

# Fms Academy 2022

## Formation Java Spring Angular

### M11 : Tests unitaires

# SOMMAIRE

- POURQUOI & COMMENT TESTER UNE APPLICATION ?
- DIFFÉRENTS TYPES DE TEST
- MAVEN ET JUNIT
- SPRING BOOT, MAVEN ET JUNIT
- TDD (TEST DRIVEN DEVELOPMENT)
- ASSERTJ
- @DATAJPATEST
- TESTER UN CONTROLLER
- DÉPLOIEMENT
- A RETENIR
- ETAPES SUIVANTES

# Pourquoi tester une application ?

Pour répondre à des impératifs (d'agilité) :

- qualité des applications (bugs...)
- conformité aux attentes du client (spécification fonctionnelles)
- anticiper sur les cas d'utilisations et sur l'évolution d'une application
- contraintes techniques & délais de livraison (cout)
- confiance du client (éviter la régression)
- communication entre les développeurs

## Comment tester une application ?

En réalisant nous même des **tests manuels**.

Le pb c'est qu'ils sont souvent incomplet voir insuffisant et ils ne tiennent pas toujours compte de l'évolution des applications.

En effet, sur des projets importants, nombreuses sont les interventions de devs, de versions en versions, est-ce qu'ils répètent tous les tests ?

**En automatisant les tests**, on garanti que tous les tests prévus sur chaque version d'une application seront effectués automatiquement

# Différents types de tests

Mike Cohn, l'un des fondateurs du mouvement Agile, présente ces types de tests sous forme de **pyramide des tests**.

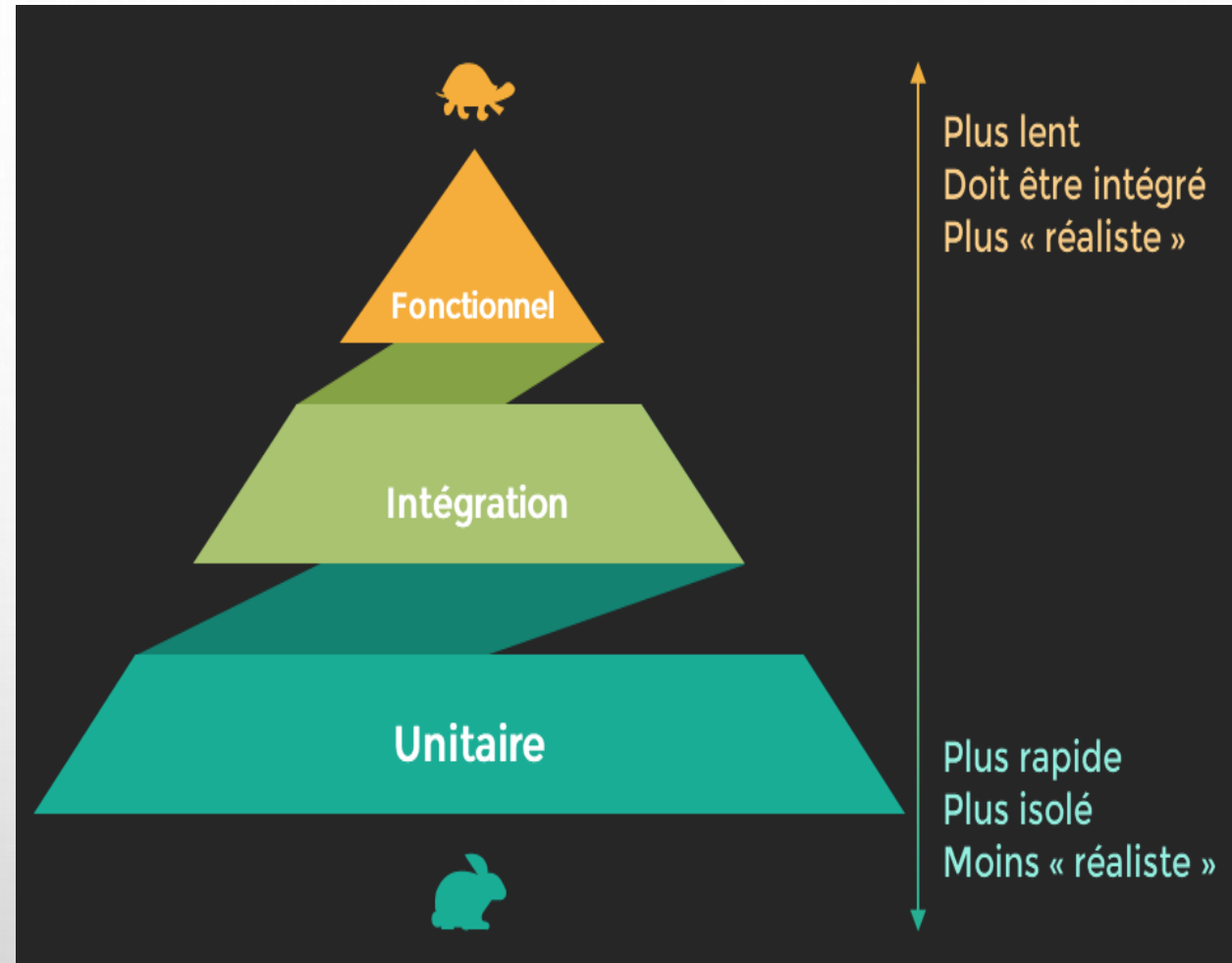
Les tests fonctionnelles (end to end) visent à simuler le comportement d'un User final.

