Le framework Spring

Module 2 – Introduction à Spring Boot

Objectifs

- Qu'est ce que Spring Boot ?
- Savoir créer un projet en utilisant Spring Boot

Spring Boot
Introduction

- Le problème :
  - Spring est un framework proposant de très nombreuses fonctionnalités
    - Web, Sécurité, batch, accès aux données, ...
  - Mais, sa configuration peut s'avérer complexe et fastidieuse
- La solution : Spring Boot, est un outil facilitant la configuration Spring grâce à :
  - L'auto-configuration
  - Les starters
  - L'approche client lourd

Spring est un méta-framework qui permet de gérer des aspects nombreux d'une application.
Cela a rendu le framework plus difficile à gérer au fur et à mesure des années.

Un nouvel outil
Pour simplifier cette configuration, Spring Boot propose **3 fonctionnalités principales** :

- l'auto-configuration,
  - Cette fonctionnalité permet aux applications Spring de découvrir automatiquement les librairies à utiliser. Par exemple, mettre la librairie h2 directement sur le classpath, permet de démarrer automatiquement cette base de donnée. Il en sera de même pour le serveur d'application qui sera démarré automatiquement au lancement de l'application.
  - Cela est possible parce que Spring boot a choisi à notre place les outils les plus adaptés et nous les propose par défaut. Il est toujours possible de changer ces choix par configuration.
  - De même, les composants Spring créés pourront être découverts automatiquement uniquement en les plaçant au bon endroit dans le projet. Ce scan automatique des composants proposé par Spring boot, permet là

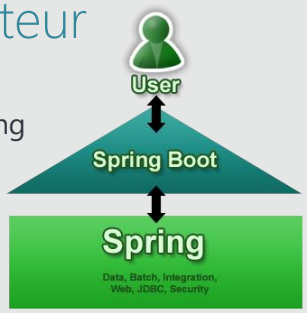
aussi de gagner du temps en configuration.

- les starters.
    - En complément de l'auto configuration, les starters sont des modules pré-définis qui répondent à des besoins de développements courants. Un starter a déjà tout son lot de dépendance configuré, ce qui allège énormément le travail.
- L'approche client lourd
    - Spring propose de démarrer les applications web comme des applications client-lourd, ce qui simplifie là encore les déploiements et rend possible les architectures micro-services.

Spring Boot

Un seul Point d'entrée pour l'utilisateur

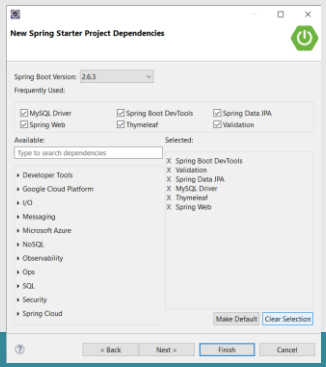
- Plus nécessaire d'intégrer tous les modules de Spring
- Les dépendances sont déjà intégrées
  - starters
- Utilisation de l'auto-configuration
  - @EnableAutoConfiguration
- Les serveurs Tomcat/Jetty sont embarqués
- Plus besoin de XML
- Application packagée en JAR orientée pour les micro-services



Spring Boot

Les Starters

- Permet de configurer les dépendances d'un projet
- Les dépendances sont organisées par catégories appelées starters
- Possibilité de travailler avec Maven ou Gradle




```
plugins {
  id 'org.springframework.boot' version '2.6.4'
  id 'io.spring.dependency-management' version '1.0.11.RELEASE'
  id 'java'
}

group = 'fr.eni'
version = '1'
sourceCompatibility = '11'

repositories {   mavenCentral() }

dependencies {
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
implementation 'org.springframework.boot:spring-boot-starter-
thymeleaf'
implementation 'org.springframework.boot:spring-boot-starter-
validation'
implementation 'org.springframework.boot:spring-boot-starter-web'
developmentOnly 'org.springframework.boot:spring-boot-devtools'
runtimeOnly 'mysql:mysql-connector-java'
testImplementation 'org.springframework.boot:spring-boot-starter-test'
}}
```

build.gradle

Les Starters représentent un **ensemble de dépendances**, communément utilisées pour un type de projet donné.
Ils permettent aussi la **gestion des versions**. Plus besoin de chercher quelles versions sont compatibles puis de les ajouter une à une dans Gradle!
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.

