Spring Boot

Qu'est ce que c'est ?

- Développé en 2012 par Pivotal
- Solution de "convention plutôt que configuration"
  - réduit la complexité de la configuration de nouveaux projets Spring
- Spring Boot définit une configuration de base incluant des directives pour l’utilisation de l’infrastructure logicielle ainsi que toutes les bibliothèques de prestataires tiers pertinentes,
  - Permet de faciliter autant que possible la création de nouveaux projets.
  - Simplifie considérablement la création d’applications indépendantes prêtes pour la production
- La majorité des nouvelles applications Spring reposent en grande partie sur Spring Boot.

Caractéristiques

- Intégration directe de serveurs / conteneurs web (Apache Tomcat ou Jetty) embarqués
- Lancement d'applications web sans utiliser de fichiers war
- Configuration simplifiée de Maven et Gradle grâce à des "Starter"
- Lorsque c’est possible, configuration automatique de Spring
- Mise à disposition de capacités non fonctionnelles telles que des outils de mesure ou des configurations délocalisées

Spring Boot

Premier Projet

- Création d'un premier projet Spring Boot via Spring Initializr
- https://start.spring.io

Project

○ Maven Project ● Gradle Project

Language

● Java ○ Kotlin ○ Groovy

Spring Boot

○ 2.4.0 (SNAPSHOT) ○ 2.4.0 (M2) ○ 2.3.4 (SNAPSHOT) ○ 2.3.3
○ 2.2.10 (SNAPSHOT) ● 2.2.9 ○ 2.1.17 (SNAPSHOT) ○ 2.1.16

Project Metadata

Group	fr.formation
Artifact	premier-boot
Name	premier-boot
Description	Demo project for Spring Boot
Package name	fr.formation
Packaging	● Jar ○ War
Java	○ 14 ○ 11 ● 8

GENERATE

Spring Boot

Premier Projet

- Maven
- Gradle

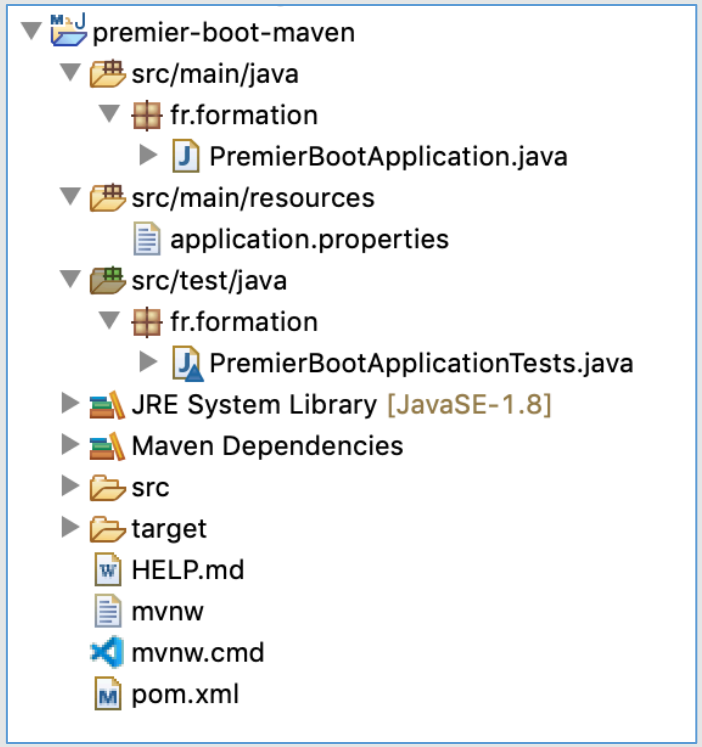
```gradle
plugins {
    id 'org.springframework.boot' version '2.2.9.RELEASE'
    id 'io.spring.dependency-management' version '1.0.9.RELEASE'
    id 'java'
}

group = 'fr.formation'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'

    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test  {
    useJUnitPlatform()
}
```

```maven
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.9.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>fr.formation</groupId>
    <artifactId>premier-boot-maven</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>premier-boot</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

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

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Spring Boot

Premier Projet

- Structure du projet Spring Boot Maven



Premier Projet

- La classe d'application

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

@SpringBootApplication
public class PremierBootApplication {

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

}
```

