CYDEO

Spring Boot

Date: 24th February 2022

Why Spring Boot?

Building a Spring application is DIFFICULT

To able to get start:

- Which Maven archetype to use?
- What dependencies to choose?
- How to set up configuration(xml, java)?
- How to install server?

A lot of configuration and very easy to make mistake

Spring Boot

The most popular Java framework in the market.

It is an open-source extension of the Spring Framework designed to simplify the Spring application development.

It allows developers to focus on the business logic rather than spending time on the technical code and the associated configurations.

Spring Boot

The popularity of Spring Boot is mostly attributed to its ability to create standalone, production-ready Spring-based applications in no time that you can just run, not worrying much about the configuration hazards.

Convention-over-configuration solution

Spring Boot Core Features

Fast Bootstrapping : Spring Boot provides a fast getting-started experience in Spring application development.

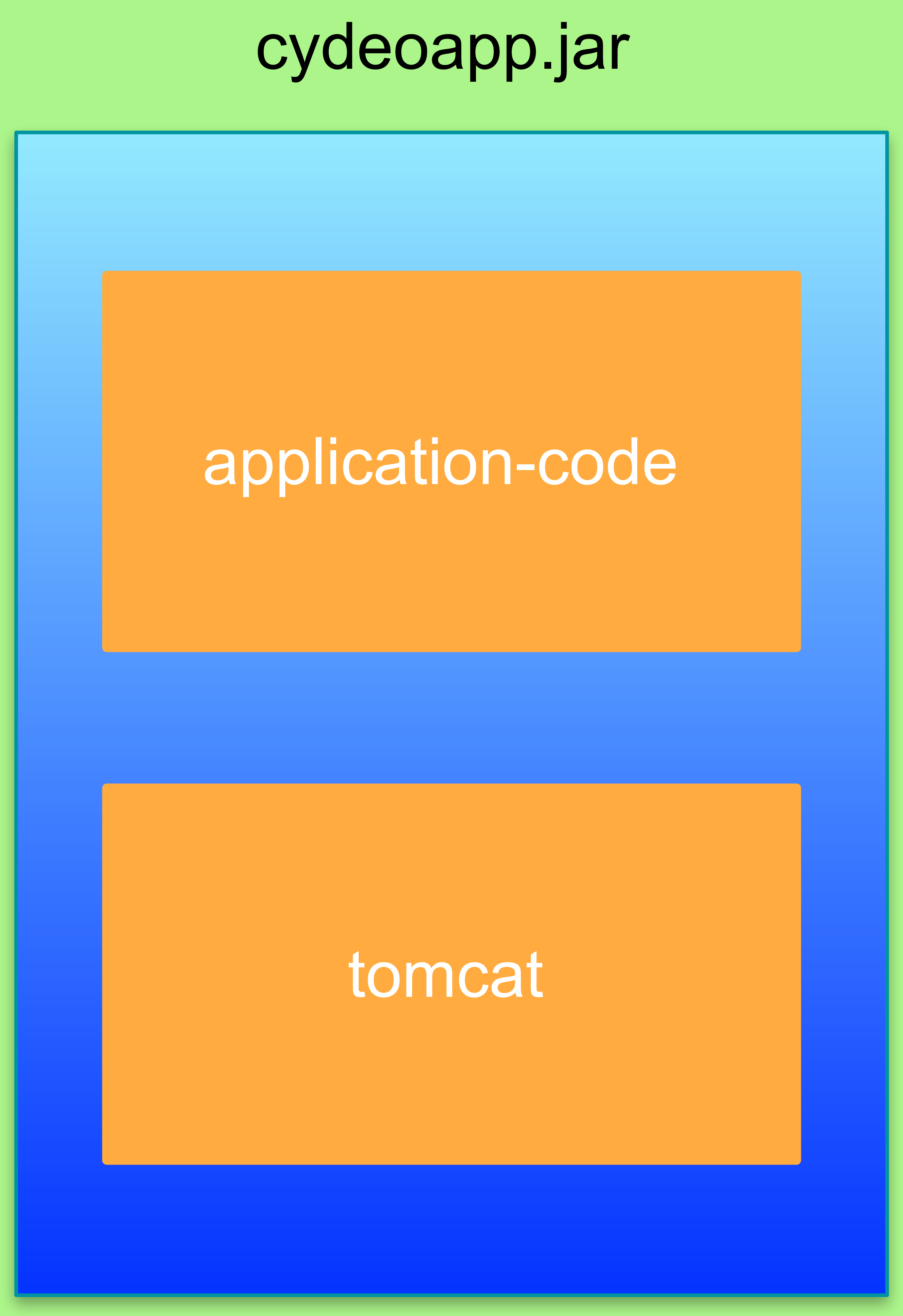
Auto-configuration : Spring Boot automatically configures the bare minimum components of a Spring application.

Spring Boot Core Features

Opinionated : It automatically configures several components to start with a Spring application. It does this with a set of setter starter dependencies. For example if you want to build a web application, you can configure the spring-boot-starter-web dependency which ensures that all related dependencies to develop a web application in the application classpath.

Spring Boot Core Features

Standalone : Spring Boot applications embed a web server so that they can run standalone and do not necessarily require an external web or application server.



