3.26 Tests

Introduction

ESUP-Commons utilise JUnit. C'est un framework de test unitaire bien intégré avec Spring.

Pour l'utiliser dans votre projet vous devez le référencer comme dépendance de votre projet avec un scope test. Ce scope fera qu'il ne sera utiliser que pendant les phases de test.

```
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.8.1</version>
 <scope>test</scope>
</dependency>
```

Votre code de test doit être placé dans **src/test/java**

Si vous avez besoin de ressources spécifiques aux tests, elles peuvent être placées dans **src/test/resources**

Le code de test est utilisé lors de la commande **mvn test**. Il est aussi appelé lors des phase de packaging, d'installation, de déploiement et de release.

Code de test

Votre code de test doit être placé dans **src/test/java**

Si vous avez besoin de ressources spécifiques aux tests, elles peuvent être placées dans **src/test/resources**

Annoter une classe de test

```
@ContextConfiguration(locations="/META-INF/applicationContext.xml")
@RunWith(SpringJUnit4ClassRunner.class)
public class ExempleTest {
```

@ContextConfiguration	Permet de spécifier où se trouve le fichier de Configuration Spring Typiquement il peut se trouver dans **src/test/resources** et être spécifique à cette classe de test Il n'y a rien d'autre à faire pour charger le context Spring
@RunWith(SpringJUnit4ClassRunner.class)	Informe JUnit qu'il s'exécute dans un environnement Spring

Annoter une méthode de test

```
@Test
public void testTruc() {
```

@Test	Marque la méthode comme une méthode de test Chaque méthode avec cette annotation fera l'objet d'un test
@Test(expected = Exception.class)	On peut parfois vouloir tester un exception

@Test(timeout=200)	On peut aussi vouloir tester un temps de réponse

Initialisation

Injecter un service

Il est classique de tester une méthode d'un service. Pour le faire il faut y avoir accès. Une solution simple consiste à utiliser une annotation pour l'injecter dans le classe de test :

```
@Autowired
DomainService domainService;
```

Si vous avez plusieurs implémentation de **DomainService** vous devez préciser celle que vous utilisez :

```
@Autowired
@Qualifier("myServiceImpl")
DomainService domainService;
```


Ces annotations sont spécifiques à Spring. JSR 250 normalise ces annotations. Pour les utiliser vous devez ajouter une dépendance :

<dependency>
 <groupId>javax.inject</groupId>
 <artifactId>javax.inject</artifactId>
 <version>1</version>
</dependency>

L'annotation à utiliser est soit :

@Inject
DomainService domainService;

soit :

@Resource(name="myServiceImpl")
DomainService domainService;

Code d'initialisation

On peut avoir besoin de faire des opérations avant et/ou après des tests. Pour cela on peut créer des méthodes que l'on peut annoter :

@Before	La méthode sera exécutée avant chaque test
@After	La méthode sera exécutée après chaque test
@BeforeClass	La méthode sera exécutée une fois avant les tests de la classe
@AfterClass	La méthode sera exécutée une fois après les tests de la classe