Spring Boot

Création automatique d'une application exécutable

```java
package fr.eni.demoWeb;

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

@SpringBootApplication
public class PremierProjetApplication {

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

}
```

```java
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan
public @interface SpringBootApplication {
```

Exécutable directement sur le tomcat intégré!

L'annotation @SpringBootApplication =
@EnableAutoConfiguration + @ComponentScan + @Configuration

- @EnableAutoConfiguration → configuration des classes de ressources, fichiers de properties, …
- @ComponentScan → Scannage des classes dans les sous packages de cette classe
- @Configuration → Création des beans automatiquement; comme celui de notre contrôleur.

L'**auto-configuration**, 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. (@EnableAutoConfiguration)

Avec cette annotation, **Spring Boot ira scanner la liste de vos dépendances**, trouvant par exemple JPA.
Ayant constaté que vous n'avez défini aucune autre *datasource*, **il créera la**

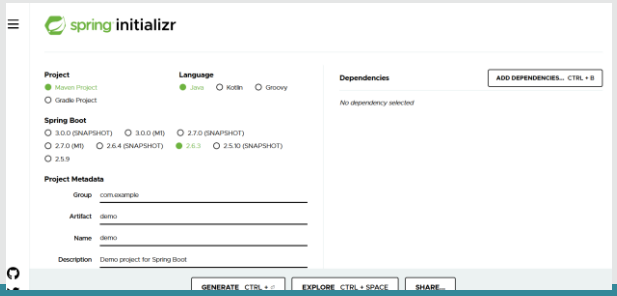
configuration nécessaire et l'ajoutera à *ApplicationContext*.

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.

Spring Boot

Outils préconisés par Spring

- Spring Tools : Plugin pour les IDEs (Eclipse, Visual Studio, Atom IDE)
- Spring initializr : https://start.spring.io/



Spring Tools : Plugin pour les IDEs (Eclipse, Visual Studio, Atom IDE)

Spring initializr : https://start.spring.io/

- Permet de créer
  - rapidement un projet Spring
  - Sélectionner les dépendances
  - Création avec Gradle ou Maven
  - Le projet créé peut être importer sous IDE.

Spring Boot

Installation de Gradle

Démonstration

Spring Boot

Installation de Spring Tools

Démonstration

Spring Boot

Première application Web avec Spring Boot

Démonstration

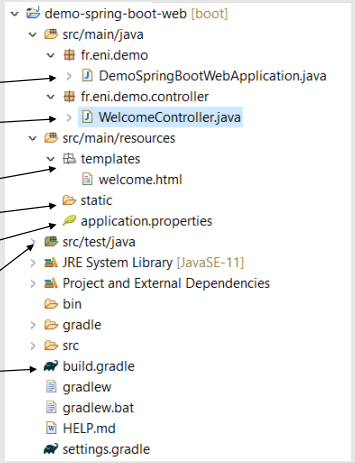
Notes :

La démonstration précédente existe sur https://spring.io/quickstart à la différence que le gestionnaire de dépendance utilisé dans notre cas est Gradle et non Maven.

Spring Boot

Architecture du projet

- La classe Main pour exécuter Spring Boot
- La classe du contrôleur de la démonstration
- Dans le répertoire « templates », ajouter les templates de page (Thymeleaf)
- Dans « static » mettre les fichiers de CSS, JS
- Fichier de propriétés pour configurer le projet
- Il y a un package de tests unitaires avec une classe par défaut
- Fichier de configuration Gradle du projet



Nous reparlerons de cette architecture plus en détail dans le module Spring Web.

Spring Boot
Conclusion

- Spring Boot permet
  - Une mise en place simplifiée d'une application Spring
  - De diminuer la configuration
- Spring Boot peut remplacer un socle d'Entreprise
  - Plus nécessaire d'avoir un serveur applicatif
  - Fabrication d'un JAR et non d'un WAR

L'utilisation de Spring Boot dépend du type de projet Spring désiré.
Au fil des modules vous verrez des starters et des configurations supplémentaires pour Spring Boot.