Spring Boot

Introduction

Spring Boot

- Spring Boot permet de créer des applications :
  - **Basée sur Spring**
    - Avec un minimum de configuration
  - **Autonomes**
    - Il n'est pas obligatoire de posséder en plus un serveur d'application.
    - L'application peut se lancer avec un "java -jar"

Spring Boot

- Objectifs
  - **Rapidité de développement**
    - Avec un minimum de configuration
  - **De la configuration par défaut**
  - **Des services techniques prêts à l'usage**
    - Serveur embarqué, sécurité, …
  - **Pas de génération de code**, pas d'obligation de faire du XML

Spring Boot

- Prérequis
  - **Java 8 recommandé**
    - Java 7 supporté
    - Java 6 toléré (configuration particulière à faire)
  - Conteneurs supportés nativement
    - Tomcat 7 & 8
    - Jetty 8 & 9
    - Undertow 1.1

Spring Boot Maven

Spring Boot avec Maven

- Spring Boot avec Maven
  - **Option 1 : Définir un projet parent “spring-boot-starter-parent"**
    - org.springframework.boot:spring-boot-starter-parent

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.5.RELEASE</version>
</parent>
```

  - **Option 2 : Une dépendance en scope “import”**

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>1.3.5.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Spring Boot avec Maven

- Pour créer une application Web, ajouter une dépendance vers le projet “spring-boot-starter-web”

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

Spring Boot Maven

- Une application Web comme un “main()” Java classique.

```
public class AppMain {

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

@EnableAutoConfiguration
public class AppSpringConfig {
}
```

Spring Boot Maven

- Une application Web comme un “main()” Java classique + scan des packages pour trouver des beans

```
public class AppMain {

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

@SpringBootApplication
public class AppSpringConfig {
}
```

Spring Boot Maven

- Plugin Maven

```
<plugin>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-maven-plugin</artifactId>
     <version>1.3.5.RELEASE</version>
</plugin>
```

- Tâches
  - La tâche **repackage** permet de générer un jar exécutable qui démarre un serveur. (spring-boot:repackage)
  - La tâche **run** démarre l'application (spring-boot:run)

Spring Boot Maven

- Configuration
  - Spring Boot se configure via le fichier "src/main/resources/**application.properties**"
  - Exemple de configuration

```
# datasource
spring.datasource.url = jdbc:h2:mem:testdb
spring.datasource.driver-class-name = org.h2.Driver
spring.datasource.username = sa
spring.datasource.password =
spring.jpa.generate-ddl=true

# mvc
spring.mvc.view.prefix = /pages/
spring.mvc.view.suffix = .html
```

Spring Boot Maven

- Ressources Web
  - Par convention, les ressources Web sont localisées dans les répertoires
    - src/main/resources**/static** : pour les ressources statiques (HTML, JS, CSS, …)
    - src/main/resources**/templates** : pour les ressources dynamiques utilisant un technologie de templating (thymeleaf, groovy, freemarker, …)

Spring Boot Maven

- WebJars (http://www.webjars.org)
  - Permettent de gérer les dépendances vers les ressources statiques automatiquement publiées via le chemin "/webjars/xxx".
  - Exemple de WebJars

```
<dependency>
      <groupId>org.webjars</groupId>
      <artifactId>jquery</artifactId>
      <version>1.11.1</version>
</dependency>
```

  - Exemple d'inclusion dans une page HTML

```
<script src="webjars/jquery/1.11.1/jquery.js"></script>
```

Spring Boot CLI

Spring Boot CLI

- Client Spring Boot
  - Télécharger Spring Boot CLI : http://repo.spring.io/release/org/springframework/boot/spring-boot-cli/
  - Spring Boot CLI donne accès à la commande “spring”
  - Spring Boot CLI permet également de d’exécuter un script Groovy
  - Plusieurs options sont disponibles :
    - run, test, grab, jar, war, install, uninstall, init, shell

```
➜  ~ spring help
usage: spring [--help] [--version]
       <command> [<args>]

Available commands are:

  run [options] <files> [--] [args]
    Run a spring groovy script

  test [options] <files> [--] [args]
    Run a spring groovy script test

  grab
    Download a spring groovy script's dependencies to ./repository

  jar [options] <jar-name> <files>
    Create a self-contained executable jar file from a Spring Groovy script

  war [options] <war-name> <files>
    Create a self-contained executable war file from a Spring Groovy script

  install [options] <coordinates>
    Install dependencies to the lib directory

  uninstall [options] <coordinates>
    Uninstall dependencies from the lib directory

  init [options] [location]
    Initialize a new project using Spring Initializr (start.spring.io)

  shell
    Start a nested shell

Common options:

  -d, --debug Verbose mode
    Print additional status information for the command you are running
```

Spring Boot CLI

- Script Groovy
  - Spring Boot CLI permet d'écrire des scripts groovy démarrés en application Web
  - Exemple de Script (fichier exempleApp.groovy)

```
@RestController
class BonjourController {

    @RequestMapping("/bonjour")
    String bonjour() {
        return "bonjour depuis Spring Boot"
    }
}
```

  - Démarrer l'application

```
spring run exempleApp.groovy
```