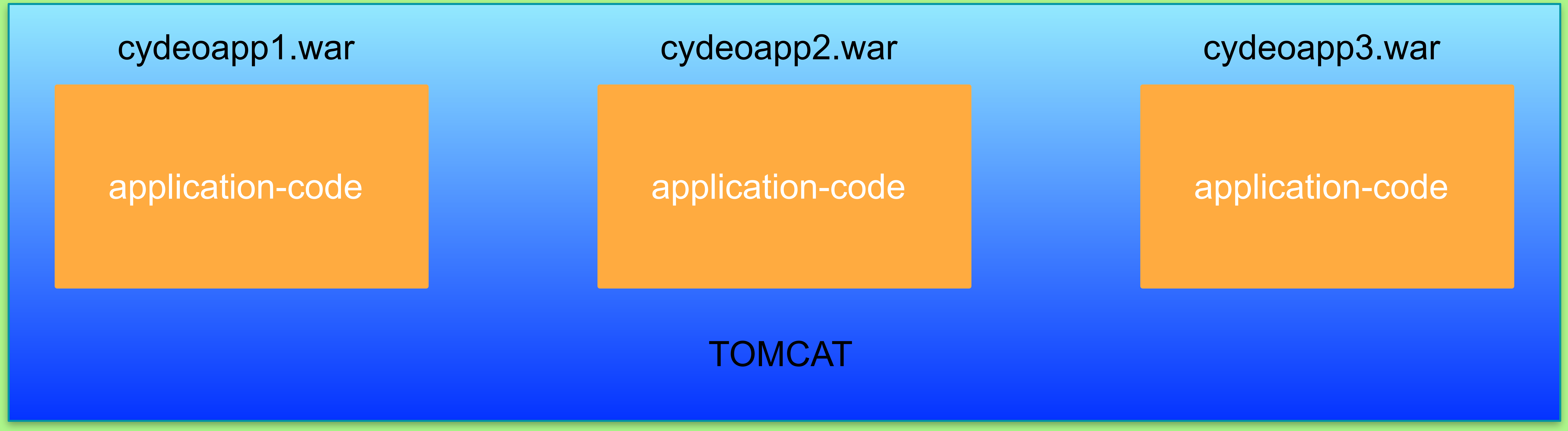
Self-contained unit

Nothing else to install

Spring Boot apps can be run standalone from the IDE or command line

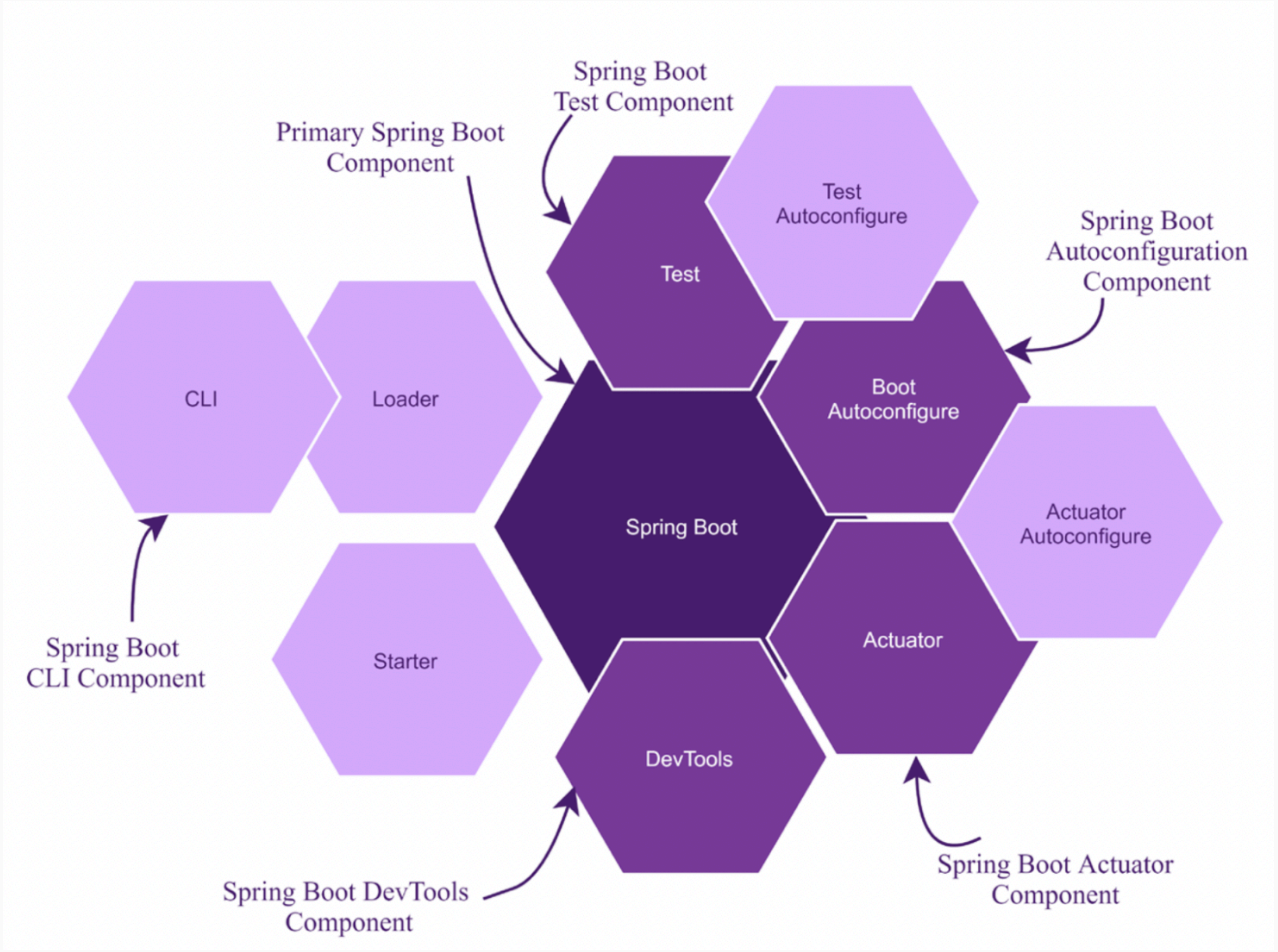
```
java -jar cydeoapp.jar
```

- Spring Boot applications can also be deployed in the traditional way
- Deploy WAR file to an external server: Tomcat, Jboss, WebSphere etc.



Spring Boot Components

Spring Boot consists of several components, with each component focuses on a specific area of the application development.



Spring Boot Test Component
Primary Spring Boot Component
Test
Test Autoconfigure
Spring Boot Autoconfiguration Component
CLI
Loader
Boot Autoconfigure
Spring Boot
Actuator Autoconfigure
Actuator
Spring Boot CLI Component
Starter
DevTools
Spring Boot DevTools Component
Spring Boot Actuator Component

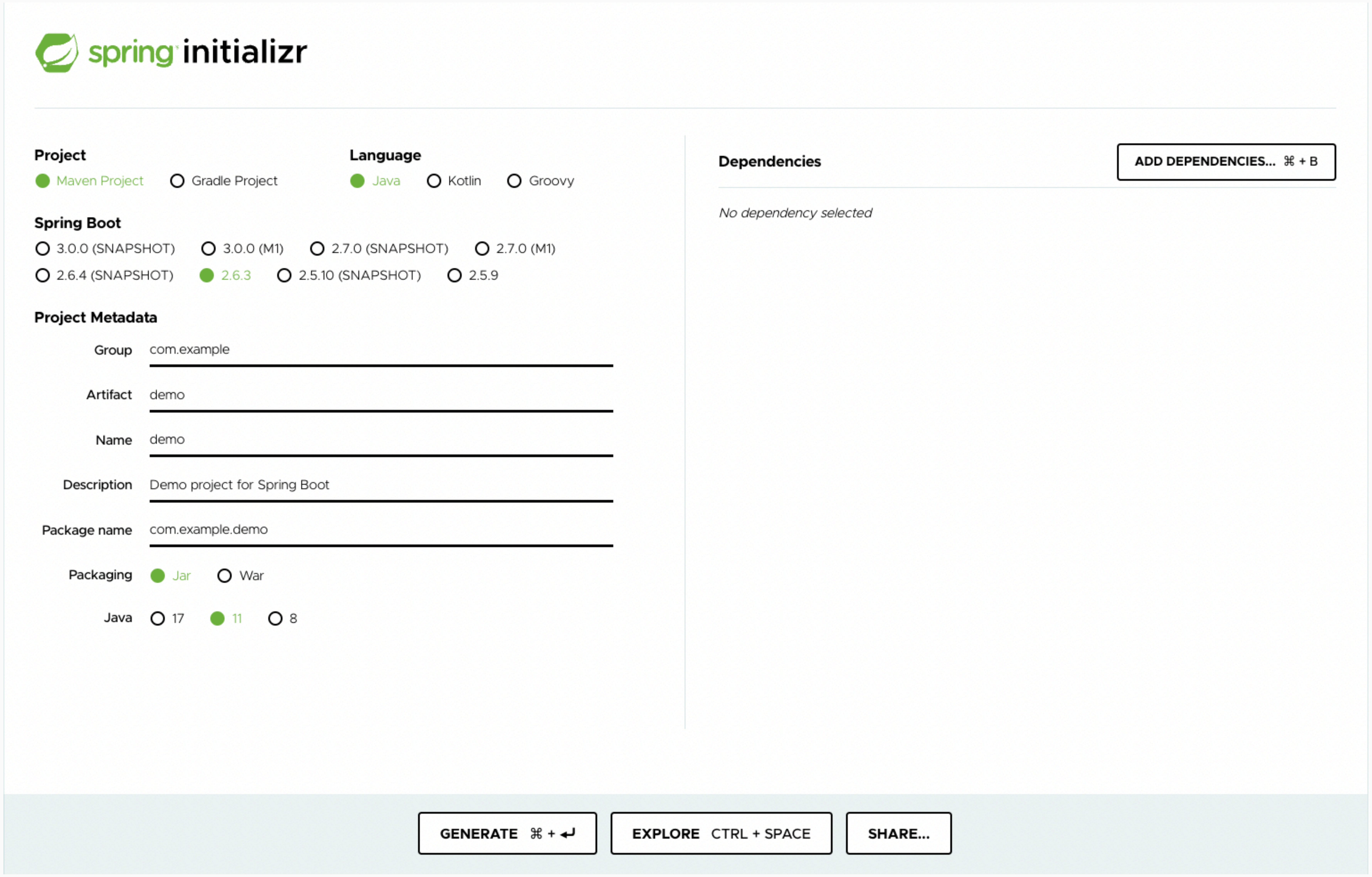
Generating a Spring Boot Project

Spring Boot provides a tool called Spring Initializr that lets you to generate a skeleton Spring Boot project.

You can access the Spring Initializr tool hosted at https://start.spring.io/

Spring Boot also provides APIs that allows the mainstream IDE vendors to integrate Spring Initializr and provide built-in support to generate Spring Boot project in the IDE itself.

Generate Spring Boot Application with Spring Initializr Web User Interface



spring initializr

Project

[✓ Maven Project] Gradle Project

Language

[✓ Java] Kotlin Groovy

Spring Boot

3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1)
2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies

