

# Spring BOOT

## SOMMAIRE

|   |          |
|---|----------|
| <b>1- Introduction.....</b>               | <b>3</b> |
| <b>2- Ce que propose Spring Boot.....</b> | <b>3</b> |
| <b>2. L'auto-configuration .....</b>      | <b>3</b> |
| <b>2.1. Les Starters .....</b>            | <b>4</b> |

## 1- Introduction

Si vous êtes amenés à travailler sur une application développée autour de l'architecture Microservices en Java, vous aurez tôt ou tard affaire à Spring Boot. Dans ce document, nous allons donc prendre le temps de comprendre ce que fait ce framework et surtout lever l'ambiguïté sur la différence avec les autres framework Spring, qui à première vue font la même chose.

Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar, totalement autonome. Ce qui nous intéresse particulièrement, puisque nous essayons de développer des Microservices !

## 2- Ce que propose Spring Boot

### 2.1. L'auto-configuration

Cette fonctionnalité est la plus importante de Spring Boot. Elle permet de **configurer automatiquement** votre application à partir des *jar* trouvés dans votre Classpath. En d'autres termes, si vous avez importé des dépendances, Spring Boot ira consulter cette liste puis produira la configuration nécessaire pour que tout fonctionne correctement.

Prenons l'exemple d'une application web dans laquelle vous avez les dépendances : Hibernate et Spring MVC. Normalement, vous devez créer les fichiers de configuration suivants :


- appconfig-mvc.xml
- web.xml
- persitence.xml

Comme vous ne connaissez pas nécessairement la syntaxe de ces fichiers par cœur, il vous faut consulter la documentation ou vous inspirer d'un ancien projet. Vous devez ensuite écrire le code Java qui permet de lier ces fichiers XML à *ApplicationContext* de Spring.

Ensuite, vous adaptez et testez, puis ré-adaptez et re-testez encore pendant un bon moment avant que tout fonctionne... Bien sûr, dès que vous faites un changement dans votre application, il ne faut pas oublier de mettre à jour tous les fichiers de configuration, puis débbugger de nouveau. Cela devient très vite très fastidieux !

Voici l'équivalent de l'ensemble de ces étapes avec Spring MVC :

```
1 @EnableAutoConfiguration
```

C'est tout !  Avec cette annotation, **Spring Boot ira scanner la liste de vos dépendances**, trouvant par exemple Hibernate. Ayant constaté que vous n'avez défini

aucun autre *datasource*, il **créera la configuration nécessaire** et l'ajoutera à *ApplicationContext*.

Bien entendu, vous pouvez très facilement personnaliser ces configurations, en créant vos *Beans* ou vos propres fichiers de configuration. Spring Boot utilisera alors en priorité vos paramètres.

## 2.2. Les Starters

Les starters viennent compléter l'auto-configuration et font gagner énormément de temps, notamment lorsqu'on commence le développement d'un Microservice. Un starter va apporter à votre projet un **ensemble de dépendances**, communément utilisées pour un type de projet donné. Ceci va vous permettre de créer un **"squelette" prêt à l'emploi** très rapidement.

L'autre énorme avantage est la **gestion des versions**. Plus besoin de chercher quelles versions sont compatibles puis de les ajouter une à une dans le *pom.xml* ! Il vous suffit d'ajouter une simple dépendance au starter de votre choix. Cette dépendance va alors ajouter, à son tour, les éléments dont elle dépend, avec les bonnes versions.

Prenons l'exemple où vous souhaitez créer un Microservice. En temps normal, vous aurez besoin des dépendances suivantes :

- Spring ;
- Spring MVC ;
- Jackson (pour json) ;
- Tomcat ;
- ...

Avec Spring Boot, vous allez tout simplement avoir une seule dépendance dans votre *pom.xml* :

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-web</artifactId>
4 </dependency>
```



Tous les starters de Spring Boot sont au format *spring-boot-starter-NOM\_DU\_STARTER*

Ce starter va charger les dépendances présentes dans le *pom* suivant :

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
  4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <parent>
5     <groupId>org.springframework.boot</groupId>
6     <artifactId>spring-boot-starters</artifactId>
7     <version>1.5.9.RELEASE</version>
8   </parent>
9   <artifactId>spring-boot-starter-web</artifactId>
10  <name>Spring Boot Web Starter</name>
11  <description>Starter for building web, including RESTful, applications using Spring
12    MVC. Uses Tomcat as the default embedded container</description>
13  <url>http://projects.spring.io/spring-boot/</url>
14  <organization>
15    <name>Pivotal Software, Inc.</name>
16    <url>http://www.spring.io</url>
17  </organization>
18  <properties>
19    <main.basedir>${basedir}/../..</main.basedir>
20  </properties>
21  <dependencies>
22    <dependency>
23      <groupId>org.springframework.boot</groupId>
24      <artifactId>spring-boot-starter</artifactId>
25    </dependency>
26    <dependency>
27      <groupId>org.springframework.boot</groupId>
28      <artifactId>spring-boot-starter-tomcat</artifactId>
29    </dependency>
30    <dependency>
31      <groupId>org.hibernate</groupId>
32      <artifactId>hibernate-validator</artifactId>
33    </dependency>
34    <dependency>
35      <groupId>com.fasterxml.jackson.core</groupId>
36      <artifactId>jackson-databind</artifactId>
37    </dependency>
38    <dependency>
39      <groupId>org.springframework</groupId>
40      <artifactId>spring-web</artifactId>
41    </dependency>
42    <dependency>
43      <groupId>org.springframework</groupId>
44      <artifactId>spring-webmvc</artifactId>
45    </dependency>
46  </dependencies>
47 </project>

