



# SPRING

Jérémy PERROUAULT



# SPRING TEST

Introduction à  
Spring Test avec JUnit

# TEST UNITAIRE

Un Test Unitaire (TU) permet de vérifier le bon déroulement d'une fonctionnalité

Il doit être atomique

- Léger
- Isolé
- Rapide

# PRÉSENTATION DE SPRING TEST

Spring permet de réaliser des tests unitaires

- Reproductibles
- Compréhensibles

Combiné à un Framework de test, tel que JUnit

# ORGANISATION

Création d'un répertoire « src/test » au même titre que « src /main»

- Dans Java Resources

Les packages restent identiques

On ajoute « Test » à la fin du nom de la classe qu'on veut tester

Exemple :

- Tester la classe `fr.formation.dao.ProduitDAO`
- ➔ Création d'une classe `fr.formation.dao.ProduitDAOTest` dans le répertoire « test »

# ORGANISATION

Création d'un répertoire « test/resources » au même titre que « main/resources »

- Dans Java Resources

Tous nos fichiers de configuration seront recopiés ici

- S'il y a besoin de les adapter pour les tests (base de données différente par exemple)

# CONFIGURATION

Ajout de la dépendance Spring-Test

```
<!-- Spring Test -->  
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-test</artifactId>  
  <version>5.0.6.RELEASE</version>  
  <scope>test</scope>  
</dependency>
```

# CONFIGURATION (JUNIT)

Ajout de la dépendance JUnit à Maven

```
<!-- JUnit -->  
<dependency>  
  <groupId>junit</groupId>  
  <artifactId>junit</artifactId>  
  <version>4.12</version>  
  <scope>test</scope>  
</dependency>
```



# CONFIGURATION (JUNIT)

Annotation	Définition
@Test	Méthode de test qui sera exécutée
@BeforeClass	Méthode statique qui sera exécutée avant le premier test
@AfterClass	Méthode statique qui sera exécutée après le dernier test
@Before	Méthode qui sera exécutée avant chaque test
@After	Méthode qui sera exécutée après chaque test

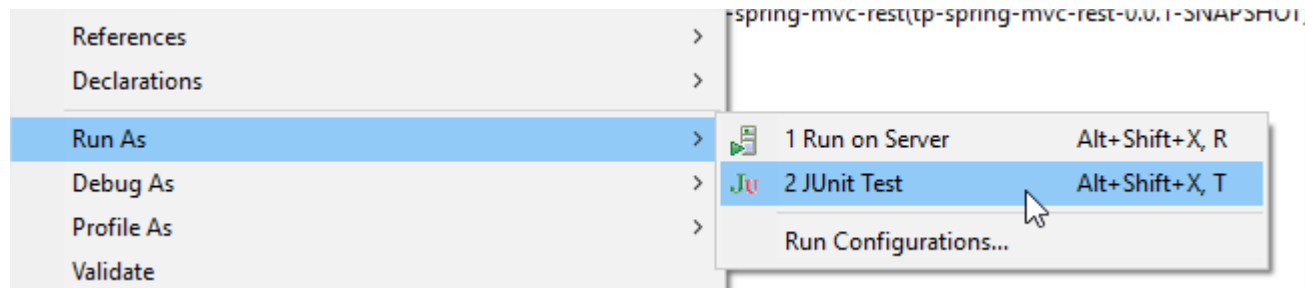
# ASSERTION

Import du package org.junit.Assert.\*

Assertion	Description	Déclenchement
fail	Echec	Toujours
assertEquals	Vérifier une égalité	Si différent
assertTrue	Vérifier une condition	Si la condition est fausse
assertNotEquals	Vérifier une non-égalité	Si équivalent
assertFalse	Vérifier une non-condition	Si la condition est vraie
assertNull	Vérifier si l'objet est null	Si l'objet n'est pas null
assertNotNull	Vérifier si l'objet n'est pas null	Si l'objet est null

# EXÉCUTION (JUNIT)

## Sélection de JUnit Test



# EXERCICE

Créer une classe SuperMaths

- additionner(a, b)
- soustraire(a, b)

Rédiger un test

- Sysout avant le démarrage du premier test
- Initialisation d'un objet SuperMaths
- Tester la méthode additionner
- Tester la méthode soustraire