ADD DEPENDENCIES... ⌘ + B

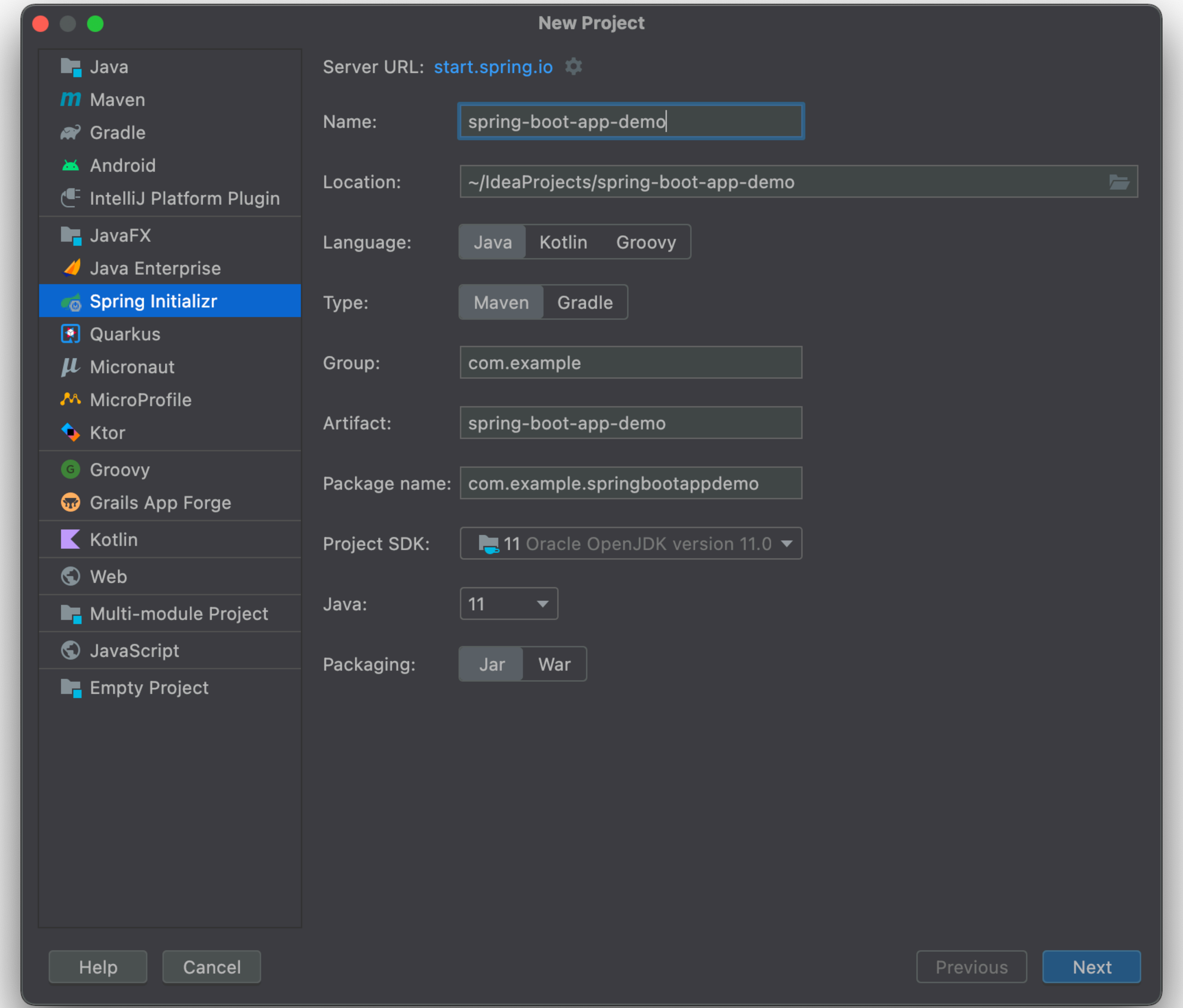
No dependency selected

GENERATE ⌘ + ↵ EXPLORE CTRL + SPACE SHARE...

Generate Spring Boot Application with Spring Initializr In IntelliJ IDEA

After specifying all the basic details such as version, project metadata, select dependencies you need it and press Generate button to generate and download the project to your machine. Spring provides a ZIP archive of the generated project. Unzip the file and import it to your IDE.

Generate Spring Boot Application with Spring Initializr In IntelliJ IDEA


New Project

Java, Maven, Gradle, Android, IntelliJ Platform Plugin, JavaFX, Java Enterprise, Spring Initializr, Quarkus, Micronaut, MicroProfile, Ktor, Groovy, Grails App Forge, Kotlin, Web, Multi-module Project, JavaScript, Empty Project

Server URL: start.spring.io

Name: spring-boot-app-demo

Location: ~/IdeaProjects/spring-boot-app-demo

Language: [✓ Java] Kotlin Groovy

Type: [✓ Maven] Gradle

Group: com.example

Artifact: spring-boot-app-demo

Package name: com.example.springbootappdemo

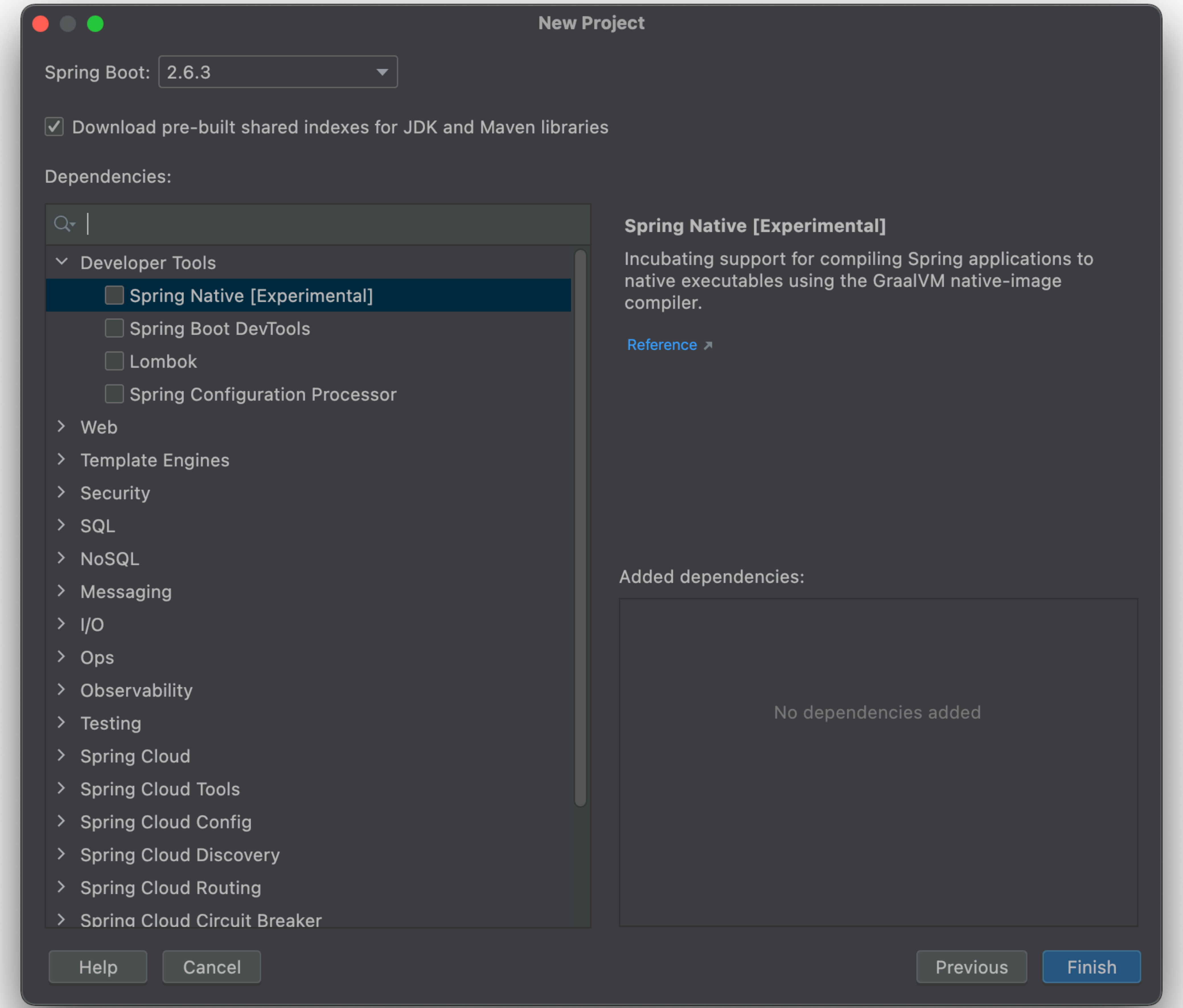
Project SDK: 11 Oracle OpenJDK version 11.0

Java: 11

Packaging: [✓ Jar] War

Help Cancel Previous Next

File › New › Project and select Spring Initializr


New Project

Spring Boot: 2.6.3

✓ Download pre-built shared indexes for JDK and Maven libraries

Dependencies:

- Developer Tools
  - ☐ Spring Native [Experimental]
  - ☐ Spring Boot DevTools
  - ☐ Lombok
  - ☐ Spring Configuration Processor
- Web
- Template Engines
- Security
- SQL
- NoSQL
- Messaging
- I/O
- Ops
- Observability
- Testing
- Spring Cloud
- Spring Cloud Tools
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Routing
- Spring Cloud Circuit Breaker

Spring Native [Experimental]

Incubating support for compiling Spring applications to native executables using the GraalVM native-image compiler.

