Appendix 02 Spring Initializr

Description: "Spring Initializr provides an extensible API to generate JVM-based projects, and to inspect the metadata used to generate projects, for instance to list the available dependencies and versions." -- Spring Initializr Reference Guide.

Project: Use start.spring.io to create a "web" project. In the "Dependencies" dialog search for and add the "web" dependency as shown in the screenshot. Hit the "Generate" button, download the zip, and unpack it into a folder on your computer.

This appendix adds details to the start.spring.io website when using the Spring Initializr. The zip file contains a very basic Spring Boot project including an independent Tomcat server. This zip file can be used to begin most Spring Boot projects.

Technology:
This project uses the following technology:
Spring Initialzr – from the start.spring.io website.

Table of Contents

Appendix: Spring Initializr

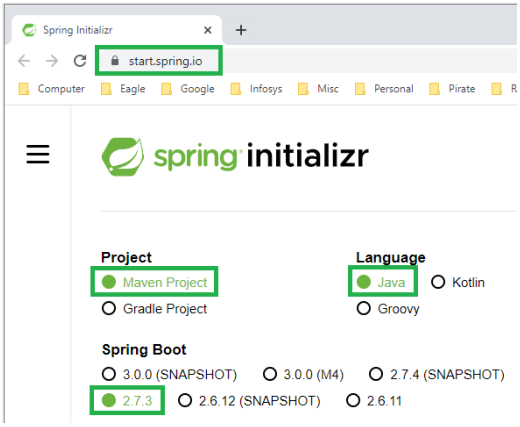
"Spring Initializr provides an extensible API to generate JVM-based projects, and to inspect the metadata used to generate projects, for instance to list the available dependencies and versions." -- Spring Initializr Reference Guide

Generate Spring Boot Download

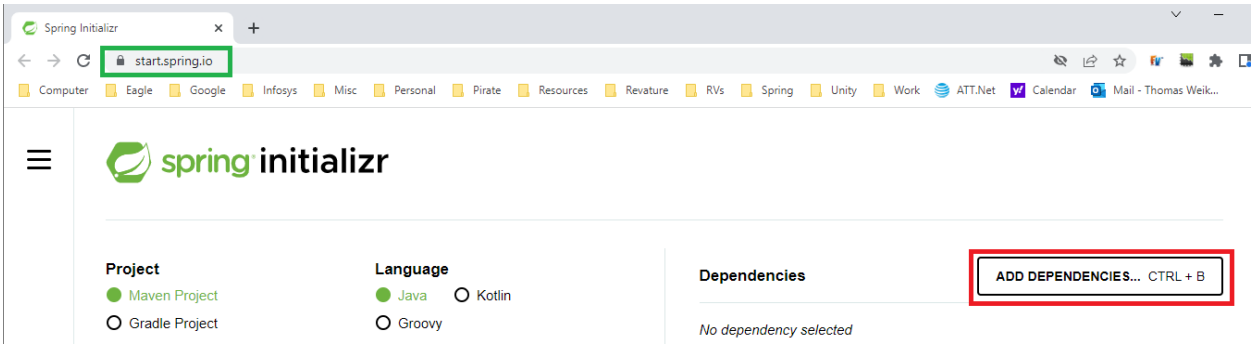
Use start.spring.io to create a "web" project. In the "Dependencies" dialog search for and add the "web" dependency as shown in the screenshot. Hit the "Generate" button, download the zip, and unpack it into a folder on your computer.

