Adopter Springboot = KISS = Facilité/Rapidité = Cloud = Sécurité

Table of Contents

Spring Framework . . . 1
Springboot Framework = leger = KISS = Standalone app . . . 2
Les fonctionnalités de SpringBoot . . . 2
1. RestController . . . 2
2. application.yml . . . 2
3. Tomcat ou Jety embarqué . . . 3
4. Boite à outils pour intégration . . . 3
5. Springboot et Docker . . . 3
4. Environnemet Prod: . . . 3
5. Cloud . . . 3
Spring Springboot, les différences . . . 4
1. Injection de dependance (IOC) utilisant XML . . . 4
2. Injection de dependance (IOC) en utilisant @Configuration . . . 5
3. Spring & Springboot (~/ressouces/) . . . 5
4. PropertyPlaceHolder . . . 5
5. Scopes (application-context.xml) . . . 5
6. Definir des profils: Profil en Spring (@Profil) . . . 6

Spring Framework

```
Spring est un framework Java très populaire pour la construction d'applications Web et
d'entreprise. Contrairement à de nombreux autres cadres, qui se concentrentsur un seul
domaine, Spring Framework offre une grande variété de fonctionnalités répondant aux
besoins commerciaux modernes grâce à ses projets de portefeuilleLe framework Spring
offre une flexibilité pour configurer les beans de plusieurs manières telles que XML ,
Annotations et JavaConfig .
Avec le nombre de fonctionnalités augmentées, la *complexité augmente* également et la
configuration des applications Spring devient *fastidieuse* et susceptible
d'erreurs.L'équipe Spring a créé `SpringBoot` pour répondre à la complexité de la
configuration.
Mais avant de plonger dans SpringBoot, nous allons jeter un coup d'œil sur le cadre du
printemps et voir quel type de problèmes SpringBoot essaie de résoudre.
```

```
Parallèlement à Spring Framework, il existe de nombreux autres projets Spring Brothers
qui aident à créer des applications répondant aux besoins des entreprises modernes:
```

- Spring Data: simplifie l'accès aux données des magasins de données relationnels et NoSQL.
- Spring Batch: offre un puissant cadre de traitement par lots.
- Spring Security: cadre de sécurité robuste pour sécuriser les applications.
- Spring Social: prend en charge l'intégration avec des sites de réseaux sociaux comme Facebook, Twitter, LinkedIn, GitHub, etc.
- Spring Integration: une implémentation de Patterns d'intégration d'entreprise pour faciliter l'intégration avec d'autres applications d'entreprise utilisant des messagerie légère et des adaptateurs déclaratifs.

Il existe de nombreux autres projets intéressants abordant divers autres besoins de développement d'applications modernes. Pour plus d'informations, consultez `http://spring.io/projects`. Dans les premiers jours, Spring Framework fournit une approche basée sur XML pour configurer les beans. Plus tard Spring a introduit des DSL basées sur XML,desannotations et des approches basées sur JavaConfig pour configurer des beans Permettez-nous de regarder rapidement comment ressemble chacun de ces styles deconfiguration.

Springboot Framework = leger = KISS = Standalone app

Spring Boot a été conçu pour rendre la vie du développeur plus simple et lui permettre de se concentrer sur le coeur de l'application et non pas sur les aspects annexes : configuration, tests, sécurité, déploiement…

Les fonctionnalités de SpringBoot

1. RestController

```
"Nouveauté" dans Springboot. Permet de representer le retour d'une requête sur le
browser.
```

2. application.yml

Plus de script de gestion d'environnement (`@Component`, `@Configuration` et possibilité de déclarer dans `application.yml` ou `application.properties`).

`@EnableAutoConfiguration`. Cette annotation va alors s'appuyer sur l'ensemble des dépendances de l'application (MVC, Tomcat, …) pour configurer l'application.

3. Tomcat ou Jety embarqué

Simplicité de déploiement (le seul prérequis est Java). Cloud-ready.

4. Boite à outils pour intégration

- Bases relationnelles : JDBC, JPA, JdbcTemplate
- Bases NoSQL : Redis, MongoDB, ElasticSearch …
- Messaging : JMS

5. Springboot et Docker

```
FROM java:8u45
MAINTAINER Patty Randriambololona "patty.randria8@gmail.com"
ADD springboot-1.0-SNAPSHOT.jar app.jar
ENTRYPOINT [ "java", "-Dspring.profiles.active=test", "-jar", "/app.jar" ]
```

```
$ docker build -t ellixo/springboot .
$ docker run -d -p=9292:9292 ellixo/springboot
```

4. Environnemet Prod:

Pour aller plus loin, il est possible d'installer le module Actuator qui fournit de nombreuses fonctionnalités d'administation système (via notamment une API Rest): health : fournit des données permettant de vérifier l'état de l'application (UP/DOWN, état disque, état systèmes externes …) metrics : fournit des métriques processus (threads, CPU, mémoire …) trace : fournit les informations des dernières connexions HTTP applicatives … Libre à vous ensuite de connecter ce module à l'outil de monitoring du système d'information (Graphite, Promotheus …)

```
.Exemple : API Health
----
$ curl http://user:password@localhost:9292/health
{"status":"UP","diskSpace":{"status":"UP","free":169718296576,"threshold":10485760},"m
ongo":{"status":"UP","version":"3.0.2"}}
----
```