Les tests d'intégrations vérifient que les unités de code fonctionnent ensemble correctement comme prévu.

Par ex, tester l'interaction entre 2 couches

Les tests unitaires sont les plus nombreux et **les plus importants**. Ils garantissent que chaque unités de code ou fonctionnalités se comporte comme prévu.

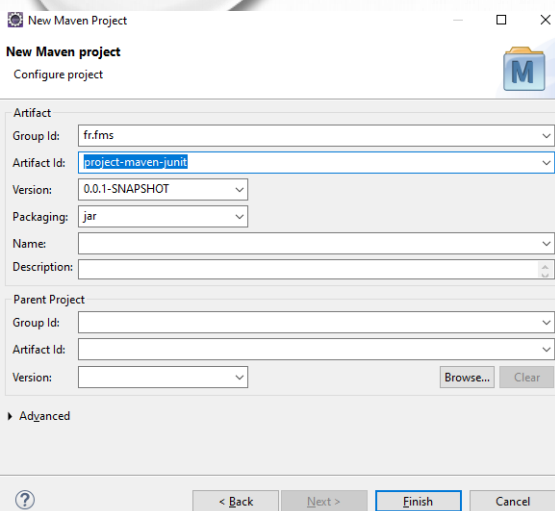
Ils sont rapides, faciles à exécuter, stables quelle que soit les évolutions et rentables car une fois écrits, ils sont exécutés de nombreuses fois.



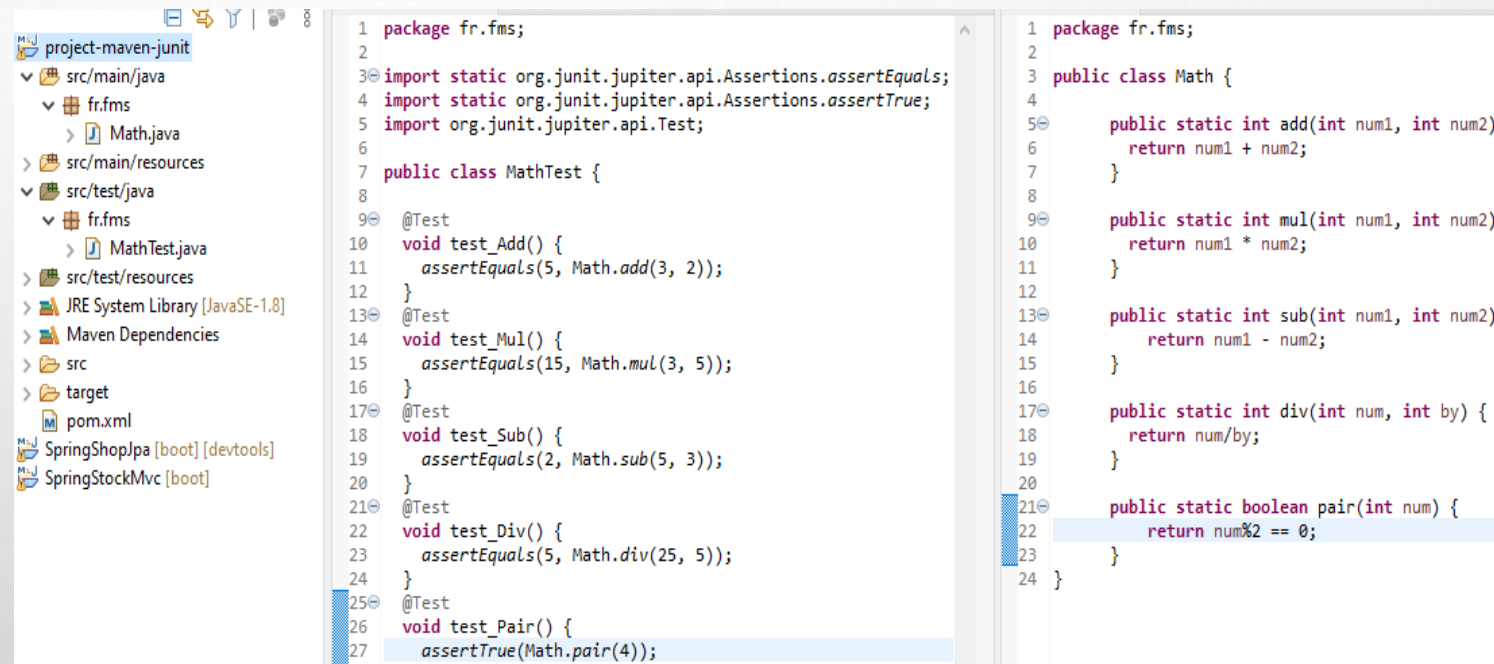
Plus un bug est détecté tôt, moins son coût de correction est élevé :

