Spring Boot : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

Spring Boot

Spring MVC

- un projet de **Spring Framework**
- basé sur l'**API Servlet** de **Java JEE**
- permettant de simplifier le développement d'applications web en respectant le patron de conception **MVC 2**

Spring Boot

Spring MVC

- un projet de **Spring Framework**
- basé sur l'**API Servlet** de **Java JEE**
- permettant de simplifier le développement d'applications web en respectant le patron de conception **MVC 2**

Inconvénients

- Dépendances explicites ⇒ risque d'incompatibilité entre les versions
- Configuration manuelle et complexe : **JPA**, sécurité, contrôleurs, vues...
- Démarrage lent : nécessité de charger et d'initialiser plusieurs composants de configuration

Spring Boot

Spring Boot ?

- Construit sur le dessus (sur-couche) de **Spring Framework** (y compris **Spring MVC**
- Considéré aussi comme un framework
- Simplifiant le processus de configuration et de développement des applications basées sur **Spring Framework**.
- Utilisant les composants et modules de **Spring Framework** pour fournir des fonctionnalités clés telles que
  - l'auto-configuration,
  - l'intégration de serveur embarqué,
  - la gestion des dépendances simplifiée,
  - la gestion des propriétés de l'application...

Spring Boot

Comment ?

- Démarreur (`starter`)
  - dépendance dont l'`artifactId` commence par `spring-boot-starter-*`
  - regroupant un paquet de dépendance (**Spring** ou autre)
  - conçu pour une fonctionnalité spécifique
  - facilitant la gestion des dépendances et réduisant les conflits potentiels
- Auto-configuration : permet de configurer automatiquement le projet à partir de
  - dépendances détectées dans le projet (**Maven**, **Gradle**...)
  - certaines valeurs par défaut (emplacement des vues, lecture de package...)
  - conventions (règles de nommage...)
- Démarrage rapide : grâce à l'**auto-configuration**

Spring Boot

Exemple, pour créer un projet web, il faut ajouter la dépendance `Maven` suivante :

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

Spring Boot

Exemple, pour créer un projet web, il faut ajouter la dépendance `Maven` suivante :

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


Pour consulter la liste des starters, aller sur

https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html

Spring Boot

La dépendance `spring-boot-starter-web` **inclut les six dépendances suivantes :**

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

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

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

<dependency>
  <groupId>org.hibernate.validator</groupId>
  <artifactId>hibernate-validator</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
</dependency>
```

Spring Boot

La dépendance `spring-boot-starter-web` permet donc de créer un projet web contenant :

- un serveur **Apache Tomcat**
- **Spring Core**
- **Spring Web MVC**
- **Jackson** pour les données au format **JSON**
- **Logback** et **Log4j**
- ...

Spring Boot

Création de projet Spring Boot

- Aller dans `File > New > Other`
- Chercher `Spring`, dans `Spring Boot` sélectionner `Spring Starter Project` et cliquer sur `Next >`
- Saisir
  - `cours-spring-boot` dans `Name`,
  - `com.example` dans `Group`,
  - `cours-spring-boot` dans `Artifact`
  - `com.example.demo` dans `Package`
- Cliquer sur `Next >`
- Chercher et cocher la case correspondante à `Spring Web`, choisir la dernière version srable puis cliquer sur `Next >`
- Valider en cliquant sur `Finish`

Spring Boot

Pourquoi a-t-on coché la case `Spring Web` à la création du projet ?

- pour ajouter la dépendance `spring-boot-starter-web`

Contenu de la section `dependencies` de `pom.xml`

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Spring Boot

Remarques

- Le package contenant le point d'entrée de notre application (la classe contenant le `public static void main`) est `com.example.demo`
- Tous les autres packages `dao`, `model`... doivent être dans le package `demo`.

Spring Boot

Le point de démarrage de l'application

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FirstSpringBootApplication {

  public static void main(String[] args) {
    SpringApplication.run(FirstSpringBootApplication.class, args);
  }
}
```

Spring Boot

Explication

- `SpringApplication` : la classe de démarrage d'une application **Spring** et qui va créer une instance de la classe `ApplicationContext`
- `ApplicationContext` : l'interface centrale d'une application **Spring** permettant de fournir des informations de configuration à l'application.
- `@SpringBootApplication` : englobe les 3 annotations suivantes :
  - `@Configuration` : fait partie du noyau de **Spring Framework** et indique que la classe annoté peut contenir des méthodes annotées par `@Bean`. Ainsi,**Spring Container** peut traiter la classe et générer des beans qui seront utilisés par l'application.
  - `@EnableAutoConfiguration` : permet, au démarrage de **Spring**, de générer automatiquement les configurations nécessaires en fonction des dépendances ajoutées.
  - `@ComponentScan` : permet de scanner les package contenant des composants

Spring Boot

Pour exécuter

- Faire un clic droit sur le projet et aller dans `Run As` et cliquer sur `Spring Boot App`
- Ou faire un clic droit sur **la classe** `FirstSpringBootApplication` dans `Package Explorer`, aller dans `Run As` et cliquer sur `Spring Boot Application`

Spring Boot

Pour exécuter

- Faire un clic droit sur le projet et aller dans `Run As` et cliquer sur `Spring Boot App`
- Ou faire un clic droit sur **la classe** `FirstSpringBootApplication` dans `Package Explorer`, aller dans `Run As` et cliquer sur `Spring Boot Application`

La console nous indique

```
Tomcat started on port(s): 8080 (http) with context path ''
```

Spring Boot

Pour exécuter

- Faire un clic droit sur le projet et aller dans `Run As` et cliquer sur `Spring Boot App`
- Ou faire un clic droit sur **la classe** `FirstSpringBootApplication` dans `Package Explorer`, aller dans `Run As` et cliquer sur `Spring Boot Application`

La console nous indique

```
Tomcat started on port(s): 8080 (http) with context path ''
```

Pour tester

Aller à `http://localhost:8080/`

Spring Boot

Résultat : message d'erreur

- On a créé un projet web, mais on n'a aucune page **HTML** ni **JSP**.
- **Spring Boot**, comme **Spring MVC**, implémente le patron de conception **MVC 2**, donc il nous faut au moins un contrôleur et une vue.