Reference ↗

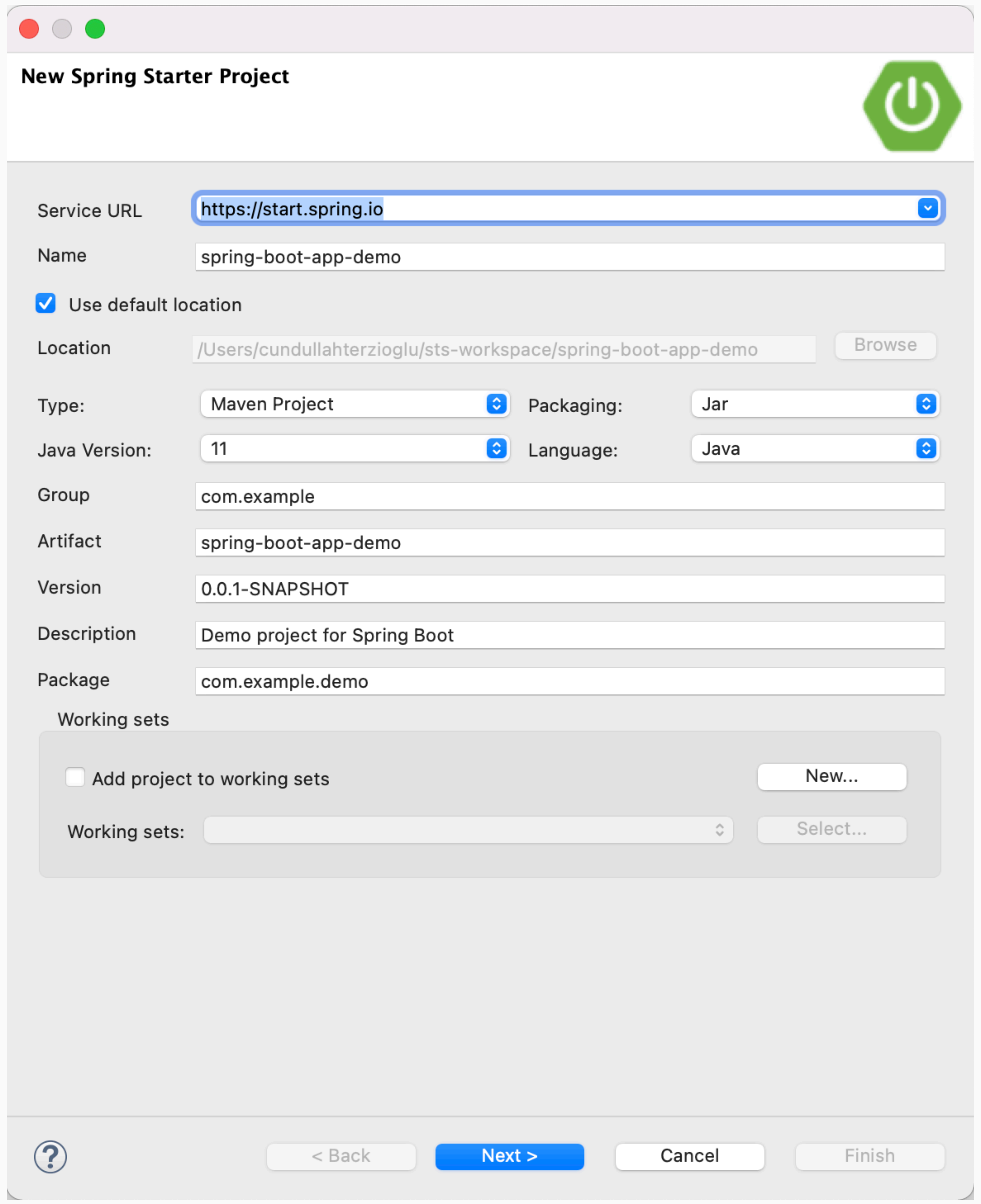
Added dependencies:

No dependencies added

Help Cancel Previous Finish

Choose the dependencies required for your project

Generate Spring Boot Application with Spring Initializr In STS



New Spring Starter Project

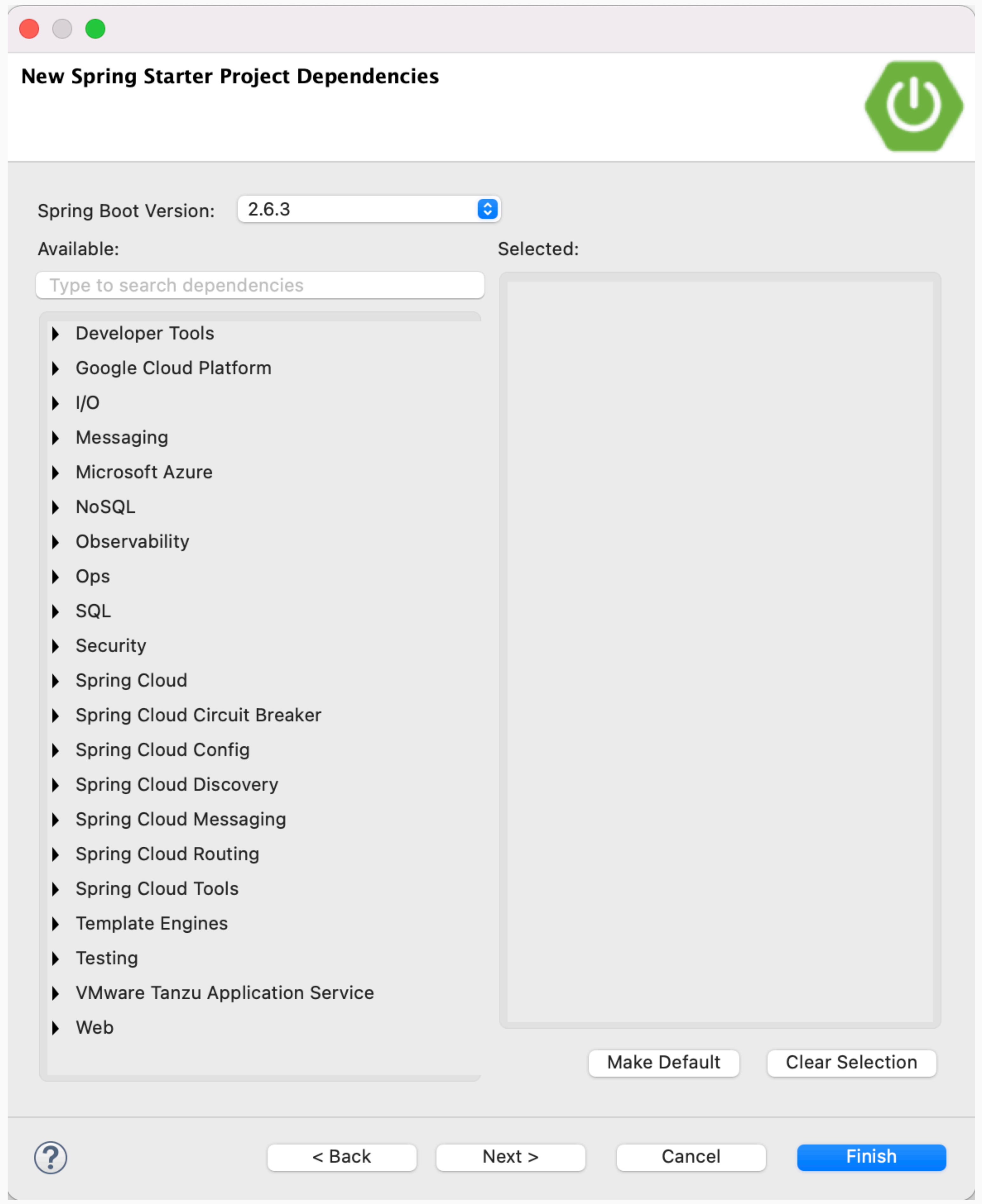
Service URL	https://start.spring.io
Name	spring-boot-app-demo
☑ Use default location	
Location	/Users/cundullahterzioglu/sts-workspace/spring-boot-app-demo
Type:	Maven Project
Packaging:	Jar
Java Version:	11
Language:	Java
Group	com.example
Artifact	spring-boot-app-demo
Version	0.0.1-SNAPSHOT
Description	Demo project for Spring Boot
Package	com.example.demo

Working sets

☐ Add project to working sets

Working sets:

File › New › Spring Starter Project



New Spring Starter Project Dependencies

Spring Boot Version: 2.6.3

Available:

Selected:

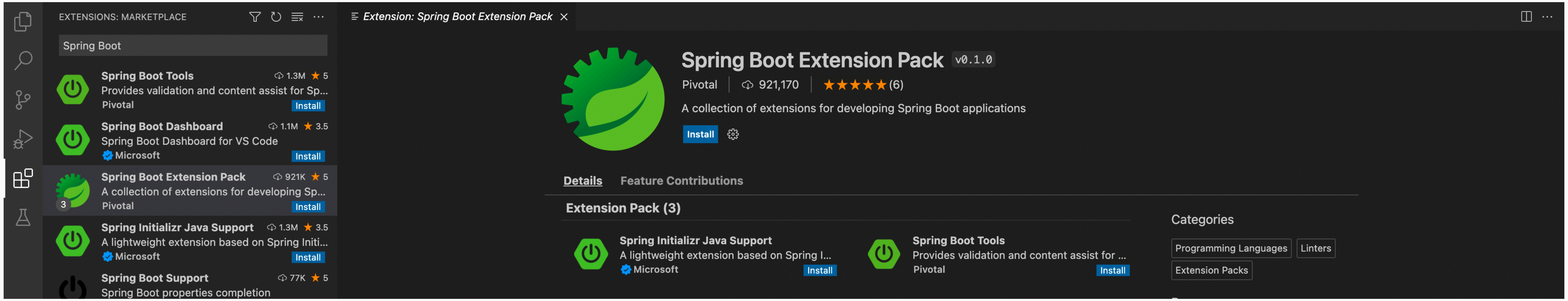
Type to search dependencies

- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Messaging
- Spring Cloud Routing
- Spring Cloud Tools
- Template Engines
- Testing
- VMware Tanzu Application Service
- Web

Choose the dependencies required for your project

Generate Spring Boot Application with Spring Initializr In VSC

VCS is an extension-based text editor from Microsoft. To able to generate a Spring Boot project in VSC, install the following extension:


EXTENSIONS: MARKETPLACE
Spring Boot
Spring Boot Tools
1.3M ★ 5
Provides validation and content assist for Sp...
Pivotal
Install
Spring Boot Dashboard
1.1M ★ 3.5
Spring Boot Dashboard for VS Code
Microsoft
Install
Spring Boot Extension Pack
921K ★ 5
A collection of extensions for developing Sp...
Pivotal
Install
Spring Initializr Java Support
1.3M ★ 3.5
A lightweight extension based on Spring Initi...
Microsoft
Install
Spring Boot Support
77K ★ 5
Spring Boot properties completion
Extension: Spring Boot Extension Pack
Spring Boot Extension Pack v0.1.0
Pivotal
921,170
★★★★★ (6)
A collection of extensions for developing Spring Boot applications
Install
Details
Feature Contributions
Extension Pack (3)
Spring Initializr Java Support
A lightweight extension based on Spring I...
Microsoft
Install
Spring Boot Tools
Provides validation and content assist for ...
Pivotal
Install
Categories
Programming Languages
Linters
Extension Packs

Generate Spring Boot Application with Spring Initializr In VSC

After successfully installing the extension pack, create Spring Boot project in the editor.

