
Customer agrees that the [Professional Services Terms and Conditions](#) and the [Education And Training Services Terms Of Use](#) are incorporated by reference into this Data Sheet and shall govern the provision of the VMware Tanzu Lab Services and content accessible from this page. Customer may not record or reproduce the training in any medium. Customer may not copy, reproduce, or distribute or otherwise share the training materials in any capacity.

INSTRUCTIONS

Initializing a Spring Project

Purpose

This document will outline two ways to initialize your Spring Boot projects:

- Spring Initializr
- Manual creation of build files

Developers may want to use the Spring Initializr to bootstrap their Spring applications projects if they are not familiar with the starter BOMS, or their organizations may compel them to use a custom Initializr for security or legal compliance reasons.

Developers familiar with the Spring Boot starters BOMs and/or their dependencies may wish to manually create their build files and manually configure.

Spring Initializr

Business Case

While Spring Boot gives a robust framework to simplify dependency management and configuration of Spring applications, the core framework does not give a way to bootstrap the application development projects.

Maven provides a feature called "archetype" that has similar purpose, to allow building instance of applications from templates. It falls short, as it does not handle all the concerns of modern Enterprise development.

The Spring Initializr is a framework for curating the creation of Spring Boot application projects:

- Curate available Spring Boot Versions for use in an organization's Spring projects.
- Curate the available starters (BOMs) for a Spring Boot application. These may include what Spring Boot supplies out-of-box, or perhaps custom starters.
- Restrict and/or identify source repositories for BOMS (for Enterprise companies this will be an internal Maven repository, not Maven Central).
- Select available starter dependencies by an intuitive component name or tag.

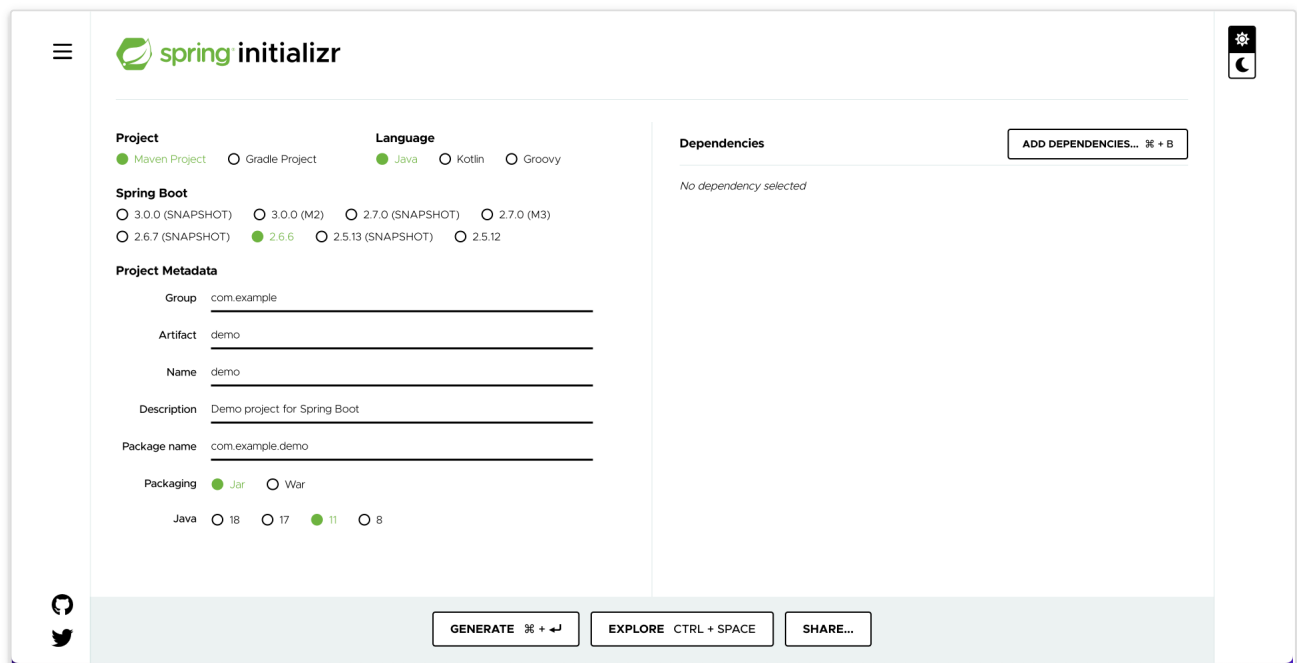
You may also build you own Spring Initializr site and APIs, see the [Spring Initializr Documentation](#) for more information.

