcritureComptable.getReference().split( regex: "[-/]")[1];
       java.sql.Date dateComptable = pEcritureComptable.getDate();
       String dateYear = String.vαlueOf(dateComptable.toLocalDate().getYear());
       if (!year.equals(dateYear)) {
           LOGGER.error("Pas le même Date de Journal Comptable : Date {} et Année dans Reference {}",dateYear,year);
           throw new FunctionalException(
```

#### ComptabiliteManagerImpl.checkEcritureComptableRG7()

```
protected void checkEcritureComptableRG7(EcritureComptable pEcritureComptable) throws FunctionalException {
   List<LigneEcritureComptable> ligneEcritureComptables = pEcritureComptable.getListLigneEcriture();
   for(LigneEcritureComptable ligneEcritureComptable : ligneEcritureComptables) {
       if(ligneEcritureComptable.getCredit() != null && getNumberOfDecimalPlaces(ligneEcritureComptable.getCredit()) > 2) {
           throw new FunctionalException("Un crédit ne peut pas avoir plus de deux chiffres après la virgule!");
       if(ligneEcritureComptable.getDebit() != null && getNumberOfDecimalPlaces(ligneEcritureComptable.getDebit()) > 2) {
           throw new FunctionalException("Un débit ne peut pas avoir plus de deux chiffres après la virgule!");
int getNumberOfDecimalPlaces(BigDecimal bigDecimal) {
   String string = bigDecimal.stripTrailingZeros().toPlainString();
   int index = string.indexOf('.');
   return index < 0 ? 0 : string.length() - index - 1;</pre>
```

#### SequenceEcritureComptableRM

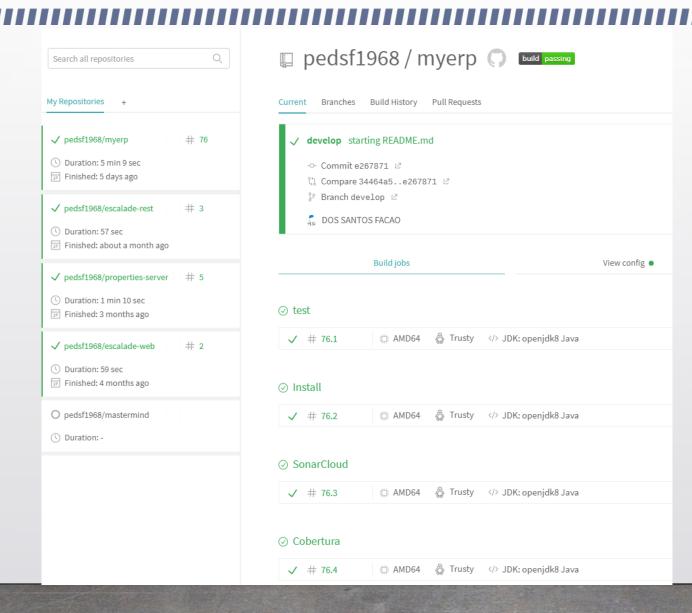
```
package com.dummy.myerp.consumer.dao.impl.db.rowmapper.comptabilite;
import com.dummy.myerp.consumer.dao.impl.db.DBN;
import com.dummy.myerp.model.bean.comptabilite.SequenceEcritureComptable;
import org.springframework.jdbc.core.RowMapper;
import java.sql.ResultSet;
import java.sql.SQLException;
public class SequenceEcritureComptableRM implements RowMapper<SequenceEcritureComptable> {
   @Override
   public SequenceEcritureComptable mapRow(ResultSet resultSet, int i) throws SQLException {
       SequenceEcritureComptable vBean = new SequenceEcritureComptable();