Open the Command Palette by browsing to **View › Command Palette** options and search for Spring Initializr.

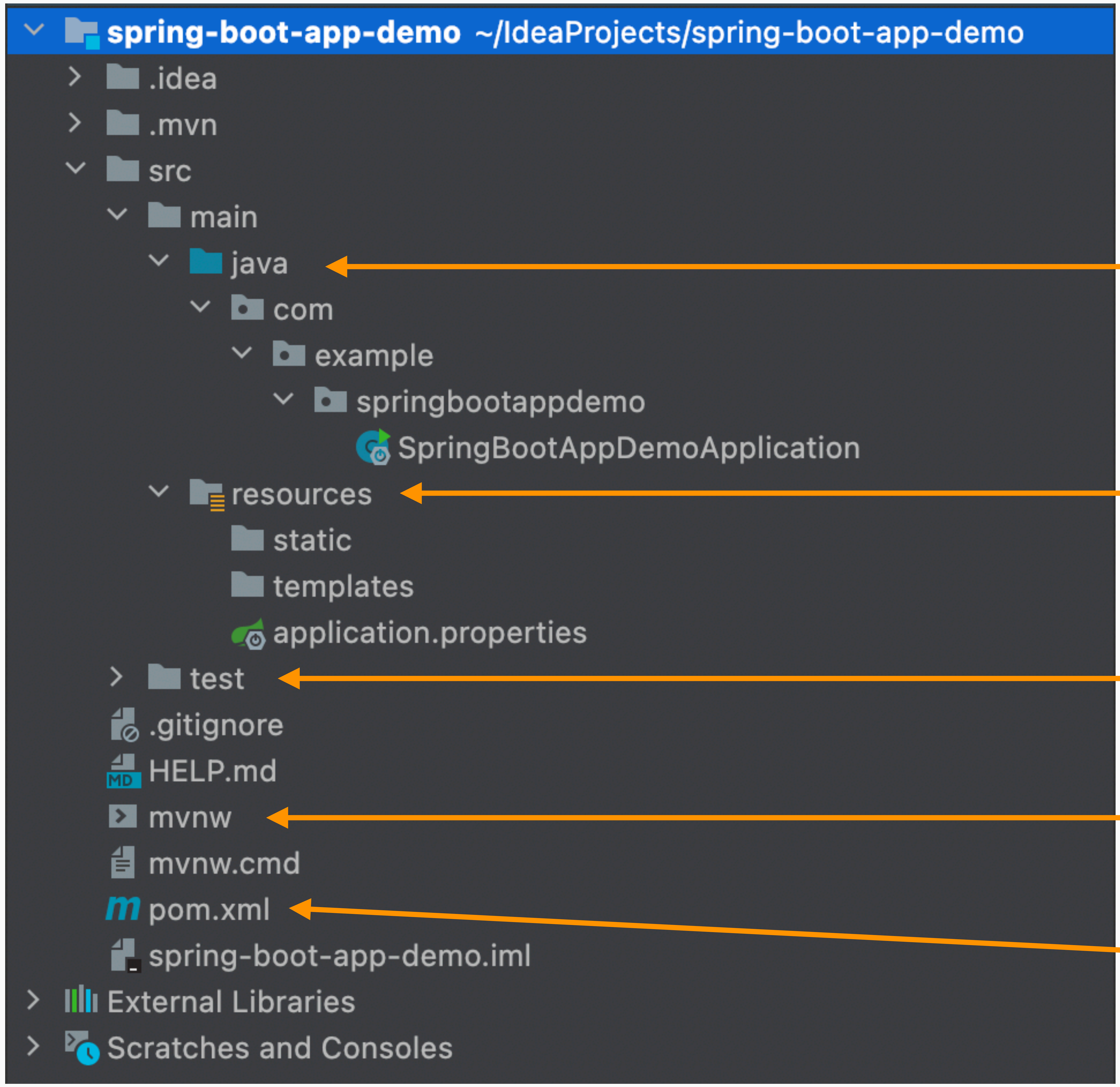
To create a project, select the required options, and follow along with parameters.

Generate Spring Boot Application with Spring Initializr In VSC

After the project is successfully generated, you will find the folder structure

After project generation, if you need to add additional dependencies, you can do it by using the Edit Starters option. Navigate to the pom.xml and right click to select Add Starters.

Spring Boot Project Structure



```
spring-boot-app-demo ~/IdeaProjects/spring-boot-app-demo
  .idea
  .mvn
  src
    main
      java
        com
          example
            springbootappdemo
              SpringBootAppDemoApplication
      resources
        static
        templates
        application.properties
    test
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  spring-boot-app-demo.iml
External Libraries
Scratches and Consoles
```

Contains Java Classes

Maintain additional project artifacts and an empty application.properties file

Contains an empty Test Class

Maven wrapper file that lets you build the project without installing Maven in your local

Contains dependencies

Build Specification

```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.3</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

The parent tag