Spring Starter Web Site

If you prefer to generate a Spring Boot project without aid of an IDE, you may leverage the [Spring Initializr website](#).

Getting started

1. Post the Spring Initializr (here we are using the [Spring Starter Site](#)). You will see the following:



The screenshot shows the Spring Initializr web interface. The header includes the Spring Initializr logo and a hamburger menu icon on the left, and a settings icon on the right. The main content area is divided into three sections: Project, Language, and Dependencies. The Project section has radio buttons for Maven Project (selected) and Gradle Project. The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for various versions, with 2.6.6 selected. The Project Metadata section includes input fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), and Package name (com.example.demo). The Packaging section has radio buttons for Jar (selected) and War. The Java section has radio buttons for versions 18, 17, 11 (selected), and 8. The Dependencies section has a button to add dependencies and a message stating 'No dependency selected'. At the bottom, there are three buttons: GENERATE (with a keyboard shortcut), EXPLORE (with a keyboard shortcut), and SHARE... (with a keyboard shortcut).

2. Choose the project type. Either *Maven* or *Gradle* are supported.

3. Choose the language type. The following are supported:

- Java
- Kotlin
- Groovy

4. Pick Spring Boot version.

Note that the spring initializr shows the latest snapshot, milestone and stable releases.

You will likely see slightly later versions.

5. Enter the *group* and *artifact* names of your new project.

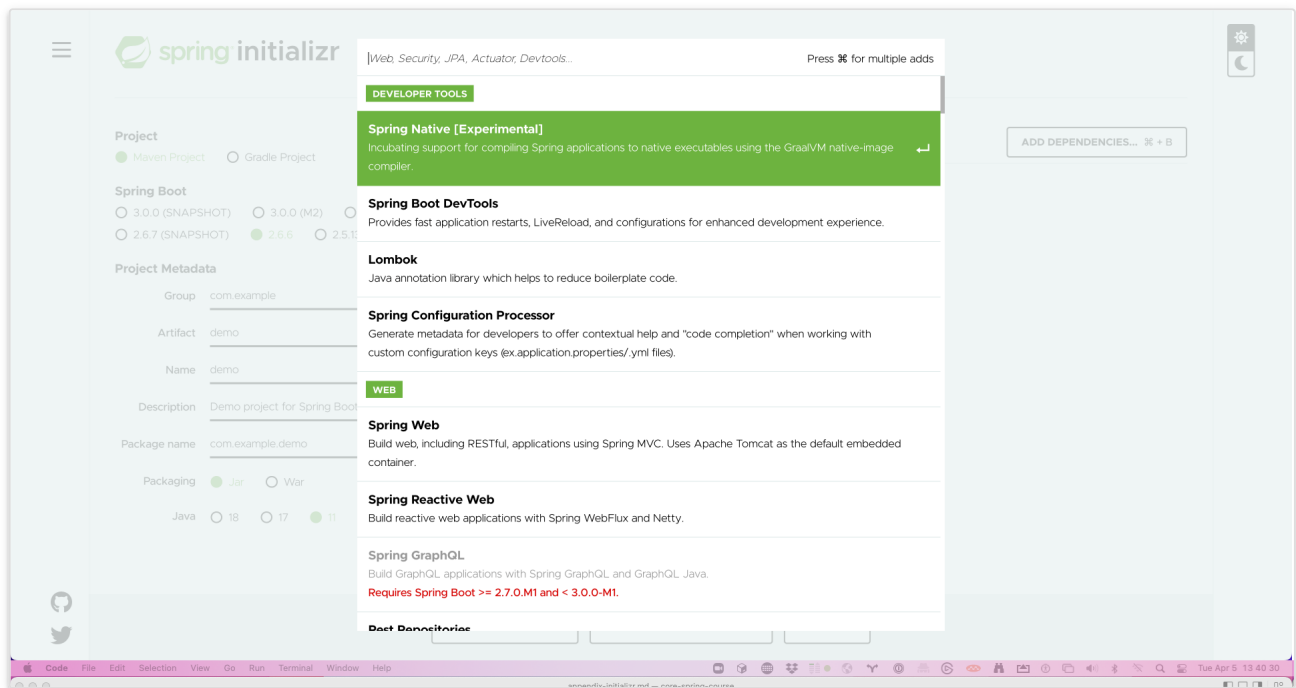
Dependencies

A key value point for the initializer is to help you select the dependencies you need for your boot application.

