Search...

Spring Initializr

Last Updated : 06 Aug, 2025

Spring Initializr is a popular tool for quickly generating **Spring Boot projects** with essential **dependencies**. It helps developers set up a new application with minimal effort, supporting Maven and Gradle builds. With its user-friendly interface, it simplifies project configuration, making it an essential resource for modern Spring Boot development.

Spring Initializr web-based tool provides simple web UI to generate the Spring Boot project structure or we can say it builds the skeleton of the Spring-based application. The spelling 'Initializr' is intentionally different from 'Initializer' to reflect its unique identity as a tool. It provides extensible APIs for creating JVM (Java Virtual Machine) based projects. Modern IDEs have integrated Spring Initializr which provides the initial project structure. The Spring Initializr tool takes care of the following configuration for any Spring-based project.

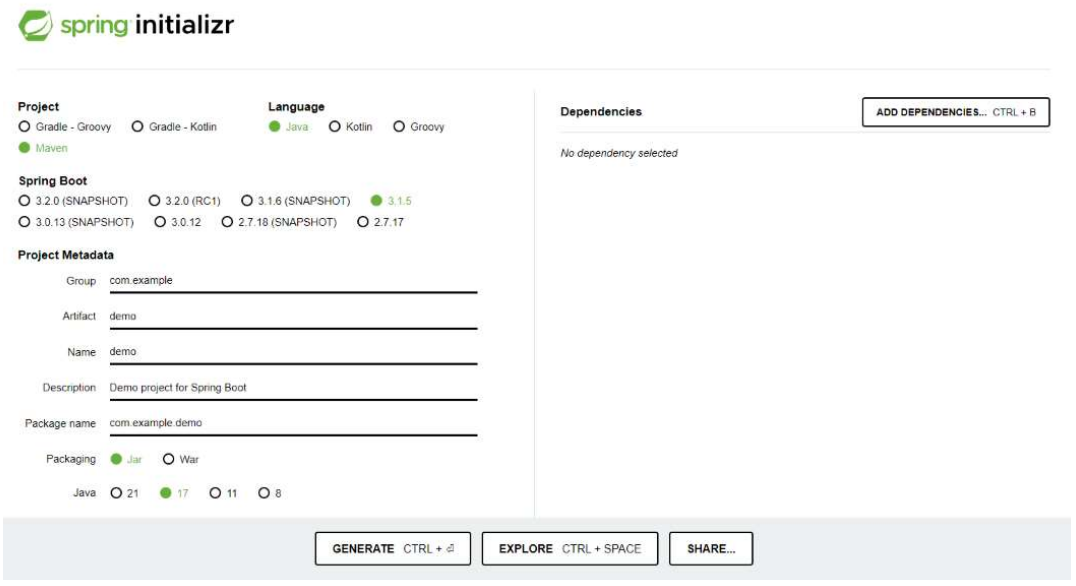
System Requirements

- An IDE (Spring Tool Suite(STS), IntelliJ IDEA Ultimate and IntelliJ IDEA Community Edition(Limited Configuration options), Netbeans, and VSCode) OR CLI.
- Java 17 or later Version
- Build tool (Maven repository or Gradle repository) to build the application.
- Spring Boot version (Dependencies are added based on the version)
- Dependencies required for the project.
- Project Metadata like name, packaging (Jar or War), package name, etc.

With all the provided information, the Spring Initializr generates the Spring project structure. We can use the Spring Initializr either from the Web or IDE or Command-Line.

Using Spring Initializr Web UI

Let us learn how to use the Spring Initializr web UI to generate the Spring Boot project for which the prior step is navigating to start.spring.io to get it. The window that will pop up is depicted below which is as follows:



spring initializr

Project
Gradle - Groovy [ ] Gradle - Kotlin [ ] [✓ Maven]

Language
[✓ Java] Kotlin [ ] Groovy [ ]

Spring Boot
3.2.0 (SNAPSHOT) [ ] 3.2.0 (RC1) [ ] 3.1.6 (SNAPSHOT) [ ] [✓ 3.1.5]
3.0.13 (SNAPSHOT) [ ] 3.0.12 [ ] 2.7.18 (SNAPSHOT) [ ] 2.7.17 [ ]