Spring Boot

Rappel

Par défaut, le serveur **Apache Tomcat** est embarqué dans `spring-boot-starter-web`

Spring Boot

Rappel

Par défaut, le serveur **Apache Tomcat** est embarqué dans `spring-boot-starter-web`

Question

Et si nous souhaitions utiliser un autre serveur (**Jetty** par exemple) ?

Spring Boot

Rappel

Par défaut, le serveur **Apache Tomcat** est embarqué dans `spring-boot-starter-web`

Question

Et si nous souhaitions utiliser un autre serveur (**Jetty** par exemple) ?

Réponse : il faudra

- exclure **Apache Tomcat** de `spring-boot-starter-web`
- inclure le starter de **Jetty**

Spring Boot

Pour exclure `Apache Tomcat` **de** `spring-boot-starter-web` **dans** `pom.xml`

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <!-- Exclude the Tomcat dependency -->
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Spring Boot

Pour exclure `Apache Tomcat` **de** `spring-boot-starter-web` **dans** `pom.xml`

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <!-- Exclude the Tomcat dependency -->
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```


Pour inclure le starter de `Jetty`

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

Spring Boot

Fichiers de configuration

- Permettent de simplifier la configuration d'un projet **Spring Boot**
- Remplaçant la création d'un certain nombre de beans (pas tous)

Spring Boot

Fichiers de configuration

- Permettent de simplifier la configuration d'un projet **Spring Boot**
- Remplaçant la création d'un certain nombre de beans (pas tous)

Pour la configuration : deux formats (fichiers) possibles

- `application.properties` (par défaut)
- `application.yaml`

Spring Boot

Format utilisé dans le fichier `application.properties`

```
propriété1.sous-propriété11=valeur11
propriété1.sous-propriété12=valeur12
propriété2.sous-propriété21=valeur21
```

Spring Boot

Pour modifier le numéro du port et le `context-path` du serveur, on ajoute les deux propriétés suivantes dans `application.properties`

```
server.port=3000
server.servlet.context-path=/boot
```

Spring Boot

Pour modifier le numéro du port et le `context-path` du serveur, on ajoute les deux propriétés suivantes dans `application.properties`

```
server.port=3000
server.servlet.context-path=/boot
```


La console nous indique

```
Tomcat started on port(s): 3000 (http) with context path '/boot'
```

Spring Boot

Pour modifier le numéro du port et le `context-path` du serveur, on ajoute les deux propriétés suivantes dans `application.properties`

```
server.port=3000
server.servlet.context-path=/boot
```


La console nous indique

```
Tomcat started on port(s): 3000 (http) with context path '/boot'
```

Pour tester

Aller à `http://localhost:3000/boot`

Spring Boot

Liste de propriétés disponibles pour `application.properties`

`https://docs.spring.io/spring-boot/appendix/application-properties/index.html`

Spring Boot

Question

Est-il recommandé de définir un `context-path` pour toute application web créée avec **Spring Boot** ?

Spring Boot

Question

Est-il recommandé de définir un `context-path` pour toute application web créée avec **Spring Boot** ?

Réponse : oui pour les raisons suivantes

- **Déploiement multiple** : permet de déployer plusieurs applications **Spring Boot** sur le même serveur.
- **Gestion des versions** : peut aider à distinguer les différentes versions et à maintenir une séparation claire entre elles.
- ...

Spring Boot

Format utilisé dans le fichier `application.yaml`

```
propriété1:
  sous-propriété11: valeur11
  sous-propriété12: valeur12
propriété2:
  sous-propriété21: valeur21
```

Spring Boot

Format utilisé dans le fichier `application.yaml`

```
propriété1:
  sous-propriété11: valeur11
  sous-propriété12: valeur12
propriété2:
  sous-propriété21: valeur21
```


N'oublions pas de commenter (ou supprimer) le contenu de `application.properties`

Spring Boot

Pour modifier le numéro du port et le `context-path` du serveur, on ajoute les deux propriétés suivantes dans `application.properties`

```
server:
  port: 5000
  servlet:
    context-path: /boot
```

Spring Boot

Pour modifier le numéro du port et le `context-path` du serveur, on ajoute les deux propriétés suivantes dans `application.properties`

```
server:
  port: 5000
  servlet:
    context-path: /boot
```

La console nous indique

```
Tomcat started on port(s): 5000 (http) with context path '/boot'
```

Spring Boot

Pour modifier le numéro du port et le `context-path` du serveur, on ajoute les deux propriétés suivantes dans `application.properties`

```
server:
  port: 5000
  servlet:
    context-path: /boot
```

La console nous indique

```
Tomcat started on port(s): 5000 (http) with context path '/boot'
```

Pour tester

Aller à `http://localhost:5000/boot`

Spring Boot

Question

Peut-on utiliser `application.properties` et `application.yaml` ensemble dans le même projet ?

Spring Boot

Question

Peut-on utiliser `application.properties` et `application.yaml` ensemble dans le même projet ?

Réponse : oui

- `application.yaml` sera chargé en premier
- Une propriété présente dans les deux fichiers aura au final la valeur définie dans `application.properties`

Spring Boot

Remarque

Dans `application.properties` et `application.yaml`, on peut définir nos propriétés personnalisées (à voir dans le chapitre suivant).