1. Click on the *Add Dependencies* button, or use the key board short cuts:

- Mac OS: *Cmd+B*
- Windows/Linux: *Ctrl+B*

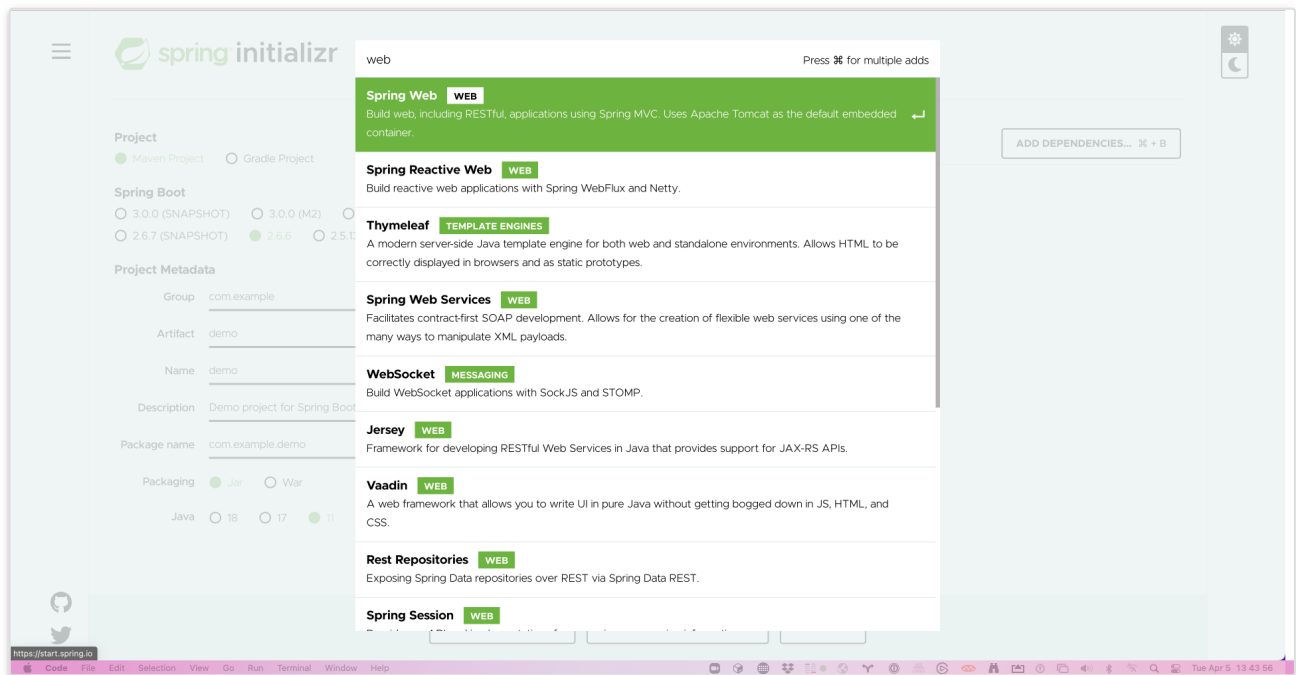
You will see a dialog with a list of available starters, with a entry box at the top to type in the search criteria for the starters you wish to add:



2. Enter the dependency tag(s) of what starter(s) you wish to build an application for in the *Dependencies >> Search for dependencies* box.