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	21 [✓ 17] 11 8

Dependencies

No dependency selected

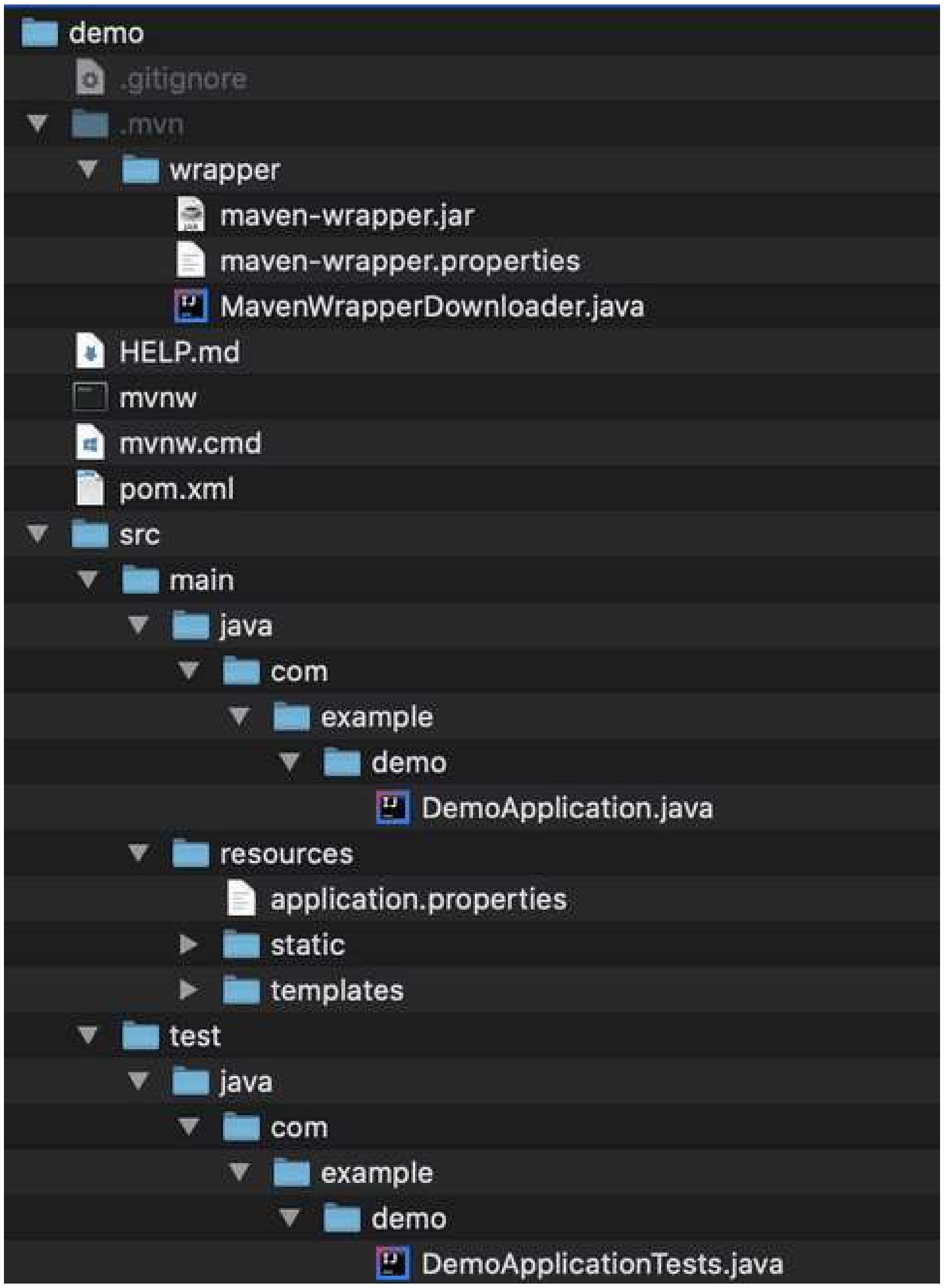
The Spring Initializr UI has the following options:

- **Project**: Using this one can create Maven or Gradle project i.e; Maven or Gradle can be used as a build tool. The default option is Maven Project. Maven project is used in the entire tutorial.
- **Language**: Spring Initializr provide Java, Kotlin and Groovy as a programming language for the project. Java is the default option.
- **Spring Boot Version**: Using this one can select the Spring Boot version for their project. Spring Boot latest Version is 3.1.5. The SNAPSHOT versions are under development and are not stable.
- **Project Dependencies**: Dependencies are artifacts that we can add to the project.
- **Project Metadata**: It is the information about the project. Information in the metadata does include below key points:

- **Group ID**: It is the ID of the project group.
- **Artifact**: It is the name of the Application.
- **Name**: Application name.
- **Description:** About the project.
- **Package name**: It is the combination of Group and Artifact Id.
- **Packaging**: Using this Jar or War packaging can be selected
- **Generate**: When generate option is clicked, the project is downloaded in zip format. The zip file can be unzipped and the project can be loaded into an IDE.
- **Explore**: This allows to see and makes changes in the generated project.