- Moins on réalise de tests unitaires, plus on va devoir faire de tests manuels pour le débuser (mauvaise pratique)
- Plus notre couverture de tests unitaires est de qualité et importante, plus vite un bug est détecté (bonne pratique)

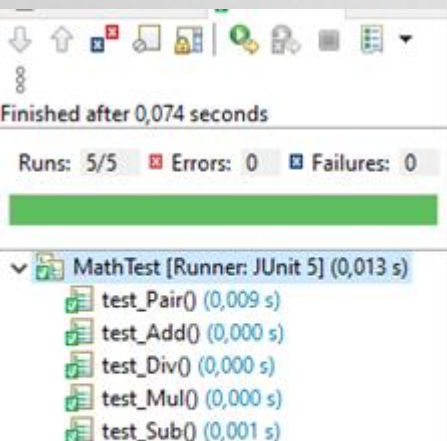
# MAVEN ET JUNIT



Ajouter les dépendances Junit dans pom.xml  
Puis ajouter une classe à tester et une classe de test dans les répertoires



Pour exécuter les tests :  
Project/clic droit/run as /  
junit test



C:\... mvn test

# SPRING BOOT, MAVEN ET JUNIT

On souhaite vérifier si le montant total de notre panier est cohérent avec les prix des articles ajoutés qq soit les quantités.



The screenshot displays an IDE environment with three main components:

- Project Explorer (Left):** Shows the project structure for 'SpringShopJpa [boot] [devtools]'. The 'src/test/java' directory is expanded, showing 'fr.fms' and 'SpringShopJpaApplicationTests.java'.
- Source Editor (Right):** Displays the code for 'SpringShopJpaApplicationTests.java'. The code includes imports for JUnit and Spring Boot test classes, and two test methods: 'contextLoads()' and 'testTotalAmountCart()'. The 'testTotalAmountCart()' method adds three items to a cart and asserts the total amount is 1250.
- JUnit Runner (Bottom):** Shows the test results. It indicates 'Finished after 5,652 seconds' and 'Runs: 2/2', 'Errors: 0', 'Failures: 0'. The test results list shows 'SpringShopJpaApplicationTests [Runner: JUnit]' with two successful tests: 'testTotalAmountCart() (0,142 s)' and 'contextLoads() (0,004 s)'.