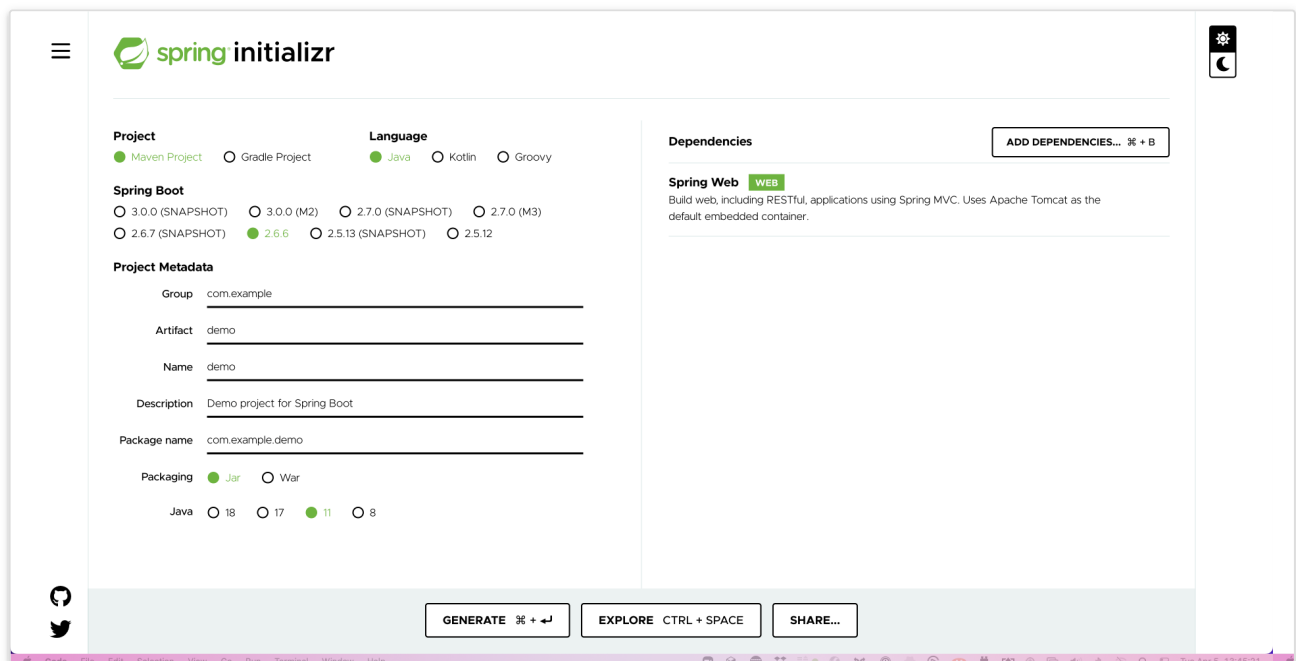
For example: *Web* is the tag for the `spring-boot-starter-web` BOM.

You will see the following when attempting to filter by *Web* tag:



This will select the starter and dependencies for a web application.

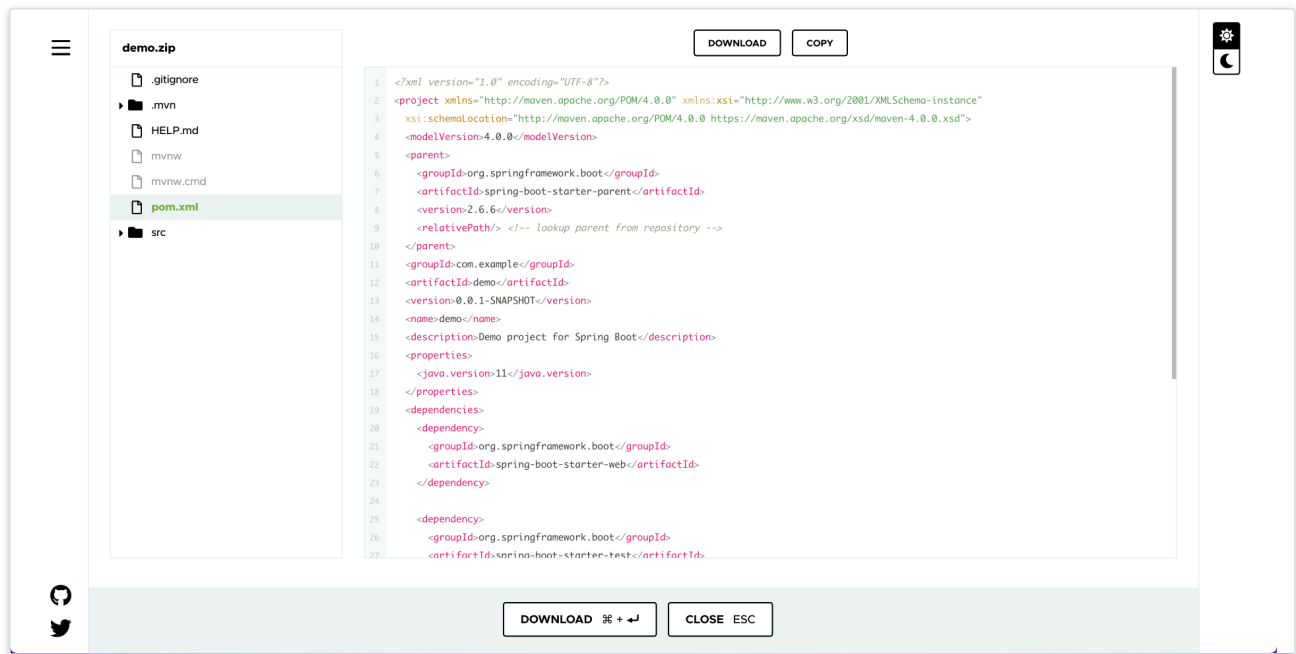
You will see the following selected tag:



Pre-reviewing the Spring Boot Project

1. Click on the *Explore* button, or use the hotkey *Ctrl+Space*.

You will see the following:



2. View the navigation tree on the left. The `pom.xml` is the default selected artifact. You can optionally view all the code and configuration artifacts that the initializer will generate for you.
3. View `pom.xml` in the details view. You can see the web starter that the initializer adds to the project on line 22.
4. When you are done reviewing the project, click the *Close* button, or type the *Escape* hotkey to exit back to the initializer front page.

Generate the Spring Boot Project

1. Click *Generate* button, or type the hotkey:
 - Mac OS: `Cmd+Return`
 - Windows or Linux: `Ctrl+Return`
2. A zip file will be generated and downloaded to your machine with the following contents:
 - Build file according to the appropriate build tool (i.e. `pom.xml` or `build.gradle`).
The build file contains the `spring-boot-starter-parent`, the starter dependencies for the category you selected, as well as the `spring-boot-starter-test` test starter dependency.
 - A `src` directory containing a Spring Boot application file, a Spring Boot test shell file, and an empty `application.properties` file.
3. This structure may be imported into either IDEA IntelliJ or Eclipse/ STS IDEs.

Spring Initializr IDE Support

STS

See here for instructions using the [New Spring Boot Starter Wizard](#).

IDEA IntelliJ

See the IntelliJ IDEA Guide [Creating your Spring Boot Project](#) article.

Manual Configuration of Project Files

Configuring a Spring boot build file requires a few conceptual steps:

1. Generate a standard Java build file with a Maven Repository that hosts or proxies all the project dependencies.
2. Adding a reference to a "parent" BOM or Spring Boot version.
3. Adding a Spring Boot plugin, which handles packaging and dependency management of your application.
4. Adding the Spring Boot starter dependencies, or (if you are feeling powerful, curate your dependencies by hand).

Maven

The Spring Boot application Maven POM requires (in addition to the standard Java application requirements):

1. A parent of `spring-boot-starter-parent` - This results in subsequent dependencies rooting from the parent specified Spring version.
2. Spring Boot Maven Plugin `spring-boot-maven-plugin` - It derives its version from the parent.
3. Starter dependencies (or hand-crafted dependencies)

An example `pom.xml` might look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.o
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.5</version>
```

```

    <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.example</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo</name>
<description>Demo project for Spring Boot</description>
<properties>
    <java.version>11</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

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

<build>

```

Gradle

The Spring Boot application Gradle build file requires (in addition to the standard Java application requirements):

1. Spring Boot Gradle Plugin `spring-boot-gradle-plugin`.

management' plugins.

3. Starter dependencies (or hand-crafted dependencies) - The starter dependencies derive their versions from the 'io.spring.dependency-management' plugin.

An example `build.gradle` file might look like this:

```

plugins {
    id 'org.springframework.boot' version '2.7.5'
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'
    id 'java'

```

```

}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.named('test') {
    useJUnitPlatform()
}

```

with its `settings.gradle` containing the `rootProject.ame` like this:

```
rootProject.name = 'demo'
```

BOMs, their dependencies, and dependency management

While this article should get you up and running with Spring Boot starters, it is worth warning that it is imperative to understand both the direct and transitive dependencies of the Spring Boot and Spring Boot derived projects (for example, Spring Cloud).

When dealing with refactoring or decomposition of complex component-based projects, it may be necessary in some cases to forgo use of the starters and curate dependencies by hand.

it is worth getting familiar with use of dependency management tools associated with your build system.

For Maven, the `dependency` [plugin](#), and in particular the `dependency:tree` goal will be your friend.

For Gradle, check out the following tasks:

- `dependencies`, see [Managing Dependencies](#)

- `dependencyManagement`, part of [Spring Dependency Management plugin](#)