```

Comme vous pouvez le voir, ce starter Spring Boot pour application web met en place la configuration des éléments suivants : tomcat, hibernate-validator, jackson, spring MVC.



D'accord, mais cela ne m'explique pas comment il devine les versions ?

Voici la réponse :

```
1 <parent>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-parent</artifactId>
4   <version>1.5.9.RELEASE</version>
5 </parent>
```

html

Ce tag, ajouté en haut du pom.xml, permet à votre pom d'hériter des propriétés d'un autre pom que vous [trouverez](https://github.com/spring-projects/spring-boot/blob/master/spring-boot-project/spring-boot-starters/spring-boot-starter-parent/pom.xml) au lien suivant : <https://github.com/spring-projects/spring-boot/blob/master/spring-boot-project/spring-boot-starters/spring-boot-starter-parent/pom.xml> (qui lui-même hérite d'un autre pom : *spring-boot-dependencies*). Il permet de définir principalement :

La version de Java à utiliser. Dans ce cas, c'est la 1.6.

Une liste complète des versions des dépendances prises en charge, ce qui permet d'ajouter des dépendances sans indiquer leur version comme dans le *pom.xml* du starter *spring-boot-starter-web* vu plus haut. Vous allez donc pouvoir ajouter les dépendances de votre choix, sans vous soucier des versions.

Bien entendu, *spring-boot-starter-web* n'est pas le seul starter disponible. Selon ce que vous comptez développer, vous trouverez pratiquement toujours un **starter adéquat**. Voici quelques exemples :

spring-boot-starter-mail : pour les applications et services d'envoi de mails.

spring-boot-starter-thymeleaf : si vous souhaitez créer une application qui offre une interface utilisateur en utilisant le moteur de template thymeleaf.

spring-boot-starter-web-services : pour les services plus classiques utilisant SOAP.

Tous les starters de Spring Boot sont définis au niveau du site officiel :

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using-boot-starter>

La liste complète est (53 en total) :