Pour 2 tests lancés avec succès :

- **contextLoads** vérifier que l'assertion est bien fausse
- **testTotalAmountCart** vérifie que le montant du panier est cohérent aux ajouts

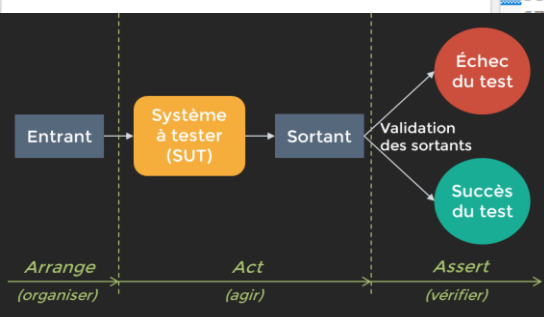
Failures = nb de test en échec  
Errors = nb d'erreur de compil



SpringShopJpaApplicationTests [Runner: JUnit]

- multiply\_shouldReturnZero(int) (0,176 s)
  - 1 x 0 doit être égal à 0 (0,176 s)
  - 2 x 0 doit être égal à 0 (0,008 s)
  - 42 x 0 doit être égal à 0 (0,005 s)
  - 1011 x 0 doit être égal à 0 (0,003 s)
  - 5089 x 0 doit être égal à 0 (0,004 s)
- orderShouldComputeLess1Second() (0,511 s)
- testTotalAmountCart() (0,006 s)

Failure Trace



```

1 package fr.fms;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
5 @SpringBootTest
6 class SpringShopJpaApplicationTests {
7     @Autowired
8     IBusinessImpl business;
9
10    private static Instant startedAt;
11
12    @BeforeEach
13    public void beforeEachTest() {
14        System.out.println("avant chaque test");
15    }
16
17    @AfterEach
18    public void afterEachTest() {
19        System.out.println("après chaque test");
20    }
21
22    @BeforeAll
23    public static void initStartingTime() {
24        System.out.println("Appel avant tous les tests");
25        startedAt = Instant.now();
26    }
27
28    @AfterAll
29    public static void showTestDuration() {
30        System.out.println("Appel après tous les tests");
31        final Instant endedAt = Instant.now();
32        final long duration = Duration.between(startedAt, endedAt).toMillis();
33        System.out.println(MessageFormat.format("Durée des tests : {0} ms", duration));
34    }
35
36    @ParameterizedTest(name = "{0} x 0 doit être égal à 0")
37    @ValueSource(ints = { 1, 2, 42, 1011, 5089 })
38    public void multiply_shouldReturnZero(int arg) {
39        assertEquals(0, arg*0);
40    }
41
42    @Timeout(1)
43    @Test
44    public void orderShouldComputeLess1Second() {
45        business.order();
46    }
47
48    @Test
49    void testTotalAmountCart() {
50        // Arrange
51        business.addArtToCart(new Article((long)1,"Samsung","Samsung S8",250,1,null));
52        business.addArtToCart(new Article((long)2,"Samsung","Samsung S9",250,1,null));
53        business.addArtToCart(new Article((long)3,"iPhone","iPhone 10",500,1,null));
54
55        // Act
56        double amount = business.getTotalAmount();
57
58        // Assert
59        assertEquals(amount,1000);
60    }
61

```

Appel avant tous les tests  
 13:51:18.517 [main] DEBUG org.springframework...  
 13:51:18.522 [main] DEBUG org.springframework...

Spring Boot (v2.6.0)

Date	Time	Level	Message
2022-06-26	13:51:18.878	INFO	16604 --- [main]
2022-06-26	13:51:18.879	INFO	16604 --- [main]
2022-06-26	13:51:20.017	INFO	16604 --- [main]
2022-06-26	13:51:20.114	INFO	16604 --- [main]
2022-06-26	13:51:21.026	INFO	16604 --- [main]
2022-06-26	13:51:21.122	INFO	16604 --- [main]
2022-06-26	13:51:21.396	INFO	16604 --- [main]
2022-06-26	13:51:21.665	INFO	16604 --- [main]
2022-06-26	13:51:21.759	INFO	16604 --- [main]
2022-06-26	13:51:21.780	INFO	16604 --- [main]
2022-06-26	13:51:22.855	INFO	16604 --- [main]
2022-06-26	13:51:22.866	INFO	16604 --- [main]
2022-06-26	13:51:23.859	WARN	16604 --- [main]
2022-06-26	13:51:24.403	INFO	16604 --- [main]
2022-06-26	13:51:25.981	INFO	16604 --- [main]

Using generated security password: be568f54

2022-06-26 13:51:26.007 INFO 16604 --- [main]  
 avant chaque test  
 après chaque test  
 avant chaque test  
 après chaque test  
 avant chaque test  
 après chaque test  
 avant chaque test  
 après chaque test  
 avant chaque test  
 après chaque test  
 avant chaque test  
 après chaque test  
 avant chaque test  
 après chaque test  
 Appel après tous les tests  
 Durée des tests : 8 362 ms