By default the following should be selected:

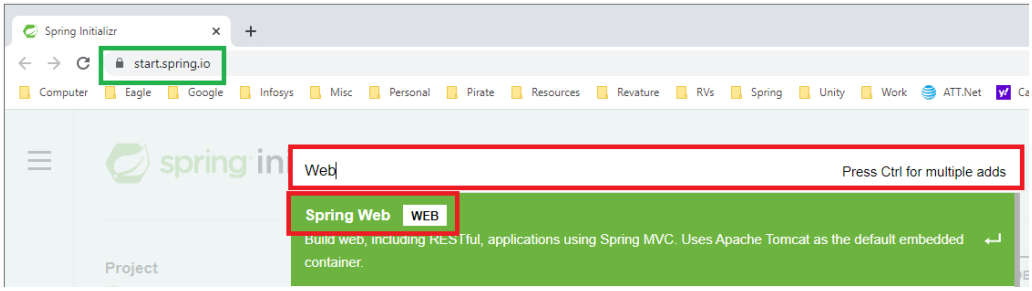
"Project": "Maven Project"
"Language": "Java"
"Spring Boot": latest non-snapshot version (2.7.3 when this document was written.)



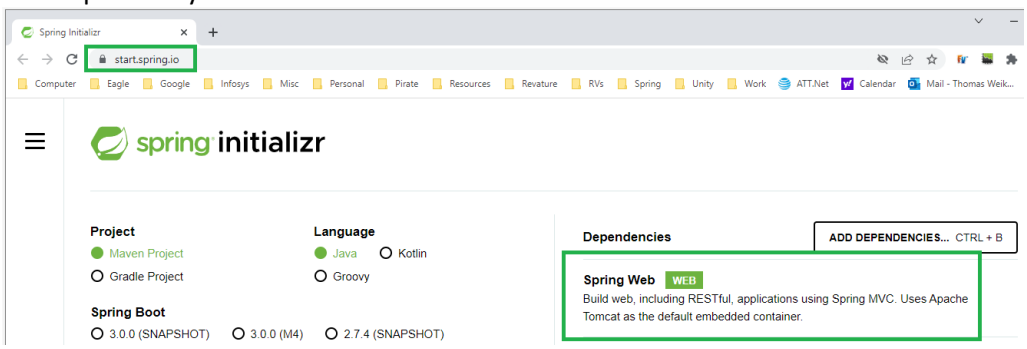
Click "ADD DEPENDENCIES…"



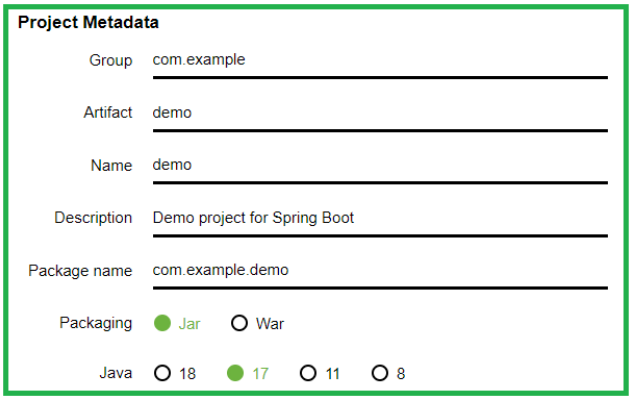
Enter “web” in search bar
Select “Spring Web”



The dependency is added for the build.



Enter “Project Metadata”



If the project document that reference using this document use values specified in the project document for all metadata shown above. For any values not supplied by the project document, use the values specified here.

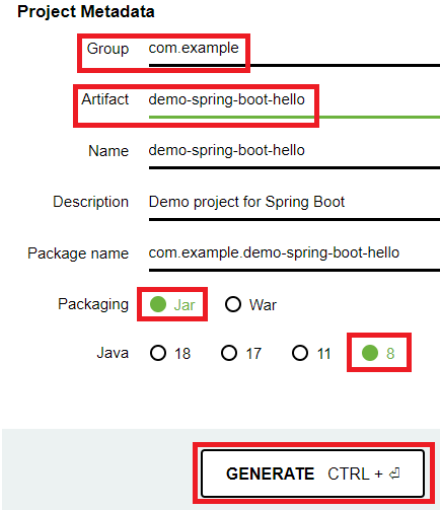
Default “Artifact”: “demo”
The project name should be changed to the specific project name being build. In this example we change the name to “**demo-spring-boot-hello**”. Changing the name here will generate unique project names and meaning project application classes.

Note: Changing the “Artifact” name will automatically change “Name” field and the “Package name”.