       vBean.setAnnee(resultSet.getInt(DBN.ANNEE));
       vBean.setDerniereValeur(resultSet.getInt(DBN.DERNIERE_VALEUR));
       return vBean;
```

# Intégration Continue avec Travis

#### Travis.yml

```
language: java
dist: trusty
jdk: openjdk8
 - chmod +x mvnw
after_success:
 - bash <(curl -s https://codecov.io/bash)
   - stage: test
     script: ./mvnw clean test
   - stage: SonarCloud
     script: ./mvnw verify sonar:sonar
   - stage: Cobertura
     script: ./mvnw cobertura:cobertura
 directories:
   - $HOME/.m2
```

#### https://travis-ci.org



# Intégration Continue avec Jenkins

#### Docker-compose.yml

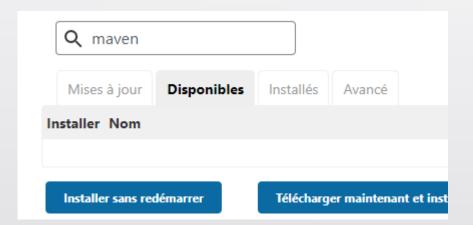
```
image: jenkins/jenkins:lts
container_name: jenkins
privileged: true
user: root
restart: unless-stopped
volumes:
  - ./docker/dev/jenkins:/var/jenkins_home
  - ./src:/var/jenkins_home/workspace/Myerp
networks:
 net:
    ipv4_address: 10.5.0.4
```

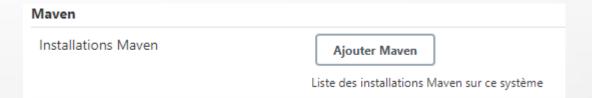
#### Configuration – Maven Integration

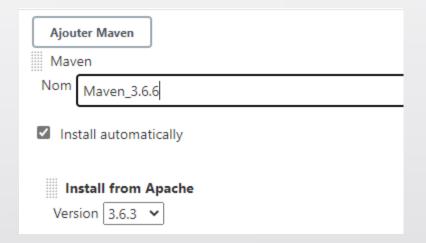


#### **Gestion des plugins**

Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.





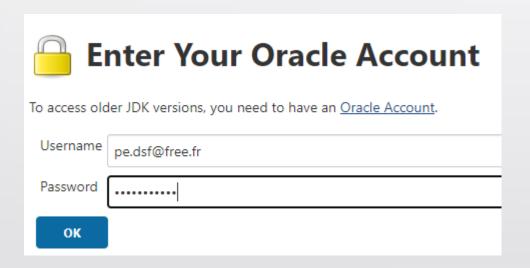


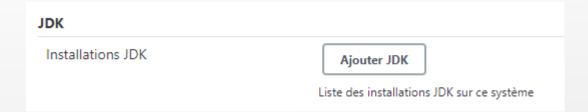
#### Configuration – JDK

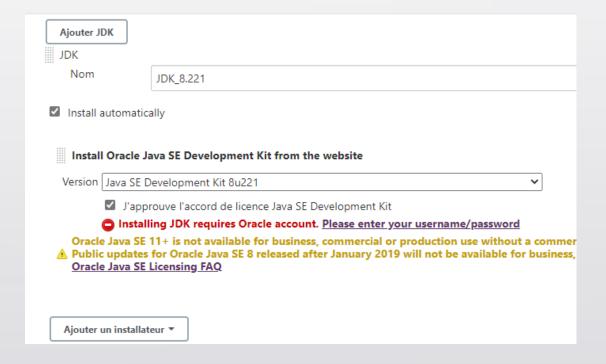


#### Configuration globale des outils

Configurer les outils, leur localisation et les installeurs automatiques.







### Configuration – JOB

#### Saisissez un nom

Myerp

» Champ obligatoire



#### Construire un projet free-style

Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous po pour tout autre chose qu'un build logiciel.



#### Construire un projet maven

Construit un projet avec maven. Jenkins utilise directement vos fichiers POM et diminue rad



#### **Pipeline**

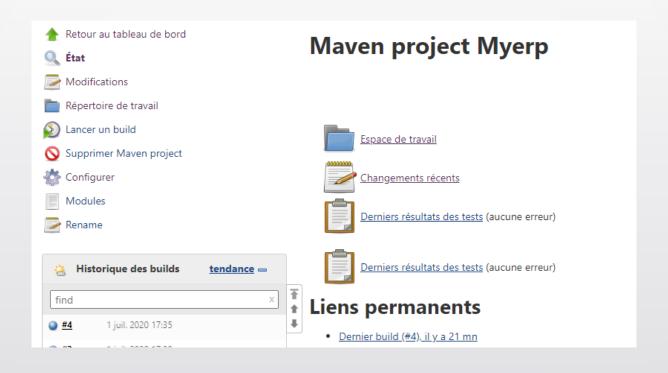
Orchestrates long-running activities that can span multiple build agents. Suitable for buildin



Construire un projet multi configuration

Build	
POM Racine	pom.xml
Goals et options	verify sonar:sonar

### Configuration - Build



## SonarQube

### https://sonarcloud.io/