# TDD : Test Driven Development Ou Développement piloté par les tests

En règle générale, on a tendance à coder une fonctionnalité puis réaliser le test associé, dans cette approche, on code le test en fonction du besoin, d'abord le test sera en échec

```
68 @Test
69 void testTotalAmountCart() {
70     // Arrange
71     business.addArtToCart(new Article((long)1, "Samsung", "Samsung S8", 250, 1, null));
72     business.addArtToCart(new Article((long)2, "Samsung", "Samsung S9", 250, 1, null));
73     business.addArtToCart(new Article((long)3, "iPhone", "iPhone 10", 500, 1, null));
74
75     // Act
76     double amount = business.getTotalAmount();
77
78     // Assert
79     assertEquals(amount, 1000);
80 }
```

The method `getTotalAmount()` is undefined for the type `IBusinessImpl`  
2 quick fixes available:  
● [Create method 'getTotalAmount\(\)' in type 'IBusinessImpl'](#)

Runs: 7/7 Errors: 1 Failures: 0

SpringShopJpaApplicationTests [Runner: JUnit]

- multiply\_shouldReturnZero(int) (0,193 s)
- orderShouldComputeLess1Second() (0,513 s)
- testTotalAmountCart() (0,010 s)

Puis on ajoute le code dans la méthode de la classe :

```
public double getTotalAmount() {
    double total = 0;
    for(Article article : cart.values()) {
        total += article.getPrice()*article.getQuantity();
    }
    return total;
}
```

Finished after 8,797 seconds

Runs: 7/7 Errors: 0 Failures: 0

SpringShopJpaApplicationTests [Runner: JUnit]

- multiply\_shouldReturnZero(int) (0,186 s)
- orderShouldComputeLess1Second() (0,510 s)
- testTotalAmountCart() (0,006 s)

Cette approche garantie qu'on reste centré sur les fonctionnalités  
De plus, le code écrit après le test est plus simple à tester,  
plus clair à comprendre et plus facile à maintenir : Robuste

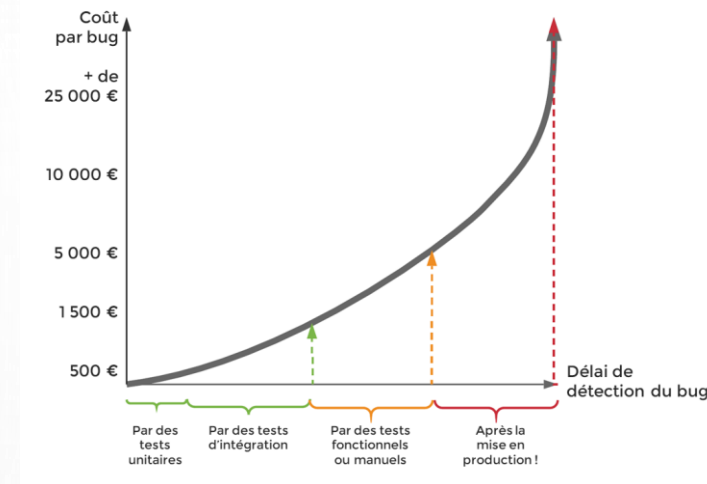


# ASSERTJ

La courbe montre que + un bug est découvert tardivement, plus il est couteux de le corriger aussi comment écrire des tests :

- Simples (à tester)
- Clairs (à comprendre)
- Faciles (à maintenir)

→ AssertJ apporte à Junit tout ce complément



## Cas de test

## Assertions JUnit

## Assertions AssertJ

**Un nom est compris entre 5 et 10 caractères.**

```
assertTrue(name.length > 4 &&
name.length < 11);
```

```
assertThat(name)
.hasSizeGreaterThan(4)
.hasSizeLessThan(11);
```

**Un nom est situé dans la première moitié de l'alphabet**

```
assertTrue(
name.compareTo("A") >= 0
&& name.compareTo("M") <= 0);
```

```
assertThat(name).isBetween("A",
"M");
```

