Spring Boot

https://fr.wikipedia.org/wiki/Framework

https://fr.wikipedia.org/wiki/Spring_(framework)

Spring boot découlent de Spring (apparu en 2003)
EJB -> conteneurs lourds et EBJ très lourds à mettre en place
EJB3 sont apparues en 2005

Spring boot est basé sur toutes les fonctionnalités par défaut du Spring.

C'est un Framework conçu pour simplifier le démarrage et le développement de nouvelles applications Java.
Il propose une approche simplifiée de la configuration, qui permet d'éviter aux développeurs de redéfinir la même configuration à plusieurs endroits du code.
Spring boot adapté pour Docker, micro services, machine virtuelle car il embarque tout.

Spring Boot ne génère pas de code ni n'apporte de modifications à vos fichiers. Au lieu de cela, lorsque vous démarrez votre application, Spring Boot connecte dynamiquement les beans et les paramètres et les applique au contexte de votre application : injection de dépendances et inversion de contrôle, introspection avec l'exploration du classpath

Voici quelques-unes des fonctionnalités de Spring Boot :

- ✓ Dépendances de type « starter » pour simplifier la construction et la configuration de l'application
- ✓ Serveur intégré pour éviter la complexité lors du déploiement d'applications
- ✓ Métriques, vérification et configuration externalisée
- ✓ Configuration automatique

Il est basé sur :

- ✓ Les annotations,
- ✓ Les injections de dépendances
- ✓ L'inversion de contrôle
  https://fr.wikipedia.org/wiki/Inversion_de_controle

Spring Boot

2 fonctionnalités principales :

- L'auto-configuration : réflexivité (introspection) : va scanner toutes les classes et les dépendances du projet en explorant le classpath

- Les starters : « méga » dépendances. Pas besoin de mettre de version

Il contient :
- Un conteneur : Tomcat embarqué
- Un moteur de template, par exemple Thymeleaf (XML/XHTML/HTML5)

L'annotation de l'application @SpringBootApplication : est composé de :

- @Configuration : Marque la classe comme source de définitions de bean pour le contexte d'application.
- `@EnableAutoConfiguration` : Indique à Spring Boot d' ajouter des beans en fonction des paramètres du classpath, d'autres beans et de divers paramètres de propriété. Par exemple, si `spring-webmvc se` trouve dans le classpath, cette annotation marque l'application en tant qu'application Web et active des comportements clés, tels que la configuration d'un fichier `DispatcherServlet`.
- `@ComponentScan` : Indique à Spring de rechercher d'autres composants, configurations et services

Ficher de application.properties
- Datasource
- Logs
- Sauvegarde de la session
- Port du server
- …

Exécution : spring-boot:run

Lancement de l'appli : localhost:numéro de port :8080 si non spécifié dans le fichier application.properties
Arrêt du serveur : Ctrl-C

Spring Boot

Compléments au cours :

- Les logs avec log4j2 :
  - Dans pom.xml

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-log4j2</artifactId>
</dependency>
```

  - Dans les classes

```
private static final Logger LOGGER =
Logger.getLogger(NomDeClasse.class.getName());
```

  - Config dans application.properties

```
logging.level.com.javadevjournal = INFO
logging.level.org.springframework.web = INFO
# logging.level.web = DEBUG
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n
logging.pattern.file= "%d{yyyy-MM-dd } [%thread] %-5level %logger{36} - %msg%n"
```

Exemple d'emplacement des fichiers :

```
logging.file.name= logs/logfile.log
```

- Spring boot JPA – Hibernate

https://www.baeldung.com/spring-boot-hibernate