| Name   | Description  |
|--|--|
| <code>spring-boot-starter</code>                         | Core starter, including auto-configuration support, logging and YAML   |
| <code>spring-boot-starter-activemq</code>                | Starter for JMS messaging using Apache ActiveMQ  |
| <code>spring-boot-starter-amqp</code>                    | Starter for using Spring AMQP and Rabbit MQ  |
| <code>spring-boot-starter-aop</code>                     | Starter for aspect-oriented programming with Spring AOP and AspectJ  |
| <code>spring-boot-starter-artemis</code>                 | Starter for JMS messaging using Apache Artemis   |
| <code>spring-boot-starter-batch</code>                   | Starter for using Spring Batch   |
| <code>spring-boot-starter-cache</code>                   | Starter for using Spring Framework's caching support   |
| <code>spring-boot-starter-cloud-connectors</code>        | Starter for using Spring Cloud Connectors which simplifies connecting to services in cloud platforms like Cloud Foundry and Heroku                         |
| <code>spring-boot-starter-data-cassandra</code>          | Starter for using Cassandra distributed database and Spring Data Cassandra   |
| <code>spring-boot-starter-data-cassandra-reactive</code> | Starter for using Cassandra distributed database and Spring Data Cassandra Reactive  |
| <code>spring-boot-starter-data-couchbase</code>          | Starter for using Couchbase document-oriented database and Spring Data Couchbase   |
| <code>spring-boot-starter-data-couchbase-reactive</code> | Starter for using Couchbase document-oriented database and Spring Data Couchbase Reactive  |
| <code>spring-boot-starter-data-elasticsearch</code>      | Starter for using Elasticsearch search and analytics engine and Spring Data Elasticsearch  |
| <code>spring-boot-starter-data-jdbc</code>               | Starter for using Spring Data JDBC   |
| <code>spring-boot-starter-data-jpa</code>                | Starter for using Spring Data JPA with Hibernate   |
| <code>spring-boot-starter-data-ldap</code>               | Starter for using Spring Data LDAP   |
| <code>spring-boot-starter-data-mongodb</code>            | Starter for using MongoDB document-oriented database and Spring Data MongoDB   |
| <code>spring-boot-starter-data-mongodb-reactive</code>   | Starter for using MongoDB document-oriented database and Spring Data MongoDB Reactive  |
| <code>spring-boot-starter-data-neo4j</code>              | Starter for using Neo4j graph database and Spring Data Neo4j   |
| <code>spring-boot-starter-data-redis</code>              | Starter for using Redis key-value data store with Spring Data Redis and the Lettuce client   |
| <code>spring-boot-starter-data-redis-reactive</code>     | Starter for using Redis key-value data store with Spring Data Redis reactive and the Lettuce client  |
| <code>spring-boot-starter-data-rest</code>               | Starter for exposing Spring Data repositories over REST using Spring Data REST   |
| <code>spring-boot-starter-data-solr</code>               | Starter for using the Apache Solr search platform with Spring Data Solr  |
| <code>spring-boot-starter-freemarker</code>              | Starter for building MVC web applications using FreeMarker views   |
| <code>spring-boot-starter-groovy-templates</code>        | Starter for building MVC web applications using Groovy Templates views   |
| <code>spring-boot-starter-hateoas</code>                 | Starter for building hypermedia-based RESTful web application with Spring MVC and Spring   |
| <code>spring-boot-starter-integration</code>             | Starter for using Spring Integration   |
| <code>spring-boot-starter-jdbc</code>                    | Starter for using JDBC with the HikariCP connection pool   |
| <code>spring-boot-starter-jersey</code>                  | Starter for building RESTful web applications using JAX-RS and Jersey. An alternative to <a href="#">spring-boot-starter-web</a>                           |
| <code>spring-boot-starter-jooq</code>                    | Starter for using jOOQ to access SQL databases. An alternative to <a href="#">spring-boot-starter-data-jpa</a> or <a href="#">spring-boot-starter-jdbc</a> |
| <code>spring-boot-starter-json</code>                    | Starter for reading and writing json   |
| <code>spring-boot-starter-jta-atomikos</code>            | Starter for JTA transactions using Atomikos  |
| <code>spring-boot-starter-jta-bitronix</code>            | Starter for JTA transactions using Bitronix  |
| <code>spring-boot-starter-mail</code>                    | Starter for using Java Mail and Spring Framework's email sending support   |
| <code>spring-boot-starter-mustache</code>                | Starter for building web applications using Mustache views   |
| <code>spring-boot-starter-oauth2-client</code>           | Starter for using Spring Security's OAuth2/OpenID Connect client features  |
| <code>spring-boot-starter-oauth2-resource-server</code>  | Starter for using Spring Security's OAuth2 resource server features  |
| <code>spring-boot-starter-quartz</code>                  | Starter for using the Quartz scheduler   |
| <code>spring-boot-starter-security</code>                | Starter for using Spring Security  |

|  |  |
|--|--|
| <code>spring-boot-starter-test</code>          | Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito  |
| <code>spring-boot-starter-thymeleaf</code>     | Starter for building MVC web applications using Thymeleaf views  |
| <code>spring-boot-starter-validation</code>    | Starter for using Java Bean Validation with Hibernate Validator  |
| <code>spring-boot-starter-web</code>           | Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container                  |
| <code>spring-boot-starter-web-services</code>  | Starter for using Spring Web Services  |
| <code>spring-boot-starter-webflux</code>       | Starter for building WebFlux applications using Spring Framework's Reactive Web support  |
| <code>spring-boot-starter-websocket</code>     | Starter for building WebSocket applications using Spring Framework's WebSocket support   |
| <code>spring-boot-starter-actuator</code>      | Starter for using Spring Boot's Actuator which provides production ready features to help you monitor and manage your application          |
| <code>spring-boot-starter-jetty</code>         | Starter for using Jetty as the embedded servlet container. An alternative to <code>spring-boot-starter-tomcat</code>                       |
| <code>spring-boot-starter-log4j2</code>        | Starter for using Log4j2 for logging. An alternative to <code>spring-boot-starter-logging</code>   |
| <code>spring-boot-starter-logging</code>       | Starter for logging using Logback. Default logging starter   |
| <code>spring-boot-starter-reactor-netty</code> | Starter for using Reactor Netty as the embedded reactive HTTP server.  |
| <code>spring-boot-starter-tomcat</code>        | Starter for using Tomcat as the embedded servlet container. Default servlet container starter used by <code>spring-boot-starter-web</code> |
| <code>spring-boot-starter-undertow</code>      | Starter for using Undertow as the embedded servlet container. An alternative to <code>spring-boot-starter-tomcat</code>                    |

## Résumons



- Spring Boot est un framework qui permet de démarrer rapidement le développement d'applications ou services en fournissant les dépendances nécessaires et en auto-configurant celles-ci.
- Pour activer l'auto-configuration, on utilise l'annotation `@EnableAutoConfiguration`. Si vous écrivez vos propres configurations, celles-ci priment sur celles de Spring Boot.
- Les starters permettent d'importer un ensemble de dépendances selon la nature de l'application à développer afin de démarrer rapidement.