**Une date et heure locale se situent aujourd'hui ou dans le futur.**

```
assertTrue(
dateTime.toLocalDate().isAfter(LocalDate.now()) ||
dateTime.toLocalDate().isEqual(LocalDate.now()));
```

```
assertThat(dateTime.toLocalDate())
.isAfterOrEqualTo(LocalDate.now());
```

# @DataJpaTest

Si vous souhaitez maintenant tester vos entités Jpa dans une base de données virtuelle (H2) avec l'accès aux données persistantes sans modifications possibles :

```
import org.junit.jupiter.api.Test;

@RunWith(SpringRunner.class)
@DataJpaTest
@AutoConfigureTestDatabase(replace=Replace.NONE)
public class SpringShopJpaTests {
    @Autowired
    ArticleRepository articleRepository;

    @Autowired
    CategoryRepository categoryRepository;

    @Test
    void test_add_article() {
        //GIVEN
        Category anonymous = categoryRepository.save(new Category(null,"anonymous",null));
        articleRepository.save(new Article(null,"incognito","incognito 007" , 375 , 1 , anonymous));

        //WHEN
        Article article = articleRepository.findByBrandContains("incognito").get(0);

        //THEN
        assertEquals("incognito 007", article.getDescription());
    }

    @Test
    void should_find_one_article() {
        Iterable<Article> articles = articleRepository.findAll();
        assertThat(articles).isNotEmpty();
    }
}
```

Notre base n'a subit aucun changement

```
21 | Panasonic | HT | 1500 | 1 | 6 |
22 | Philips | L43 | 450 | 1 | 6 |
26 | samsung | samsung S200 | 1500 | 1 | 1 |
29 | Iphone | Iphone 10 | 500 | 1 | 1 |
+-----+-----+-----+-----+-----+-----+
24 rows in set (0.000 sec)

MariaDB [Stock]>
```

# Tester un Controller (MockMvc & [Mockito](#))

```
Debug Project Explorer JUnit x
Finished after 7,661 seconds
Runs: 1/1 Errors: 0 Failures: 0
SpringShopControllersTests [Runner: JUnit 5] (0,239 s)
  test_get_welcome() (0,239 s)
```

```
SpringShopControllersTests.java x
1 package fr.fms;
2
3 import static org.hamcrest.CoreMatchers.containsString;
22
23 @SpringBootTest
24 @AutoConfigureMockMvc
25 public class SpringShopControllersTests {
26
27     @Autowired
28     private MockMvc mvc;
29
30     @Test
31     public void test_get_welcome() throws Exception {
32         //GIVEN
33         mvc.perform(get("/"))
34             .andExpect(status().isOk())
35             .andExpect(content().string(containsString("Hello")));
36     }
37 }
```

```
@RequestMapping("/")
public @ResponseBody String home() {
    return "Hello";
}
```

```
Finished after 0,169 seconds
Runs: 1/1 Errors: 0 Failures: 0
SpringShopControllersTests [Runner: JUnit 5] (0,169 s)
  test_get_welcome() (0,169 s)
```

```
3 import static org.hamcrest.CoreMatchers.containsString;
24
25 @WebMvcTest(ArticleController.class)
26 public class SpringShopControllersTests {
27
28     @Autowired
29     private MockMvc mvc;
30
31     @MockBean
32     private IBusinessImpl businessImpl;
33
34     @Test
35     public void test_get_welcome() throws Exception {
36         when(businessImpl.great()).thenReturn("Hello, Mock");
37
38         this.mvc.perform(get("/greeting"))
39             .andDo(print())
40             .andExpect(status().isOk())
41             .andExpect(content().string(containsString("Hello, Mock")));
42     }
43 }
44 }
```

```
public String great() {
    return "Hello World";
}
```

```
@RequestMapping("/greeting")
public @ResponseBody String greeting() {
    return businessImpl.great();
}
```

```
@Test
public void test_get_articles() throws Exception {
    //.....
    this.mvc.perform(get("/articles")) //THEN
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.brand", is("Samsung")));
    //$ : pointe sur la racine de la structure JSON
    //[0] : pour le 1er élément
    //brand: brand pour l'attribut
    //is : correspond au résultat attendu
}
```

Ex avec une  
api, à suivre

Runs: 10/10
Errors: 0
Failures: 0

SpringShopJpaTests [Runner: JUnit 5] (0,334 s)

- should\_find\_one\_article() (0,269 s)
- test\_add\_article() (0,064 s)