- Lancement

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.9.RELEASE)

2020-09-10 16:13:30.412  INFO 19425 --- [           main] fr.formation.PremierBootApplication      : Starting PremierBootApplication on iMac-de-Emmanuel-8.local with PID 19425
2020-09-10 16:13:30.415  INFO 19425 --- [           main] fr.formation.PremierBootApplication      : No active profile set, falling back to default profiles: default
2020-09-10 16:13:30.758  INFO 19425 --- [           main] fr.formation.PremierBootApplication      : Started PremierBootApplication in 0.706 seconds (JVM running for 0.988)
```

L'annotation @SpringBootApplication

- @SpringBootApplication équivaut à :
  - @Configuration
    - Permet de définir des beans injectables par Spring
  - @ComponentScan
    - Permet la découverte automatique des classes @Component et dérivées (inclu @Configuration)
    - Avec par défaut le package courant comme basePackages
  - @EnableAutoConfiguration
    - Indique à Spring Boot de "deviner" doit être configuré Spring, en fonction des dépendances de jar.
      - Par exemple, si H2 se trouve dans le classpath et qu'aucun bean de connexion à une base de données n'a été défini, Spring configurera automatiquement une base de données H2 en mémoire.
      - Peut lancer un serveur Tomcat embarqué s'il découvre qu'il est dans le classpath

Exécuter une application en ligne de commande

- Existence de 2 interfaces (au choix)
  - ApplicationRunner
  - CommandLineRunner

```java
@SpringBootApplication
public class ApplicationRunnerrApplication implements ApplicationRunner{

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

    @Override
    public void run(ApplicationArguments args) throws Exception {

        System.out.println("Hello ApplicationRunner");
    }
}
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.9.RELEASE)

2020-09-10 16:42:39.352  INFO 19700 --- [           main] f.f.CommandLineRunnerApplication
2020-09-10 16:42:39.354  INFO 19700 --- [           main] f.f.CommandLineRunnerApplication
2020-09-10 16:42:39.772  INFO 19700 --- [           main] f.f.CommandLineRunnerApplication
Hello ApplicationRunner
```

```java
@SpringBootApplication
public class CommandLineRunnerApplication implements CommandLineRunner {

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

    @Override
    public void run(String... args) throws Exception {

        System.out.println("Hello CommandLineRunner");
    }
}
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.9.RELEASE)

2020-09-10 16:42:39.352  INFO 19700 --- [           main] f.f.CommandLineRunnerApplication
2020-09-10 16:42:39.354  INFO 19700 --- [           main] f.f.CommandLineRunnerApplication
2020-09-10 16:42:39.772  INFO 19700 --- [           main] f.f.CommandLineRunnerApplication
Hello CommandLineRunner
```

Démonstration

Spring Boot

Couche de persistance

- Avec Spring initialirz

SQL

JDBC API
Database Connectivity API that defines how a client may connect and query a database.

Spring Data JPA
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Data JDBC
Persist data in SQL stores with plain JDBC using Spring Data.

IBM DB2 Driver
A JDBC driver that provides access to IBM DB2.

Apache Derby Database
An open source relational database implemented entirely in Java.

H2 Database
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

HyperSQL Database
Lightweight 100% Java SQL Database Engine.

MS SQL Server Driver
A JDBC and R2DBC driver that provides access to Microsoft SQL Server and Azure SQL Database from any Java application.

MySQL Driver
MySQL JDBC and R2DBC driver.

Oracle Driver
A JDBC driver that provides access to Oracle.

PostgreSQL Driver
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

NOSQL

Spring Data Redis (Access+Driver)
Advanced and thread-safe Java Redis client for synchronous, asynchronous, and reactive usage. Supports Cluster, Sentinel, Pipelining, Auto-Reconnect, Codecs and much more.

Spring Data Reactive Redis
Access Redis key-value data stores in a reactive fashion with Spring Data Redis.

Spring Data MongoDB
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Spring Data Reactive MongoDB
Provides asynchronous stream processing with non-blocking back pressure for MongoDB.

Spring Data Elasticsearch (Access+Driver)
A distributed, RESTful search and analytics engine with Spring Data Elasticsearch.

Spring Data for Apache Solr
Apache Solr is an open source enterprise search platform built on Apache Lucene.

Spring Data for Apache Cassandra
A free and open-source, distributed, NoSQL database management system that offers high-scalability and high-performance.

Couche de persistance

- Exemple avec Spring Data JPA et H2 Database

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

<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

