Tester vos apps Spring Boot

Sous toutes les coutures

Daniel Garnier-Moiroux

Devoxx France, 2025-04-18

Daniel Garnier-Moiroux

Software Engineer

- Spring
- @garnier.wf
- https://garnier.wf/
- github.com/Kehrlann/
- contact@garnier.wf

Avertissement

Cette présentation ne parle pas des **tests en général**, comment les organiser, quelle architecture utiliser…

Cherchcez "Devoxx tests", trier par "nombre de vues" :)

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

Spring Boot Testing

1. 🕸️ `@SpringBootTest` **et tests web**
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

`@SpringBootTest` et tests web

- `WebEnvironment.MOCK` environnement servlet "mock"
  - Suffisant dans la plupart des cas

- `WebEnvironment.RANDOM_PORT` serveur web complet (e.g. Tomcat)
  - Utile pour tests en profondeur (e.g. sérialisation sessions)
  - Utile pour débugguer
  - `@LocalServerPort`, `WebTestClient` ou `TestRestTemplate`

`@SpringBootTest` et tests web

- `MockMvc` ou `MockMvcTester` pour les tests par requête
  - (Seulement en mode `MOCK`)
- HtmlUnit's `WebClient` pour tests browser (légers)
- Selenium / WebDriver pour un browser complet
- Tests d'intégration avec des outils adaptés, typiquement Javascript (Playwright, Cypress)

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` **et mise en cache**
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

🐌 `TestContext` et mise en cache

- `@SpringBootTest` sont lents (~ secondes)
- Éviter `@ActiveProfiles`, `@MockitoBean`, `@DirtiesContext` ...
- Réutiliser le context qui a été mis en cache, pour éviter de payer le prix de démarrage à chaque test

🐌 `TestContext` et mise en cache

Après la pub:

- Test slices:
  - `@WebMvcTest`, `@DataJpaTest`, ...

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 **Boîte à outils des tests!**
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

🧰 Boîte à outils des tests!

- `assertJ` "fluent assertions"
  - `MockMvcTester` : `MockMvc` avec assertJ (Boot >= 3.4)
  - Assertions assertJ custom 🤯
- `@OutputCaptureExtension` : capturer toutes les sorties stdout/stderr

🧰 Boîte à outils des tests!

Vous pouvez:

- Demander à un LLM

Vous *devriez*:

- Lire la doc *(wow such old very 2023)*

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 **Slice tests: ApplicationContext, version "ultra-léger**
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

🍕 Slice tests: ApplicationContext, version "ultra-léger"

- Les slice tests sont légers, focalisés
- Besoin de mocks or de beans de remplacement
- `@WebMvcTest`, `@DataJpaTest`
  - Slices custom, avec `@SpringBootTest(classes = ...)`

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 **Testcontainers**
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

Testcontainers

- Setup simple avec @ `start.spring.io`
- Configuration automatique avec `@ServiceConnection`
  - Postgres, Otlp..., Redis, Rabbit...
- Configuration dynamique avec `DynamicPropertyRegistrar`
- Pattern singleton avec containers `static`

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 **Boîte à outils des tests!**
7. ⚙️ Test des `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

🧰 Boîte à outils des tests!

- `awaitility` pour attendre et "sonder"
  - Ne jamais, jamais utiliser `Thread.sleep` *
- Apprenez à configurer Mockito `mockito`

* sauf quand, vous savez, It Depends™

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ **Test des** `@ConfigurationProperties`
8. 🔐 Tests de la couche sécurité

⚙️ Test des `@ConfigurationProperties`

- Construire des objets property les valider
  - `Validation.buildDefaultValidatorFactory().getValidator()`
  - Utiliser du YAML?
- Pour les tests d'intégration:
  - Utiliser `@SpringBootTest(classes = { ... })`
  - Pour les exceptions, utiliser `SpringApplicationBuilder#run`

Spring Boot Testing

1. 🕸️ `@SpringBootTest` et tests web
2. 🐌 `TestContext` et mise en cache
3. 🧰 Boîte à outils des tests!
4. 🍕 Slice tests: ApplicationContext, version "ultra-léger"
5. 📦 Testcontainers
6. 🧰 Boîte à outils des tests!
7. ⚙️ Test des `@ConfigurationProperties`
8. **🔐 Tests de la couche sécurité

🔐 Tests de la couche sécurité

- Compatible avec `@WebMvcTest` & `@SpringBootTest`
- `@WithMockUser` pour injecter un utilisateur
  - `@WithUserDetailsService`

- `SecurityMockMvcRequestPostProcessors` pour `MockMvc(Tester)`
  - `.csrf()`, `.opaqueToken()`, `.oidcLogin()` ...
  - ... et plus encore!

References

https://github.com/Kehrlann/spring-boot-testing

Faites-moi signe!

- @garnier.wf
- https://garnier.wf/
- contact@garnier.wf


Merci 😊
VMware Tanzu
VOXXED DAYS
ZURICH