SpringShopJpaApplicationTests [Runner: JUnit 5] (0,573 s)

- multiply\_shouldReturnZero(int) (0,026 s)
- orderShouldComputeLess1Second() (0,521 s)
- testTotalAmontCart() (0,002 s)

SpringShopControllersTests [Runner: JUnit 5] (0,082 s)

- test\_get\_welcome() (0,082 s)

Failure Trace

```

2022-06-27 12:33:32.346 WARN 30012 --- [           main] Jp
2022-06-27 12:33:32.637 INFO 30012 --- [           main] o.
2022-06-27 12:33:33.052 INFO 30012 --- [           main] o.
2022-06-27 12:33:33.052 INFO 30012 --- [           main] o.
2022-06-27 12:33:33.052 INFO 30012 --- [           main] o.
2022-06-27 12:33:33.058 INFO 30012 --- [           main] .s

Using generated security password: d2c9f27c-d1d9-4b98-8680-7

2022-06-27 12:33:33.067 INFO 30012 --- [           main] fr

MockHttpServletRequest:
    HTTP Method = GET
    Request URI = /greeting
    Parameters = {}
    Headers = []
    Body = null
    Session Attrs = {}

Handler:
    Type = fr.fms.web.ArticleController
    Method = fr.fms.web.ArticleController#greeting()

Async:
    Async started = false
    Async result = null

Resolved Exception:
    Type = null

ModelAndView:
    View name = null
    View = null
    Model = null

FlashMap:
    Attributes = null

MockHttpServletResponse:
    Status = 200
    Error message = null
    Headers = [Content-Type:"text/plain;charset=UTF-8"]
    Content type = text/plain;charset=UTF-8
    Body = Hello, Mock
    Forwarded URL = null
    Redirected URL = null
    Cookies = []

```

FMS-EA © El Babili - Tous droits réservés

12

# Déploiement avec maven


Sous Linux, il faut d'abord installer maven :

- sudo apt-get update + sudo apt install maven
- mvn package : compile + test + génère le .jar dans target

Sous Windows, fichier pom.xml/clic droit/ run as / maven build...

Dans le dossier Target donc vous pouvez récupérer « SpringShopMvc-0.0.1-SNAPSHOT.jar » puis le copier coller sur votre bureau et fermer Eclipse avant de l'exécuter.

```
C:\Users\moam\Desktop>java -jar SpringShopMvc-0.0.1-SNAPSHOT.jar
```



The image shows a terminal window with the command to run the Spring Boot application. Below the command, the Spring Boot logo is displayed, which consists of a stylized 'S' and 'B' made of lines, followed by the text 'Spring Boot' and the version '(v2.5.3)'.

Puis rendez vous sur un navigateur pour utiliser l'application normalement

Il est possible sur un même réseaux local de faire un appel depuis un poste client vers un poste dit serveur contenant l'appli qui tourne. Pour ce faire, à la place de localhost, insérer l'adresse IP du poste serveur.

Commande pour obtenir votre Ip sous Linux : hostname -i

Sous windows : ipconfig

NB : marche pas avec les pc airbus hyper verrouillé ;)



# A RETENIR

Rappelons que les tests unitaires ont pour vocation à tester uniquement le contenu d'une méthode, alors que les tests d'intégration impliquent de tester plus largement une fonctionnalité.

La génération de notre Jar est précédé par l'ensemble des tests :

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ SpringShopJpa ---
[INFO] Building jar: C:\Users\EI-BabiliM\SpringWorkspace\SpringShopJpa\target\SpringShopJpa-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.6.0:repackage (repackage) @ SpringShopJpa ---
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.607 s
[INFO] Finished at: 2022-06-27T14:35:26+02:00
[INFO] -----
```

## ETAPES SUIVANTES

- Vérifier la couverture de nos tests avec Jacoco par ex
  - Améliorer la qualité du code avec le formatage et la détection de code mal écrit (SonarQube)
  - Sécuriser notre appli en vérifiant les dépendances (OWASP Dependency Check)
- Maven va nous aider à automatiser ces outils durant le build
- SonarQube va nous aider à observer tous ces indicateurs comme dans un cockpit
- Cucumber permet d'écrire des scénarios de tests(US) repris sous forme de steps (Given, When, Then)  
→ (pour moi) il est clairement orienté vers les tests d'intégrations
- Jenkins va gérer le tout à l'aide de pipeline :

