

Cómo ejecutar una aplicación Spring Boot como un servicio Linux

Fernando de la Torre

19/04/2021 · **Fernando de la Torre** · Arquitecto de Software

Contenido

Empaquetado, configuración y construcción del proyecto

Empaquetado de aplicaciones Spring Boot

Configuración de nuestro pom.xml

Construcción de nuestra aplicación

Configuración de un servicio en Linux

Estructuración de carpetas

Application.yml

Creación del .sh

Permisos de nuestro bash

Referencia de servicio

Inicio del servicio

Parada del servicio

Ver los servicios activos

Conclusión

En este post veremos **cómo ejecutar una aplicación Spring Boot como un servicio** [Linux](#) configurado desde un *bash* con configuración externa.

Empaquetado, configuración y construcción del proyecto

En primer lugar, explicaremos las **opciones de empaquetado, la configuración del proyecto y la construcción de este**.

Empaquetado de aplicaciones Spring Boot

Las aplicaciones [Spring Boot](#) se pueden empaquetar como **archivos WAR y JAR**.

Configuración de nuestro pom.xml

Lo primero que deberemos hacer es la **configuración de nuestro pom.xml**, al cual tenemos que añadirle la dependencia *parent* (**org.springframework.boot:spring-boot-starter-parent:2.4.3**) y la configuración del *build*:

```

<packaging> jar </packaging>

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.4.3</version>
</parent>

<dependencies>
  ....
</dependencies>

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

```

Nota: El **packaging jar** viene por defecto, por lo que no hace falta ponerlo.

Establecemos el parámetro a **true** para el artefacto **spring-boot-maven-plugin**. Esto hace que se agregue el archivo **MANIFEST.MF** al paquete JAR. Este manifiesto contiene una entrada de clase principal que especifica qué clase define el método principal para su aplicación.

Construcción de nuestra aplicación

El siguiente paso será la ejecución del comando *clean package* de Maven. El siguiente comando dentro del directorio raíz de su aplicación:

```
mvn clean package
```

El **archivo jar ejecutable** se encuentra disponible en la carpeta **Target**.

Podemos ejecutar nuestro jar simplemente con el comando:

```
java -jar mi-profile-app.jar
```

Configuración de un servicio en Linux

En las secciones siguientes, exploramos **diferentes alternativas que tenemos al configurar un servicio en Linux**.



Conseguir que un **ejecutable de Spring Boot con empaquetado Jar a servicio** es un proceso bastante fácil. Lo primero que debemos hacer es **colocar el jar en la ruta deseada**, en nuestro caso lo colocaremos en la siguiente ruta:

```
/var/applications/miProfileApp/mi-profile-app.jar
```

Ahora tan solo tendremos que **crear un enlace entre nuestro fichero y asignarlo al servicio**. Para ello, debemos ejecutar la siguiente línea en la consola:

```
ln -s /var/applications/miProfileApp/mi-profile-app.jar /etc/init.d/profile_app_service
```

Después lo único que tendríamos que realizar es la **inicialización del servicio** mediante la ejecución de la siguiente línea:

```
service profile_app_service start
```

El funcionamiento de estos dos últimos comandos lo explicaremos más adelante.

Lo que ocurre es que esta opción trabajaría con el fichero de configuración que exista dentro de nuestro archivo jar. Para poder **ejecutar un fichero de configuración externo** al que exista en el jar, deberemos **crear un bash que sea el que asociemos a nuestro servicio**.

Por ello, ahora procederemos a explicar dónde debemos colocar nuestro fichero de configuración, la estructura de nuestro *bash*, los permisos que debe tener, como crear la referencia al servicio y los procesos de start / stop y cómo ver los servicios activos de nuestro servidor.



Estructuración de carpetas

Lo primero que tenemos que tener claro es **dónde vamos a colocar nuestros ficheros de configuración, nuestro jar,...**

Para ello nosotros hemos decidido la siguiente estructura:

```
/var
/ - applications
/   - miProfileApp
/     + mi-profile-app-1.0.0-SNAPSHOT.jar
/     - config
/       + application.yml
/ - scripts
/   + mi_profile_app.sh
```

Application.yml

Lo primero que haremos es **colocar nuestro fichero en la ruta correspondiente**:

```
/var/applications/miProfileApp/config/application.yml
```

Creación del .sh

Una vez que tenemos colocado el fichero de configuración en la ruta correspondiente **crearemos el fichero *bash***, que contendrá la información para ejecutar la configuración externa.

Para ello crearemos en nuestra ruta de scripts nuestro bash “**mi_profile_app.sh**”. Para crear el sh realizaremos lo siguiente:

```
vi /vars/scripts/mi_profile_app.sh
```

Y como **contenido de nuestro *bash*** sería:

```
#!/bin/bash

JAR_NAME = mi-profile-app-1.0.0-SNAPSHOT.jar
OPTIONS = " -c /var/applications/miProfileApp/config/application.yml"

cd /var/applications/miProfileApp
"${JAVA_HOME}/bin/java" ${JAVA_OPTIONS} -jar $JAR_NAME $OPTIONS
```

Entendemos que tendremos configurado las [variables](#) de entorno de **JAVA_HOME** y **JAVA_OPTIONS**. En caso de no tener configuradas las **JAVA_OPTIONS**, podremos añadir a nuestro *bash* la siguiente línea:

```
JAVA_OPTIONS=" -Xms256m -Xmx512m -server "
```

Permisos de nuestro bash

Cuando tengamos creado nuestro fichero bash, le daremos **permisos mediante el comando *chmod* de usuario (u) y de ejecución (x)**. La línea que pondremos en nuestra consola sería la siguiente:

```
chmod u+x mi_profile_app.sh
```

Referencia de servicio

Para **crear nuestro *bash* como referencia de nuestro servicio** debemos escribir la siguiente línea:

```
ln -s /var/scripts/mi_profile_app.sh /etc/init.d/profile_app_service
```

Este comando lo que realiza es la **creación de un enlace simbólico a nuestro fichero *bash***. La ruta donde se encuentra en *script* debe ser la ruta completa. Sino el enlace simbólico no funciona de la forma adecuada.

Inicio del servicio

Una vez creada nuestra referencia, podemos **iniciar nuestra [aplicación Spring Boot](#) como servicio Linux**. Para ello ejecutaremos el siguiente comando:

```
service profile_app_service start
```

Parada del servicio

Si por el contrario queremos **parar nuestro servicio**, ejecutaremos el comando de *stop*:

```
service profile_app_service stop
```

Ver los servicios activos

Si lo que deseamos es **ver todos los servicios activos** que tenemos, podemos ejecutar el comando:

```
service --status-all
```

Conclusión

De esta forma, podemos **administrar el estado de nuestra aplicación Spring Boot mediante un servicio Linux de forma eficiente** y tener toda su configuración externalizada. La cual siempre estará lista para arrancar o parar según nuestras necesidades. Al fin y al cabo, lo importante es poder tener la configuración de nuestra aplicación en un fichero externo, el cual podemos cambiar el contenido del mismo y solo tendríamos que reiniciar nuestro servicio.