```
package fr.formation;

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

@SpringBootApplication
public class BootDataJpaApplication {

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

}
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.9.RELEASE)

2020-09-12 14:18:24.367  INFO 41024 --- [           main] fr.formation.BootDataJpaApplication      : Starting BootDataJpaApplication on iMac-de-Emmanuel-8.local with PID 41024
2020-09-12 14:18:24.371  INFO 41024 --- [           main] fr.formation.BootDataJpaApplication      : No active profile set, falling back to default profiles: default
2020-09-12 14:18:24.694  INFO 41024 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2020-09-12 14:18:24.709  INFO 41024 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 9ms. Found 0 JPA repository in
2020-09-12 14:18:24.976  INFO 41024 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2020-09-12 14:18:25.157  INFO 41024 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2020-09-12 14:18:25.196  INFO 41024 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2020-09-12 14:18:25.235  INFO 41024 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.18.Final
2020-09-12 14:18:25.339  INFO 41024 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2020-09-12 14:18:25.432  INFO 41024 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
2020-09-12 14:18:25.613  INFO 41024 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transac
2020-09-12 14:18:25.619  INFO 41024 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2020-09-12 14:18:25.696  INFO 41024 --- [           main] fr.formation.BootDataJpaApplication      : Started BootDataJpaApplication in 1.564 seconds (JVM running for 1.849)
2020-09-12 14:18:25.699  INFO 41024 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2020-09-12 14:18:25.700  INFO 41024 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Starting delayed evictData of schema as part of SessionFactory
2020-09-12 14:18:25.702  INFO 41024 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2020-09-12 14:18:25.703  INFO 41024 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Couche de persistance

- Exemple avec Spring Data JPA et H2 Database

```java
@Entity
public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String login;
    private String password;

    // [ ... ]
```

```java
public interface UserDao extends JpaRepository<User, Integer> {

}
```

```java
@SpringBootTest
class BootDataJpaApplicationTests {

    @Autowired
    UserDao uDao;

    @Test
    void testAjoutUserEnBase() {
        User u = new User("form", "ation");
        uDao.save(u);

        assertAll(
                ()->assertEquals(1, u.getId()),
                ()->assertEquals("form", u.getLogin()),
                ()->assertEquals("ation", u.getPassword())
        );
    }
}
```

Couche de persistance

- Exemple avec Spring Data JPA et H2 Database
- Convention over configuration
  - Découverte de H2 dans le classpath
  - Utilisation de H2 en mémoire
  - Possibilité de définir une autre configuration
    - Dans le fichier application.properties

```
#spring.datasource.url=jdbc:h2:mem:testdb;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.url=jdbc:h2:file:~/demo

spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# create (par défaut)
spring.jpa.hibernate.ddl-auto=none
```

Démonstration

Spring Boot

Couche Web

- Avec Spring initialirz

Project
Maven Project Gradle Project

Language
Java Kotlin Groovy

Spring Boot
2.4.0 (SNAPSHOT) 2.4.0 (M2) 2.3.4 (SNAPSHOT) 2.3.3
2.2.10 (SNAPSHOT) 2.2.9 2.1.17 (SNAPSHOT) 2.1.16

Project Metadata

Group	fr.formation
Artifact	boot-web
Name	boot-web
Description	Demo project for Spring Boot
Package name	fr.formation
Packaging	Jar War
Java	14 11 8