```xml
<dependencies>

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

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>

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

</dependencies>
```

The dependencies section

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Spring Boot Maven Plugin

The Parent Tag

The **spring-boot-starter-parent** is the parent dependency for all Spring Boot starter dependencies. It indicates that the current Spring Boot project is a child Spring Boot project and extends few details from the parent project.

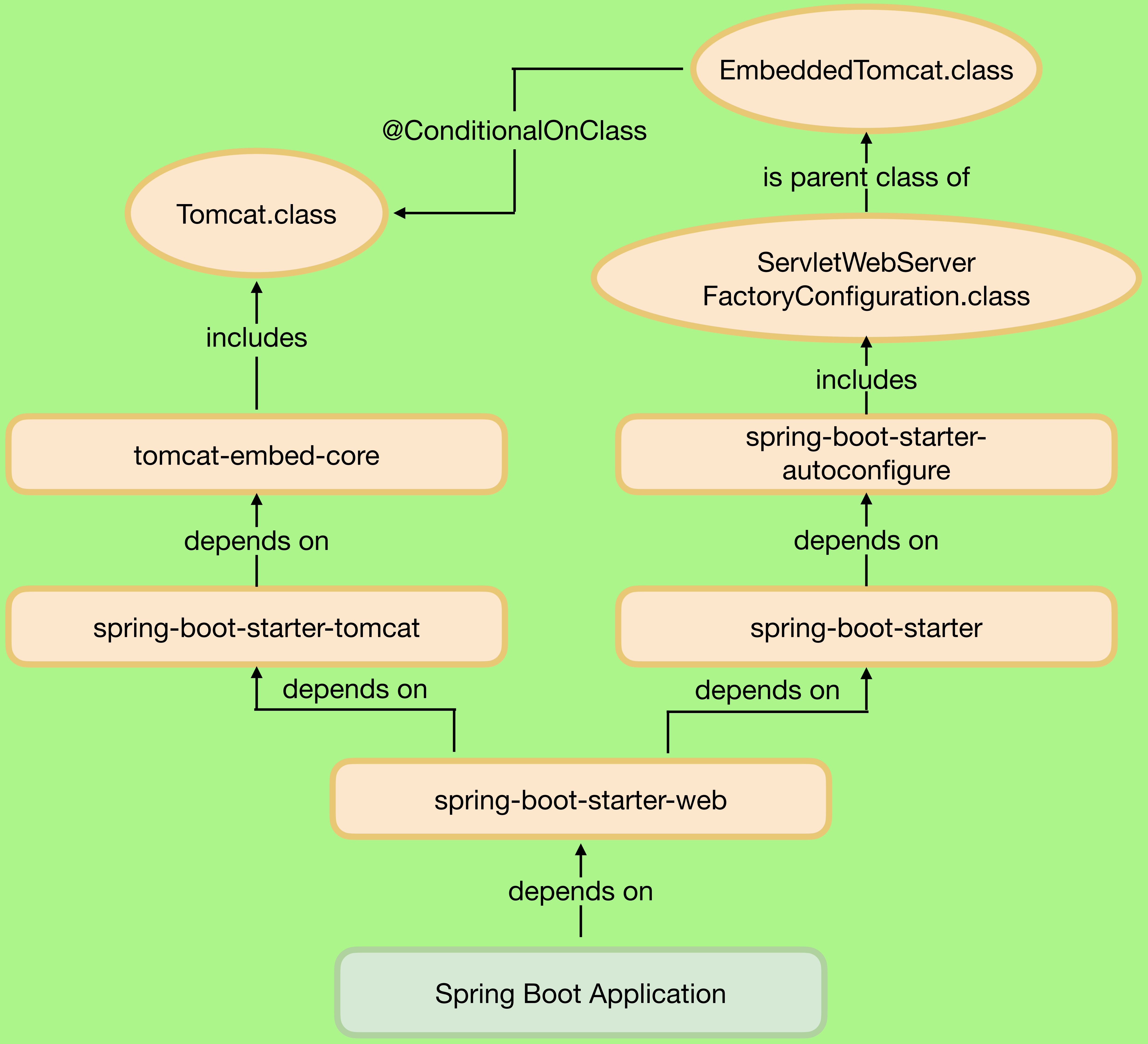
spring-boot-starter-parent is a special type of starter dependency that provides several default configurations such as default java version, default configurations for several maven plugins to a Spring Boot project.

Starter Dependencies

Starter dependency groups together a set of dependencies that you might need to develop a part of your application.

If you choose to develop a web application with Spring Boot, you will most likely choose the **spring-boot-starter-web** dependency. It ensures that all required dependencies to develop a web application are available in your application.

A starter dependency can also depend on another starter dependency.


EmbeddedTomcat.class
@ConditionalOnClass
Tomcat.class
is parent class of
ServletWebServer FactoryConfiguration.class
includes
includes
tomcat-embed-core
spring-boot-starter-autoconfigure
depends on
depends on
spring-boot-starter-tomcat
spring-boot-starter
depends on
depends on
spring-boot-starter-web
depends on
Spring Boot Application

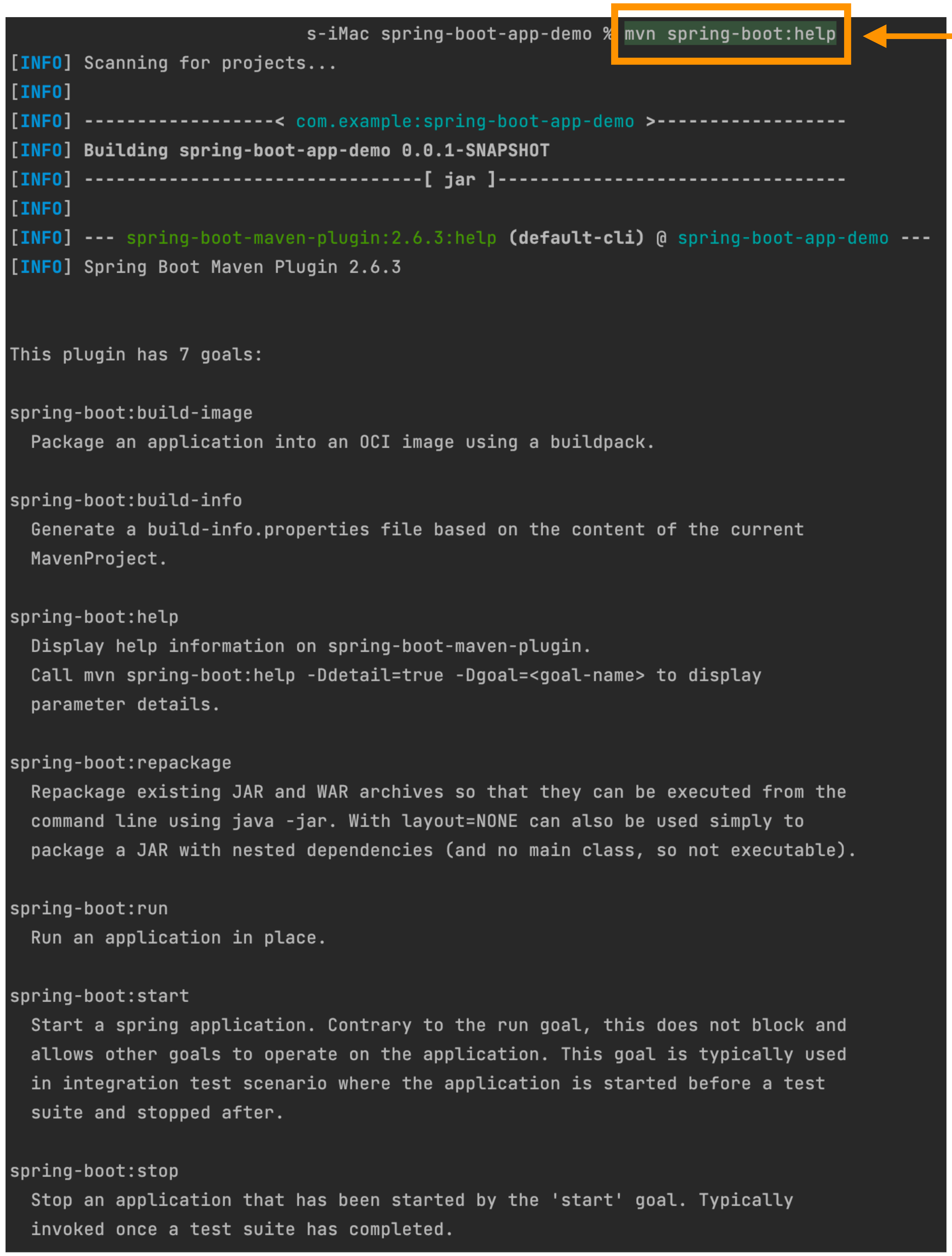
Spring Boot Maven Plugin

This plugin is provided for developer convenience to simplify several application management activities.

There are several available goals of **spring-boot-maven-plugin**.

- spring-boot:build image
- spring-boot:build-info
- spring-boot:help
- spring-boot:repackage
- spring-boot:run
- spring-boot:start
- spring-boot:stop

Goal	Description
spring-boot:build-image	Packages an application into a OCI image.
spring-boot:build-info	Generate a build-info.properties file based on the current Maven project.
spring-boot:help	Display help information on spring-boot-maven-plugin.
spring-boot:repackage	Repackage existing JAR and WAR archives so that they can be executed from command line using java -jar.
spring-boot:run	Run an application in place.
spring-boot:start	Start a Spring application. This goal is typically used in integration test scenario where the application is started before a test suite and stopped after.
spring-boot:stop	Stop an application that has been started by the “start” goal.

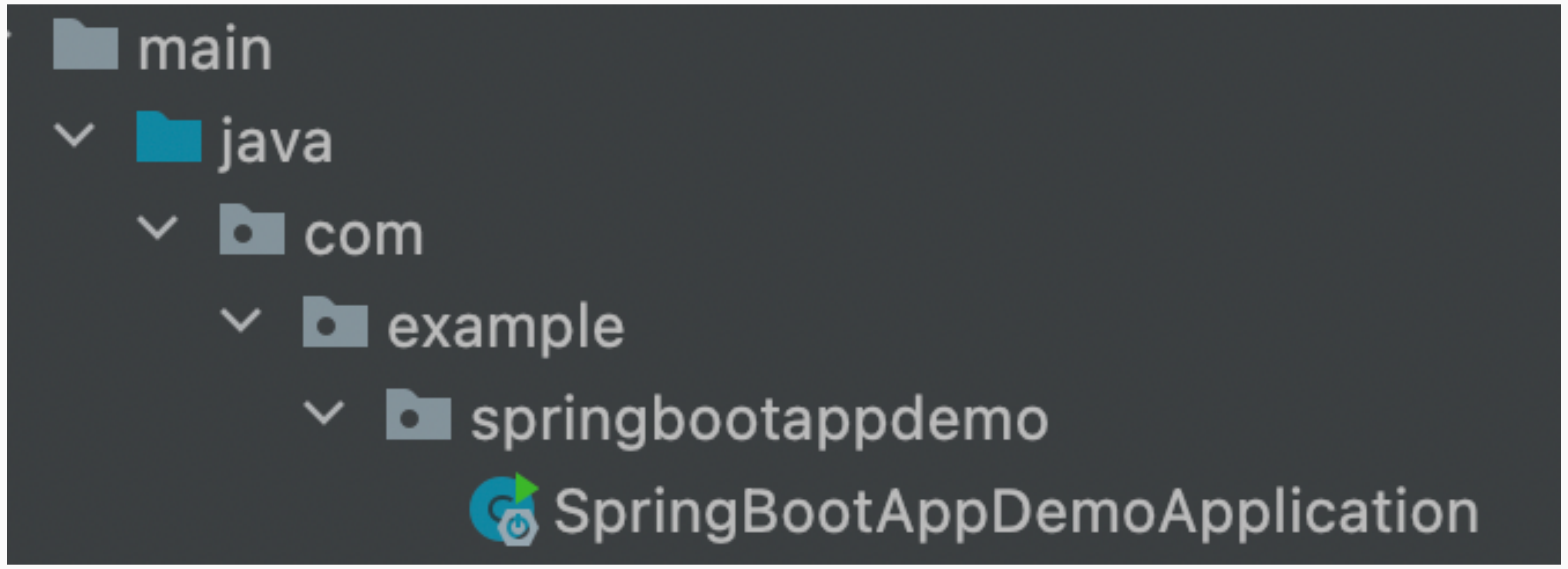


```
                            s-iMac spring-boot-app-demo % mvn spring-boot:help
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.example:spring-boot-app-demo >------------------
[INFO] Building spring-boot-app-demo 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- spring-boot-maven-plugin:2.6.3:help (default-cli) @ spring-boot-app-demo ---
[INFO] Spring Boot Maven Plugin 2.6.3