Project Structure

The Spring Boot project looks as follows:



```
demo
    .gitignore
  ▼ .mvn
      ▼ wrapper
          maven-wrapper.jar
          maven-wrapper.properties
          MavenWrapperDownloader.java
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
  ▼ src
      ▼ main
          ▼ java
              ▼ com
                  ▼ example
                      ▼ demo
                          DemoApplication.java
          ▼ resources
              application.properties
            ▶ static
            ▶ templates
      ▼ test
          ▼ java
              ▼ com
                  ▼ example
                      ▼ demo
                          DemoApplicationTests.java
```

Ways to Use Spring Initializr

Spring Initializr can be achieved via two ways which are as follows:

Spring Initializr using IDE's

Spring Initializr is supported by various IDEs like Spring Tool Suite(STS), IntelliJ IDEA Ultimate and IntelliJ IDEA Community Edition(Limited Configuration options), Netbeans, and VSCode.

- If you are using Netbeans, **Spring Initializr plugin** can be added to the IDE.
- If you are using VSCode, **vscode-spring-initializr** plugin can be added to the VSCode.

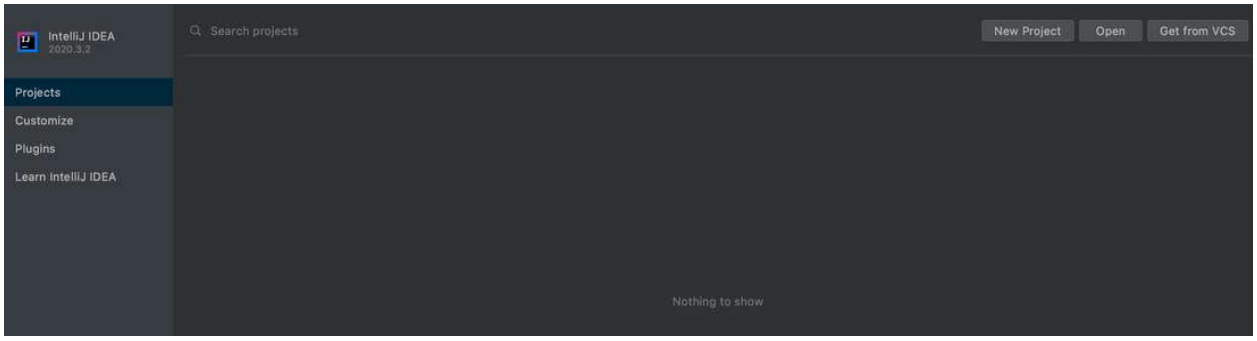
1. Using IntelliJ Community Edition

Steps to be followed:

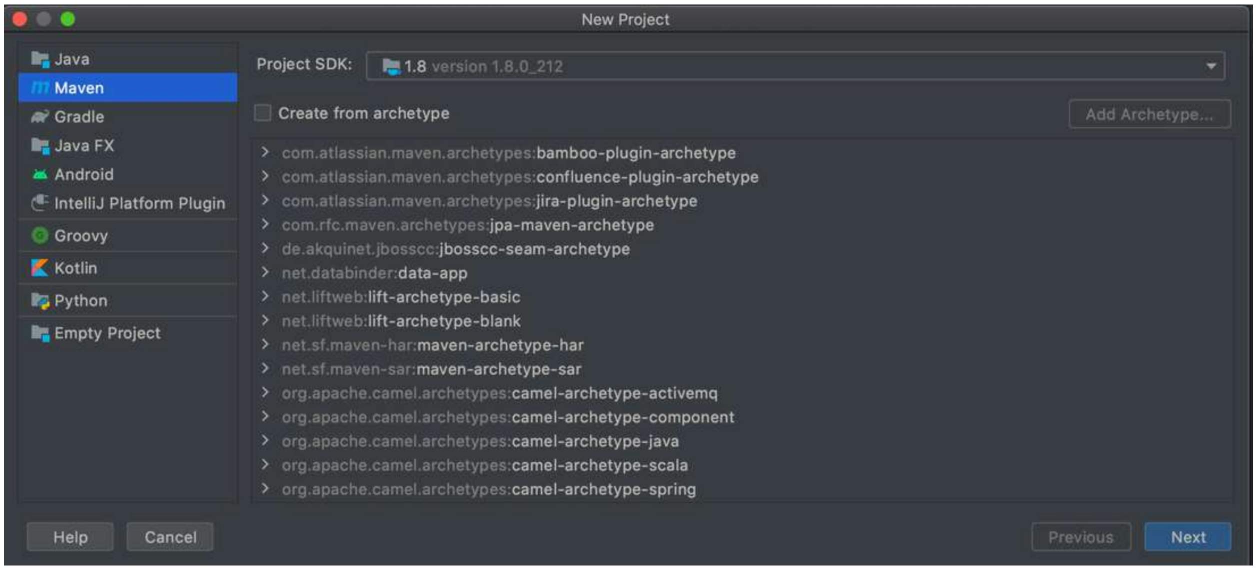
1. Open IntelliJ CE IDE.
2. Click on New Project.
3. Select Maven Project and Java JDK Home path
4. Enter Project Metadata details
5. Click the Finish button.