Default "Group": com.example
If you have a specific package domain you change it here under "Group". Changing group will automatically change "Package name".

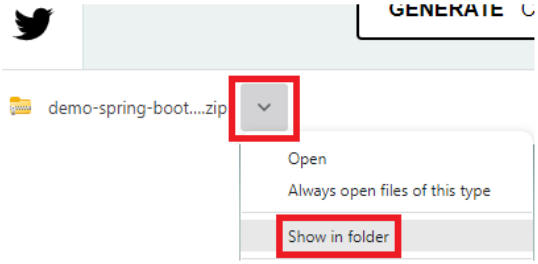
Default "Packaging": "Jar"
Select "Java": "8" – Java version 8 is used in various coaching demo documents.

Click "GENERATE"

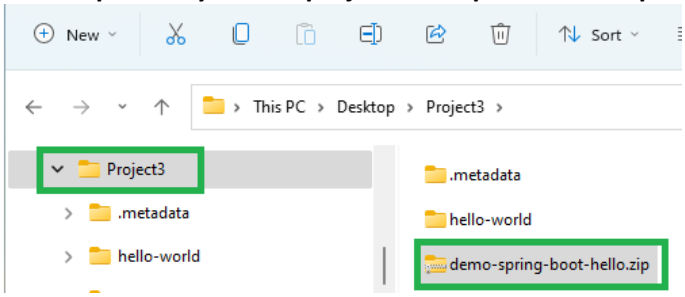


Extract the Generated Project from the Zip File

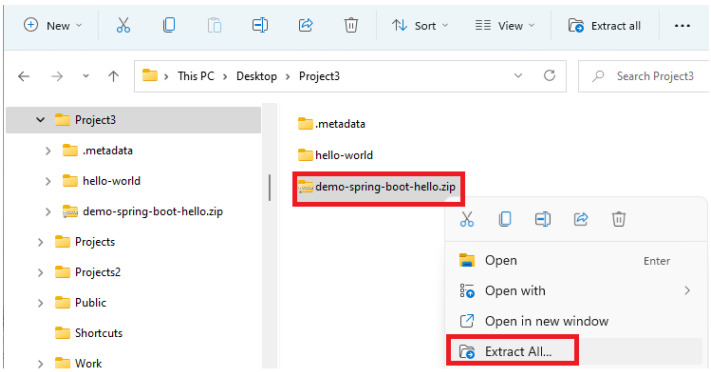
The browser used and browser settings might perform the download differently.
You will want to find the file created for your project. This example is "demo-spring-boot.zip".



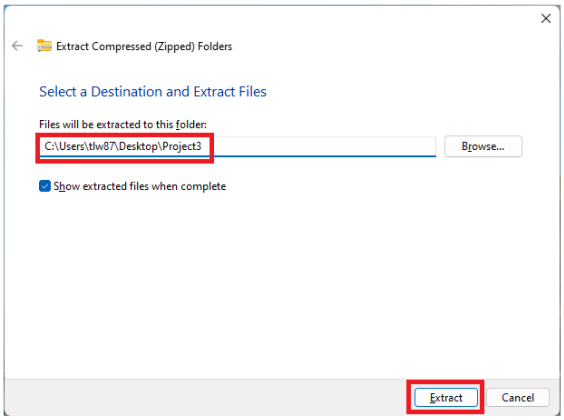
Move zip file to your IDE project workspace and unzip the file.