Dependencies ADD DEPENDENCIES... ⌘ + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot

Couche Web

- Exemple avec Spring Web

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

```
@SpringBootApplication
public class BootWebApplication {

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

}
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.9.RELEASE)

2020-09-12 15:05:57.725  INFO 41475 --- [           main] fr.formation.BootWebApplication          : Starting BootWebApplication on iMac-de-Emmanuel-8.local with PID 41475
2020-09-12 15:05:57.727  INFO 41475 --- [           main] fr.formation.BootWebApplication          : No active profile set, falling back to default profiles: default
2020-09-12 15:05:58.461  INFO 41475 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2020-09-12 15:05:58.471  INFO 41475 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-09-12 15:05:58.472  INFO 41475 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.37]
2020-09-12 15:05:58.531  INFO 41475 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2020-09-12 15:05:58.531  INFO 41475 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 766 ms
2020-09-12 15:05:58.715  INFO 41475 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2020-09-12 15:05:58.878  INFO 41475 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2020-09-12 15:05:58.881  INFO 41475 --- [           main] fr.formation.BootWebApplication          : Started BootWebApplication in 1.51 seconds (JVM running for 1.954)
```

Spring Boot

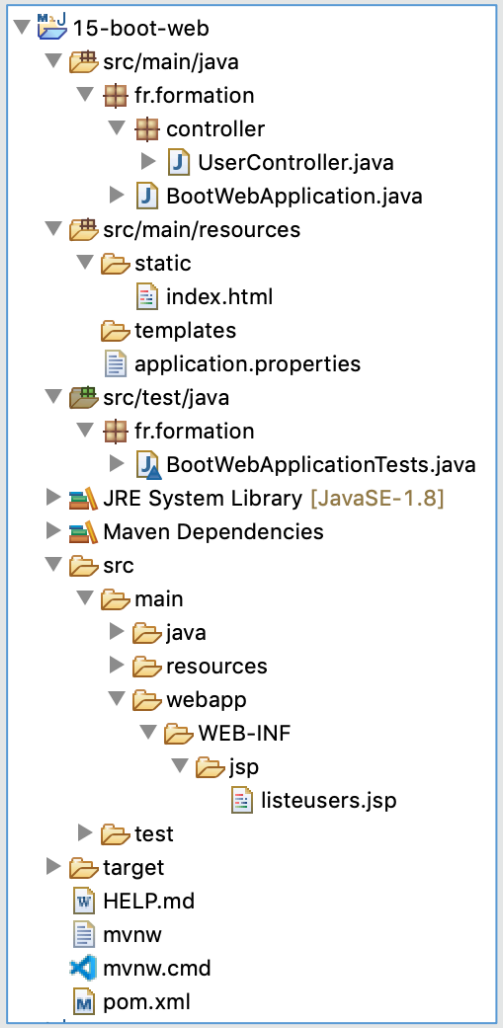
Couche Web

- Exemple avec Spring Web MVC
  - Ajout de la dépendance Tomcat Embed pour les JSP et de JSTL

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>
```

  - Ajout dans le fichier application.properties des informations relatives aux jsp

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

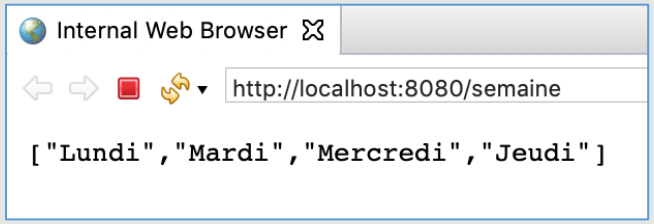


Démonstration

Couche Web

- Exemple avec Spring Web REST
  - Création d'un Controleur Rest (et c'est tout)

```
@RestController
public class UserRestController {

    @RequestMapping(value="/semaine")
    public ResponseEntity<List<String>> listeChaine() {
        List<String> listeS = Arrays.asList("Lundi", "Mardi", "Mercredi", "Jeudi");
        return new ResponseEntity<List<String>>(listeS, HttpStatus.OK);

    }
}
```



Démonstration