This plugin has 7 goals:

spring-boot:build-image
  Package an application into an OCI image using a buildpack.

spring-boot:build-info
  Generate a build-info.properties file based on the content of the current
  MavenProject.

spring-boot:help
  Display help information on spring-boot-maven-plugin.
  Call mvn spring-boot:help -Ddetail=true -Dgoal=<goal-name> to display
  parameter details.

spring-boot:repackage
  Repackage existing JAR and WAR archives so that they can be executed from the
  command line using java -jar. With layout=NONE can also be used simply to
  package a JAR with nested dependencies (and no main class, so not executable).

spring-boot:run
  Run an application in place.

spring-boot:start
  Start a spring application. Contrary to the run goal, this does not block and
  allows other goals to operate on the application. This goal is typically used
  in integration test scenario where the application is started before a test
  suite and stopped after.

spring-boot:stop
  Stop an application that has been started by the 'start' goal. Typically
  invoked once a test suite has completed.
```

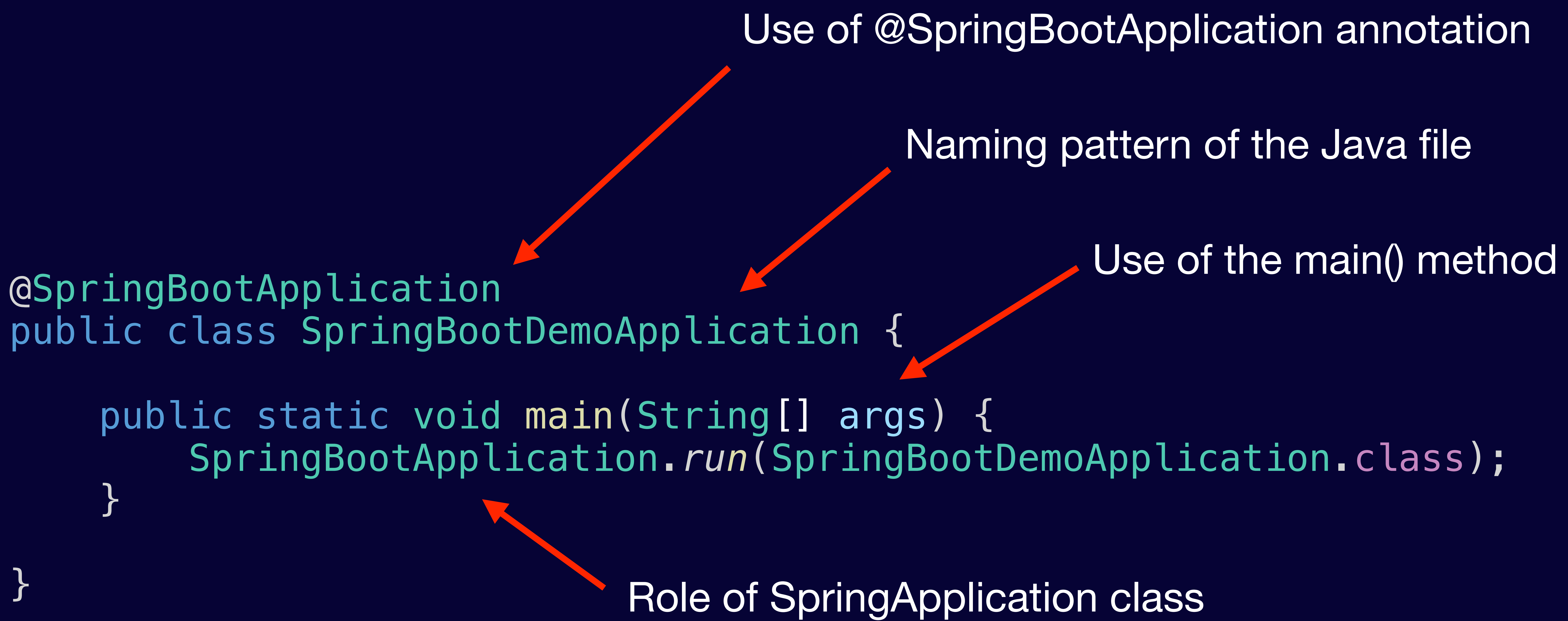
mvn spring-boot:help

Spring Boot Main Class

In the generated project, you can find that Spring Initializr has generated a Java Class with a main() method in it.



```
main
  java
    com
      example
        springbootappdemo
          SpringBootAppDemoApplication
```


Use of @SpringBootApplication annotation
Naming pattern of the Java file
Use of the main() method
Role of SpringApplication class

```java
@SpringBootApplication
public class SpringBootDemoApplication {

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

}
```

Components of the generated Java file

Naming Pattern of the Java File

Java file name follows a specific pattern.

Spring Boot application name concatenated with the word Application.

`SpringBootAppDemo` + `Application` = `SpringBootAppDemoApplication`

`SpringBootAppDemo`: Name of the Spring Boot Application

`Application`: Application

Use the Main Method

In general, to run a web application, you build and package the application components in a WAR or EAR archive file and deploy it into a web or application server (Apache Tomcat)

Spring Boot simplifies the process to a certain degree. It does not enforce you to build a WAR file of your application. Instead, it lets you run the Spring Boot application like a regular Java Application using a conventional main() method.

Use the Main Method

Spring Boot performs a decent amount of heavy lifting behind the scenes. For example, Spring Boot uses Apache Tomcat server in application by default. When you start your Spring Boot application using the main() method, Spring Boot starts an embedded instance of the Apache Tomcat server and runs the application inside it.

@SpringBootApplication Annotation

This annotation consists of three annotations :

- @EnableAutoConfiguration
- @ComponentScan
- @SpringBootConfiguration

@SpringBootApplication Annotation

@EnableAutoConfiguration : Provides necessary support for Spring Boot to auto-configure your application based on the jar dependencies present in the application class path.

@ComponentScan : Provides support to scan the packages for Spring components in the application. It scans for all components present in the root package and sub packages under it.

@SpringBootConfiguration : Indicates that the annotated class provides the Spring Boot application configuration.

Role Of Spring Application Class

This class is provided by Spring Boot to conveniently bootstrap a Spring Boot application. Spring Boot performs several activities while it executes the static run() method of SpringApplication:

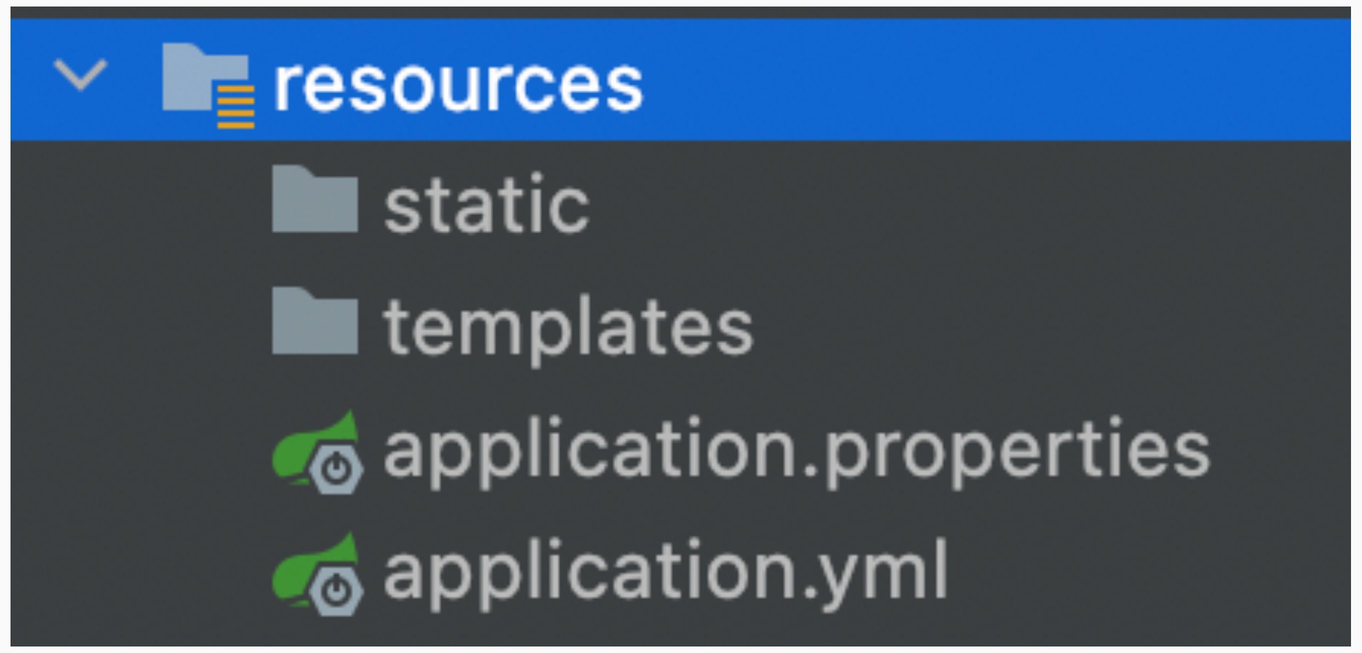
- Creates an instance of an ApplicationContext.
- Registers a CommandLinePropertySource to expose common line arguments as String properties.
- Refreshes the ApplicationContext to load all singleton beans.
- Triggers the ApplicationRunners and CommandRunners configured in the application.

Application Properties File

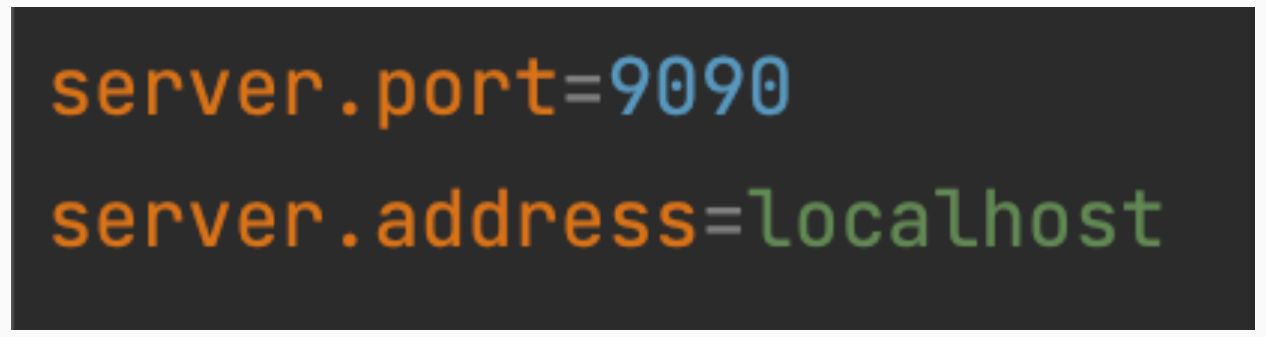
Spring Initializr generates an empty **application.properties** file in the **src/main/resources** folder.

This file lets you to externalize various application configurations in a **key-value** pair format.

You can check common properties on the Spring Boot website.
(https:// docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html)

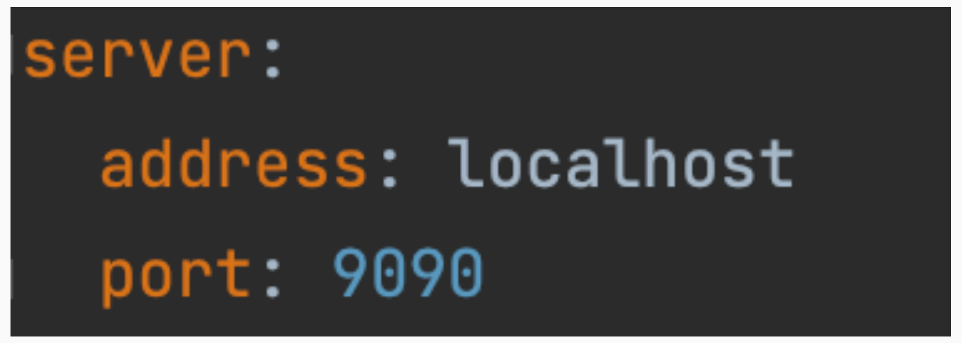