Let us pictorially depict the above steps to get a fairer understanding

- Open IntelliJ CE IDE.
- Click on New Project.



Select Maven Project and Java JDK Home path.



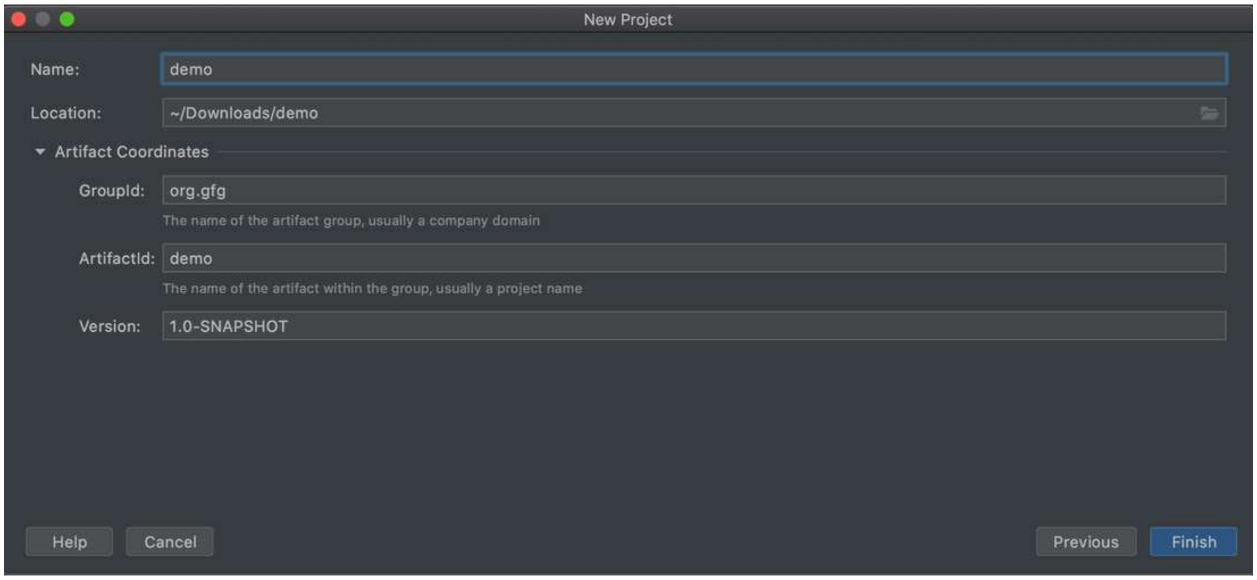
Project SDK: 1.8 version 1.8.0_212

Create from archetype

Add Archetype...

- com.atlassian.maven.archetypes:bamboo-plugin-archetype
- com.atlassian.maven.archetypes:confluence-plugin-archetype
- com.atlassian.maven.archetypes:jira-plugin-archetype
- com.rfc.maven.archetypes:jpa-maven-archetype
- de.akquinet.jbosscc:jbosscc-seam-archetype
- net.databinder:data-app
- net.liftweb:lift-archetype-basic
- net.liftweb:lift-archetype-blank
- net.sf.maven-har:maven-archetype-har
- net.sf.maven-sar:maven-archetype-sar
- org.apache.camel.archetypes:camel-archetype-activemq
- org.apache.camel.archetypes:camel-archetype-component
- org.apache.camel.archetypes:camel-archetype-java
- org.apache.camel.archetypes:camel-archetype-scala
- org.apache.camel.archetypes:camel-archetype-spring

Enter Project Metadata details such as Name, GroupId, ArtifactId.