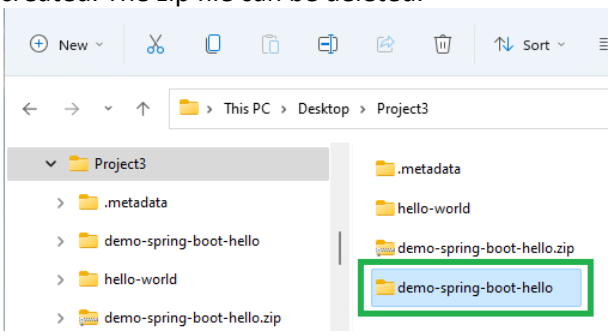
As with the browser, your unzip program might be different than what is shown below.
Right Click zip file: your project name. This example "demo-spring-boot.zip".
Select "Extract All…"



Verify / change extract folder to your IDE workspace top level folder.
In this example the folder will extract to "C:\Users\tlw87\Desktop\Project3\demo-spring-boot-hello" by creating sub folders "\demo-spring-boot-hello\demo-spring-boot-hello" two levels deep. We only want one level deep so change the extract to your top-level workspace folder like "C:\Users\tlw87\Desktop\Project3"
Click "Extract"

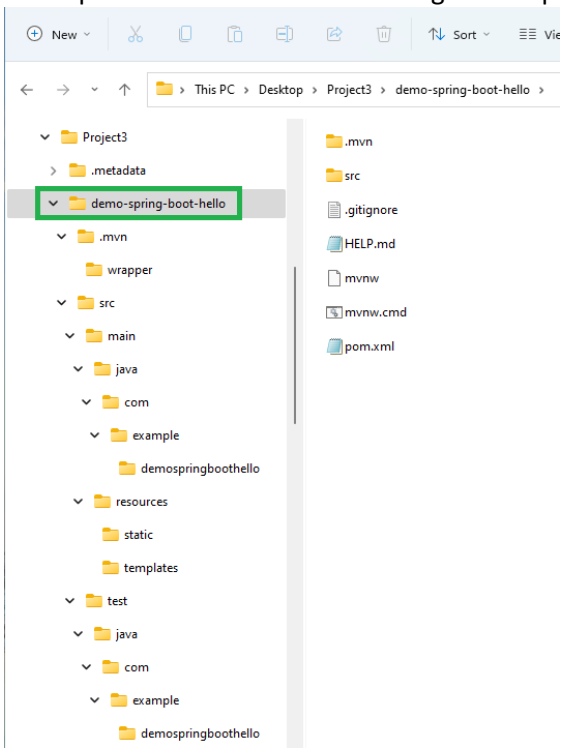


If the "Show extracted files …" was left checked, an explorer window will open showing the folder created. The zip file can be deleted.



Spring Boot Zip File Folder Structure

Once the zip file is extracted the folder structure aligns with a Spring Boot IDE project with Maven. The top-level folder is the name we gave our project: “demo-spring-boot-hello”

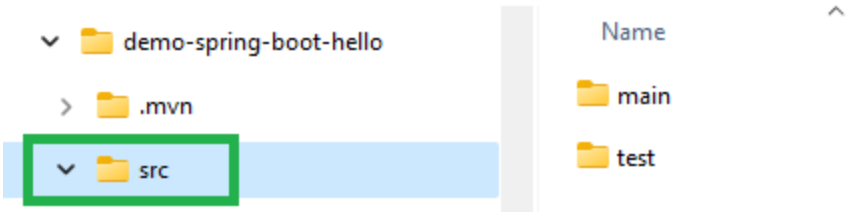


Spring Boot Sub-Folders

Sub-folders are:

.mvn - containing the Maven wrapper files. Inside another subfolder “wrapper” is an executable jar file which will contain an Apache Tomcat server which Spring Boot application will run on. The Tomcat server inside the IDE will not be used in a Spring Boot application.

src – the source folder is a typical IDE project folders containing the Java source files for the application and folders for test Java source files.



Spring Boot Files

The files are:

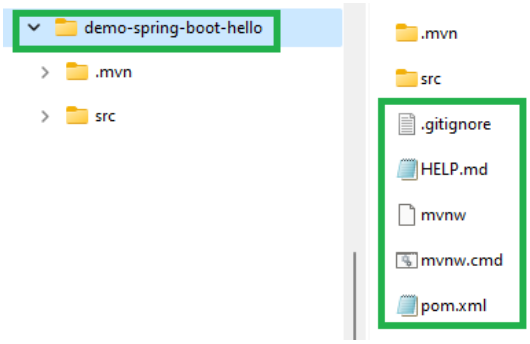
.gitgnore – This file is related to using Git Hub. This file tells Git which files to ignore when committing your project to the GitHub repository.