5. Cloud

Encore une fois, le fait qu'une application Spring Boot embarque son propre containeur (Tomcat ou Jetty par défaut donc) simplifie un déploiement cloud. Pour démontrer le rapidité du processus, j'ai décidé d'exposer le déploiement sous la plateforme Cloud Foundry de Pivotal (à tout seigneur, tout honneur) : Une fois votre compte Pivotal Web Services créé et le client associé installé, la seule

commande à exécuter sur votre environnement est :

`cf push springboot-demo -p springboot-1.0-SNAPSHOT.jar`

```
Uploading app files from: springboot-1.0-SNAPSHOT.jar
Uploading 623.8K, 96 files
Done uploading
OK
```

Par défaut, **Cloud Foundry** prend en compte le profil "cloud" ; pour autant, il est possible d'activer un autre profil en positionnant la variable d'environnement JAVA_OPTS (exemple : -Dspring.profiles.active=test). L'application est alors disponible via l'URL nom_app.cfapps.io (http://springboot-demo.cfapps.io ici)

Spring Springboot, les différences

1. Injection de dependance (IOC) utilisant XML

TIP	@ComponentScan, @Component, @Autowired, @Qualifier

Dans Spring @ComponentScan representte la configuration de base de `Srping` dans `Spring` l'annotation vamapper `@ComponentScan(basePackages ={"spring.bean"}` ce qui est remplacer par `application.yml ou.properties` dans `Springboot`.

- @ComponentScan(basePackages ={"spring.bean"} recherche des @Bean pour être utilisé dans la classe `main`.
  - @Component : est une classe que va chercher componentScan. Le @component seront des "@bean" dans le `main`
    - @Autowired : injection dans un service.
    - @Qualifier : moyen pour donner un nom a un component (le nom de la class) qui va être son ID
- pacquet `org.springframework.stereotype` est composé de
  - @Component: une classe candidat dans le but est de scanner les composant de Spring
  - @Controller: utilisé dans une application Spring MVC
  - @Repository: utilisé pour définir une classe DAO (Data Access Object)
  - @Service: utilisé pour définir le bisiness Services

IMPORTANT	Meta-Annotations sont des méthodes de classe mère exemple @Transactionnal

`@Autowired`: l'annotation fait de l'injection de dependance, elle peut être appelé sur un constructeur, une classe, une methode.

exemple sur une classe DAO - classe de configuration de la base de données (DAO) coit être annoté pas @Component — l'annotation `@Autowird` peut être appelé dans cette classe, sur les constructeurs et les méthodes

Question: comment on fait si on veut injecter dans un champ private ?

- on utilise `@Autowird` ou `@Ressource`

Question: qu'est ce qu'un prefix?

- Est utile pour mapper et accéder à une ressource. Dans `Spring` on peut utiliser des préfix sur le `classpath`,`file`,`http`,`(none)`

2. Injection de dependance (IOC) en utilisant @Configuration

`@Configuration`: permet d'eviter de rediger un `fichier xml`. La configuration peut directement être écrit dans le main annoté @Bean. exemple:

@Bean public DataBaseService data(){ DataBaseService data = new DataBaseService(); return data; } ---

3. Spring & Springboot (~/ressouces/)

- abstract=true bean sert a regrouper des propiété de class dans <bean></bean> Question: pourquoi on ne peut pas mettre @Bean sur une methodes final
- toutes les classes annotées @Configuration utilisé CGLIB de JavaConfig, en consequence toutes ses occurences ne peuvet être marquées en final ou private, donc @Bean n'ai pas pas accepté non plus en raison de CGLIB.

4. PropertyPlaceHolder

C'est un moyen de resoudre `${}` dans un fichier XML ou system ou variable d'environnement.

```
1) déclarer en tant que @Bean dans main
2) creer un fichier de configuration.properties
3) mettre les variables dans ce fichier
4) annoter la classe l'utilisant avec l'annotation @PropertySource
```

5. Scopes (application-context.xml)

Les scopes sont définis das la balise `<bean .. scope="xxx">` soit par singleton, prototype, session, request, global_session, application.

- Singleton: est une et une seule instance d'un objet dans le Springframework

- Prototype: exemple sur le cas de login, au lieu de redefinir plusieurs fois une configuration, on va écrire dans application.xml la configuration et on l'annote @Prototype
- Session: environment web dont une instance va être créer à chaque session HTTP
- Request: environment web dont une instance va être créer à chaque requette
- Global session:

6. Definir des profils: Profil en Spring (@Profil)

`@Profile` peut être defini dans un fichier XML ou dans une class. Ca consiste a definir un profil basé sur une configuration. exemple: On a deux @bean methode de connexion à une base de données.

```
1) on peut definir nommé un profil dessus
2) on peut l'appelé sur une `class` avec @ActiveProfil("nomProfil")
```