# CONFIGURATION (JUNIT)

## Classe de test annotée

- `@RunWith(SpringJUnit4ClassRunner.class)`
- Selon XML ou Classe
  - `@ContextConfiguration(locations={ "/application-context.xml" })`
  - `@ContextConfiguration(classes={ AppConfig.class })`
- `@WebAppConfiguration` (sera utilisée pour retrouver le contexte d'application Web)

# EXERCICE

Rédiger un test pour vérifier qu'un bean « guitariste » existe bien

# EXERCICE

Il est possible de tester l'exécution d'un AOP dans un environnement contrôlé

- Utilisation d'un *proxy* **AspectJProxyFactory**

```
PublicAspect myAspect = new PublicAspect();
AspectJProxyFactory myFactory = new AspectJProxyFactory(this.pianiste);

myFactory.addAspect(myAspect);
IMusicien myProxyMusicien = myFactory.getProxy();
```

- L'idée : isoler le pianiste dans un nouveau contexte, et n'y attacher qu'un Aspect

# EXERCICE

Rédiger un test pour vérifier le bonne exécution de PublicAspect sur un pianiste



# HIBERNATE & DAO

Spring gère nos DAO et les transactions

Une transaction peut être commitée ou rollbackée

Par défaut

- L'ajout, la modification et la suppression en base de données est effective (commit)
- Spring gère la transaction et la commit automatiquement

Pour modifier ce comportement (sur la méthode ou sur la classe de test)

- `@Transactional`
- `@Rollback(true)` (`@Commit` pour le cas inverse)

# EXERCICE

Rédiger un test pour

- Rechercher un produit
  - Ajouter un produit
  - Supprimer un produit
- 
- On veut vérifier que tout est fonctionnel mais sans les modifications appliquées en base de données

# MOCK — MOCK MVC

Une classe qui va simuler l'exécution d'une « vraie » classe

- Mock MVC va simuler nos classes contrôleurs
  - Sans avoir besoin de démarrer le serveur d'application

# MOCK — MOCK MVC

On injecte WebApplicationContext

```
@Autowired  
private WebApplicationContext webApplicationContext;
```

On construit mockMvc (de type MockMvc) avec MockMvcBuilders

```
@Before  
public void before() {  
    this.mockMvc = MockMvcBuilders.webAppContextSetup(this.webApplicationContext).build();  
}
```

# MOCK — MOCK MVC

Vérifier que l'adresse "/produit" est accessible (GET)

```
this.mockMvc.perform(get("/produit"))  
    .andExpect(status().isOk());
```

Vérifier que "/home" est accessible (GET) et retourne la vue "home"

```
this.mockMvc.perform(get("/home"))  
    .andExpect(status().isOk())  
    .andExpect(view().name("home"));
```

# MOCK — MOCK MVC

Vérifier que l'adresse `"/produit"` (PUT) ajoute le produit

```
ObjectMapper mapper = new ObjectMapper();
Produit myProduit = new Produit();
String myJSON = mapper.writeValueAsString(myProduit);

this.mockMvc.perform(
    put("/produit")
    .contentType(MediaType.APPLICATION_JSON).content(myJSON)
)
.andExpect(status().isOk());
```

# MOCK — MOCK MVC

Vérifier que l'adresse `"/produit"` (GET) retourne un flux JSON

```
this.mockMvc.perform(get("/produit"))  
    .andExpect(status().isOk())  
    .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8));
```

Vérifier que l'adresse `"/produit"` (GET) retourne

- Un flux JSON
- Contenant 6 éléments
- Dont le 1<sup>er</sup> a l'id == 1

```
this.mockMvc.perform(get("/produit"))  
    .andExpect(status().isOk())  
    .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))  
    .andExpect(jsonPath("$", hasSize(6)))  
    .andExpect(jsonPath("$[0].id", org.hamcrest.CoreMatchers.is(1))));
```

# MOCK — MOCK MVC

Il faut ajouter 2 dépendances supplémentaires

- **hamcrest-library**
- **json-path**



# EXERCICE

Reprendre le projet « REST Produits »

Rédiger un test pour

- Ajouter un produit
- Afficher la liste des produits
- Modifier un produit
- Supprimer un produit