.md (HELP.md) - MD files are saved in plain text format that uses Markdown language. The HELP.md file is just that a help file or Read Me First file.

mvnw - it's an executable Unix shell script used in place of a fully installed Maven. This calls the executable jar in the **.mvn\wrapper** folder.

mvnw.cmd - it's for Windows environment. This calls the executable jar in the **.mvn\wrapper** folder.

pom.xml - The Maven Project Object Model (POM) XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

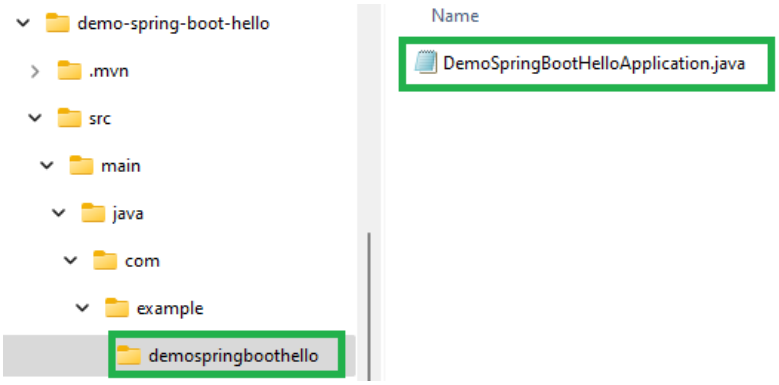


Spring Boot Package Folders

The package name and the project name may vary depending on the project you are creating.

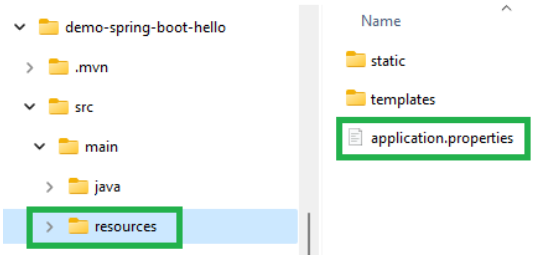
In the src folder structure is the package we defined when creating the zip file: “com.example”.

DemoSpringBootHelloApplication.java – was generated as the class to run the application containing the public static void main(String[] args){} method.



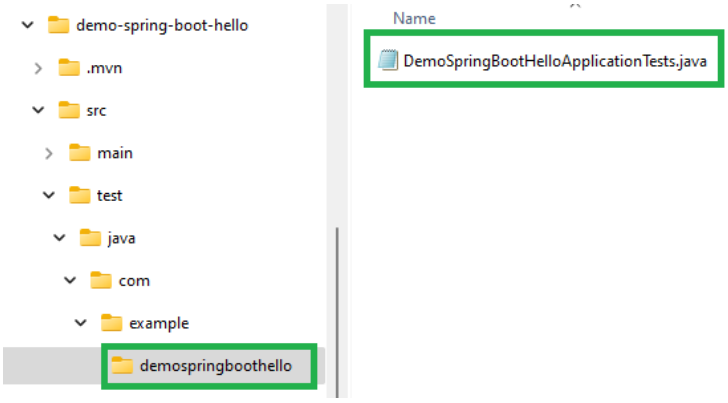
In the resource folder there is a property file shell, it is created as a placeholder, an empty file.

> **application.properties** - This file contains the different configuration which is required to run the application in a different environment, and each environment will have a different property defined by it. It can contain database related properties such as connection string, username, password, and other properties.



A default test package was also created with “com.example” for unit testing.

> **DemoSpringBootHelloApplicationTests.java** – was generated with class annotation @SpringBootTest to identify as a test class.



Import Spring Boot Project

The next step is to import the project created with the Spring Initializr into your IDE. The document title "**Appendix 03 Import Project**" covers the import process for Spring Tool Suites IDE used in various coaching documents created for Spring Boot projects.