Name: demo

Location: ~/Downloads/demo

Artifact Coordinates

GroupId: org.gfg

The name of the artifact group, usually a company domain

ArtifactId: demo

The name of the artifact within the group, usually a project name

Version: 1.0-SNAPSHOT

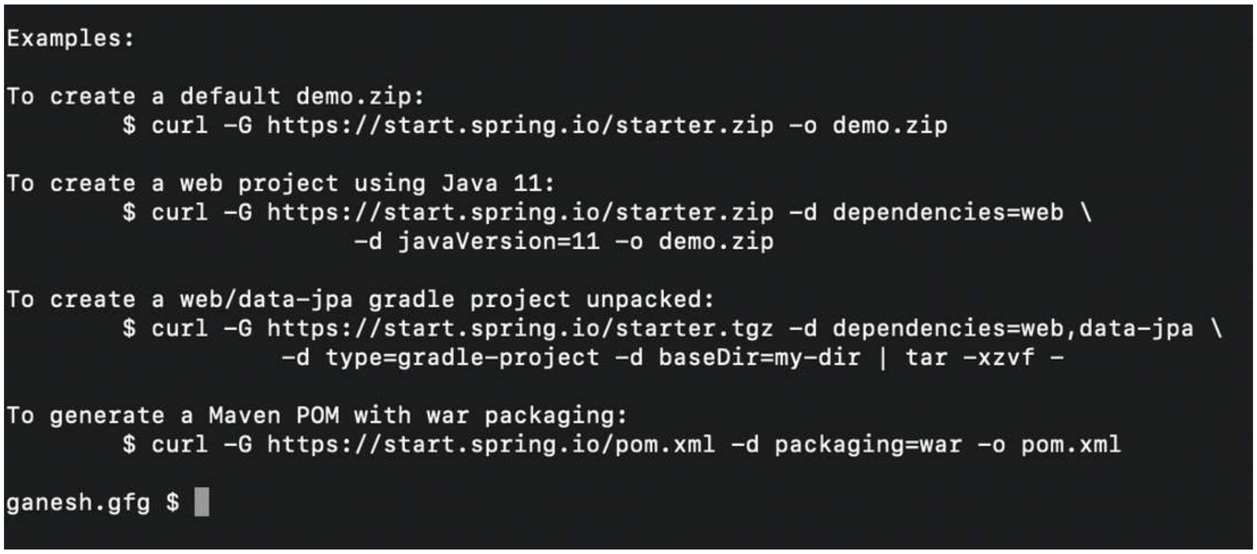
Click the Finish button.

> ***Note**: If you are using IntelliJ Ultimate Edition, the steps are available here.*

A lot of developers love to do things from the command line. For them, there is an option to create the Spring project using command-line utilities like Spring Boot CLI (Command Line Interface), cURL or HTTPie. To use cURL or HTTPie, one has to install them prior to the usage.

> *curl https://start.spring.io/*

The above command will give the entire instructions on how to create a project using cURL.



```
Examples:

To create a default demo.zip:
        $ curl -G https://start.spring.io/starter.zip -o demo.zip

To create a web project using Java 11:
        $ curl -G https://start.spring.io/starter.zip -d dependencies=web \
                        -d javaVersion=11 -o demo.zip

To create a web/data-jpa gradle project unpacked:
        $ curl -G https://start.spring.io/starter.tgz -d dependencies=web,data-jpa \
                   -d type=gradle-project -d baseDir=my-dir | tar -xzvf -

To generate a Maven POM with war packaging:
        $ curl -G https://start.spring.io/pom.xml -d packaging=war -o pom.xml

ganesh.gfg $ 
```

Let’s assume that you want to generate a demo.zip project based on Spring Boot 3.1.5 RELEASE, using the web and developer tools dependencies (remember, those two IDs are displayed in the capabilities of the service):

> *curl https://start.spring.io//starter.zip -d dependencies=web,devtools -d bootVersion=3.1.5 -o demo.zip*

The exact same project can be generated using the HTTP command as well:

> *http https://start.spring.io//starter.zip dependencies==web,devtools bootVersion==3.1.5 --download --output=demo.zip*

Note: Apart from all these, Spring Initializr teams also provide extensible APIs for creating JVM-based projects. Also, one can create their own instance of Spring Initializr for their own projects. It also provides various options for the project that are expressed in a metadata model. The metadata model allows us to configure the list of dependencies supported by JVM and platform versions, etc.