How to Use Spring Boot CLI

Spring Boot CLI (Command Line Interface) is a powerful tool that allows you to quickly develop, test, and prototype Spring Boot applications directly from the command line. It's especially useful for running Groovy scripts without setting up a full project structure, generating new projects, or experimenting with Spring Boot features. Below is a complete guide on how to install and use Spring Boot CLI effectively.

Installation

Before you can use Spring Boot CLI, you need to install it. There are two primary methods depending on your operating system:

1. Using SDKMAN! (Recommended for Unix-based Systems like Linux or macOS)

SDKMAN! is a tool for managing software development kits, making it an easy way to install Spring Boot CLI.

- **Step 1: Install SDKMAN!** Open your terminal and run:

  ```
  curl -s "https://get.sdkman.io" | bash
  ```

  Follow the prompts to initialize SDKMAN! by sourcing the script:

  ```
  source "$HOME/.sdkman/bin/sdkman-init.sh"
  ```

- **Step 2: Install Spring Boot CLI** Run the following command:

  ```
  sdk install springboot
  ```

2. Manual Installation (For Windows or Manual Setup)

If you're on Windows or prefer manual installation: - Download the Spring Boot CLI ZIP file from the official Spring website. - Extract the ZIP file to a directory of your choice. - Add the `bin` directory from the extracted folder to your system's PATH environment variable.

Verify Installation

To confirm that Spring Boot CLI is installed correctly, run this command in your terminal:

```
spring --version
```

You should see the installed version of Spring Boot CLI (e.g., `Spring CLI v3.3.0`). If this works, you're ready to start using it!

Key Ways to Use Spring Boot CLI

Spring Boot CLI provides several features that make it ideal for rapid development and prototyping. Here are the main ways to use it:

1. Running Groovy Scripts

One of the standout features of Spring Boot CLI is its ability to run Groovy scripts directly without requiring a full project setup. This is perfect for quick prototyping or experimenting with Spring Boot.

- **Example: Creating a Simple Web Application** Create a file named `hello.groovy` with the following content:

  ```
  @RestController
  class HelloController {
      @RequestMapping("/")
      String home() {
          "Hello, World!"
      }
  }
  ```

- **Run the Script** In your terminal, navigate to the directory containing `hello.groovy` and run:

  ```
  spring run hello.groovy
  ```

  This starts a web server on port 8080. Open a browser and visit `http://localhost:8080` to see "Hello, World!"displayed.

- **Adding Dependencies** You can include dependencies directly in the script using the `@Grab` annotation. For example:

  ```
  @Grab('org.springframework.boot:spring-boot-starter-data-jpa')
  @RestController
  class HelloController {
      @RequestMapping("/")
      String home() {
          "Hello, World!"
      }
  }
  ```

This adds Spring Data JPA to your script without needing a build file.

- **Running Multiple Scripts** To run all Groovy scripts in the current directory, use:

  ```
  spring run *.groovy
  ```

2. Creating New Spring Boot Projects

Spring Boot CLI can generate a new project structure with your desired dependencies, saving you time when starting a full application.

- **Example: Generate a Project** Run this command to create a new project with web and data-jpa dependencies:

  ```
  spring init --dependencies=web,data-jpa my-project
  ```

  This creates a directory called `my-project` containing a Spring Boot application configured with Spring Web and Spring Data JPA.

- **Customization Options** You can specify additional options like:

  - Build tool: `--build=maven` or `--build=gradle`
  - Language: `--language=java`, `--language=groovy`, or `--language=kotlin`
  - Packaging: `--packaging=jar` or `--packaging=war`

  For example:

  ```
  spring init --dependencies=web --build=gradle --language=kotlin my-kotlin-project
  ```

3. Packaging Applications

Spring Boot CLI allows you to package your scripts into executable JAR or WAR files for deployment.

- **Create a JAR File**

  ```
  spring jar my-app.jar *.groovy
  ```

  This packages all Groovy scripts in the current directory into `my-app.jar`.

- **Create a WAR File**

  ```
  spring war my-app.war *.groovy
  ```

  This generates a `my-app.war` file suitable for deployment to a servlet container.

4. Running Tests

If you have Groovy test scripts, you can execute them with:

```
spring test *.groovy
```

This runs all test scripts in the current directory.

5. Using the Interactive Shell

For an interactive experience, launch the Spring Boot CLI shell:

```
spring shell
```

Inside the shell, you can run commands like `run`, `grab`, or `jar` interactively, which is great for experimentation.

Summary

Spring Boot CLI is an excellent tool for developers who want to work with Spring Boot quickly and efficiently from the command line. Here's how to use it in a nutshell:

1. **Install it** using SDKMAN! (`sdk install springboot`) or manually by downloading the ZIP and updating your PATH.
2. **Run Groovy scripts** with `spring run <script.groovy>` for rapid prototyping.
3. **Create new projects** with `spring init --dependencies=<deps> <project-name>`.
4. **Package applications** into JAR or WAR files using `spring jar` or `spring war`.
5. **Experiment interactively** with `spring shell`.

Whether you're prototyping an idea, learning Spring Boot, or automating tasks, Spring Boot CLI offers a lightweight and flexible way to get started without the overhead of a full IDE or project setup.