```
resources
    static
    templates
    application.properties
    application.yml
```



```
server.port=9090
server.address=localhost
```

application.properties

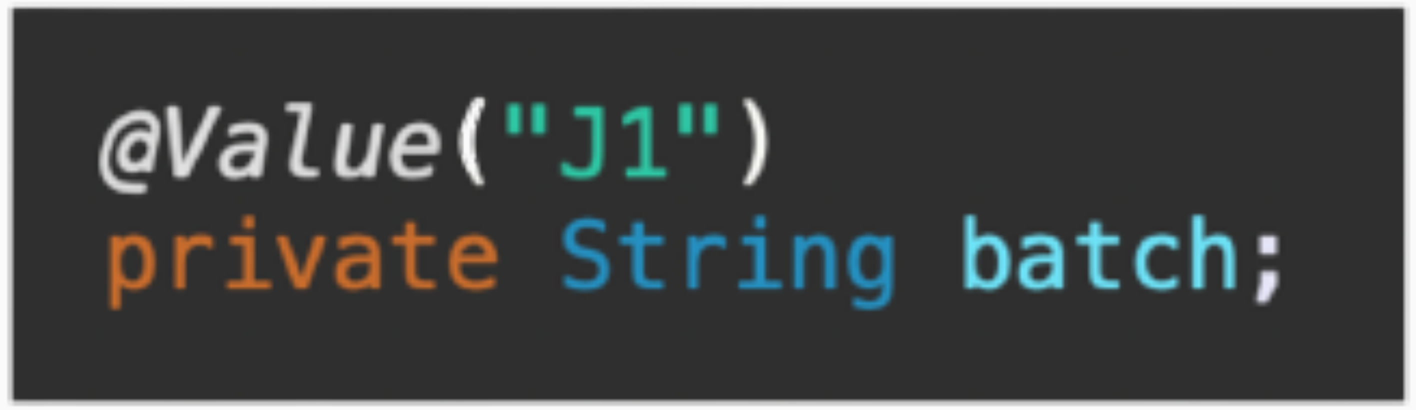


```
server:
  address: localhost
  port: 9090
```

application.yml

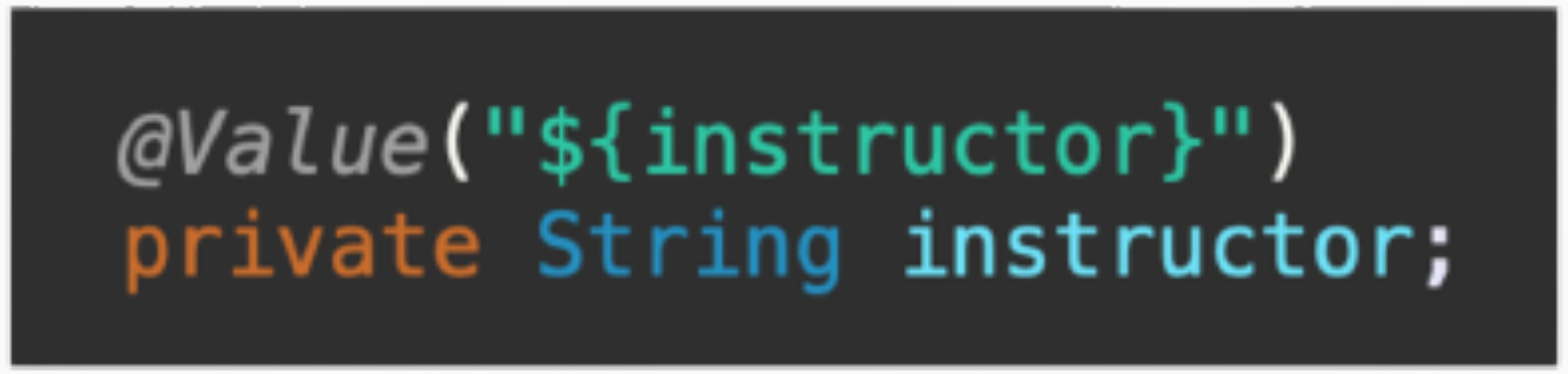
@Value Annotation

@Value can be used for injecting values into fields in Spring-managed beans and it can be applied at the field or constructor/method parameter level.



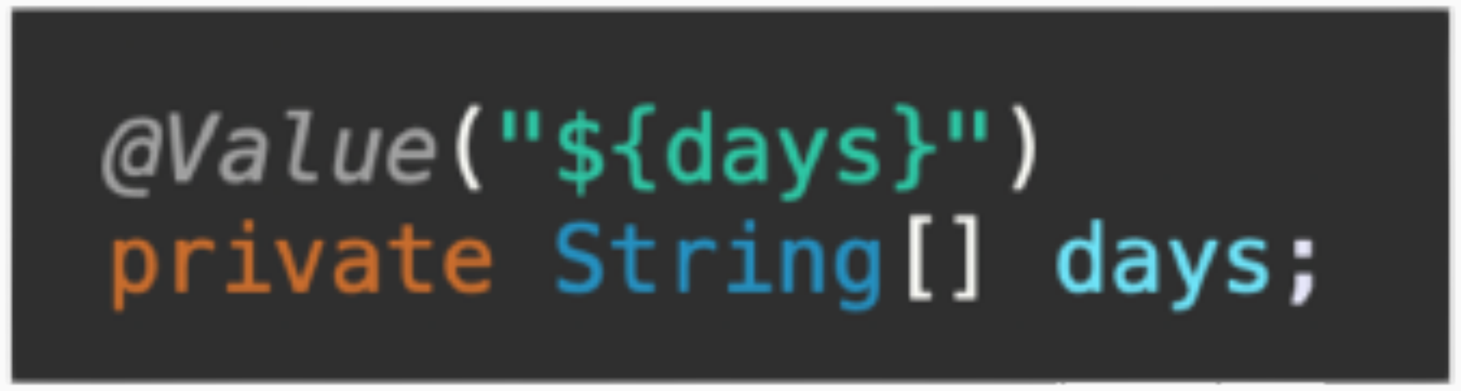
```java
@Value("J1")
private String batch;
```

Injecting from annotation field



```java
@Value("${instructor}")
private String instructor;
```

Injecting from properties files



```java
@Value("${days}")
private String[] days;
```

Injecting bunch of values from properties file