

# 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 Initializr – from the [start.spring.io](#) website.

Unzip program – within Window File Explorer

## Revision History

Date	Description
08/30/2022	Initial document created.
04/27/2023	Document updates related to version 3.0.6 of Spring Initializr tool.

## Glossary of Terms

Common terminology used with this set of documents is found in the document “[Appendix 01 Glossary](#)”.

## Table of Contents

Appendix: Spring Initializr .....	3
Generate Spring Boot Download .....	4
Extract the Generated Project from the Zip File.....	7
Spring Boot Zip File Folder Structure .....	9
Spring Boot Sub-Folders.....	9
Spring Boot Files.....	10
Spring Boot Package Folders.....	10
Import Spring Boot Project .....	11

## 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](#)

The 3.0.6 version of the Spring Initializr used in this document looks like the following with default values indicated.

The screenshot shows the Spring Initializr web interface in a browser window. The URL bar shows `start.spring.io`. The interface is divided into several sections:

- Project:** Includes radio buttons for `Gradle - Groovy` (selected), `Gradle - Kotlin`, and `Maven`.
- Language:** Includes radio buttons for `Java` (selected), `Kotlin`, and `Groovy`.
- Spring Boot:** Includes radio buttons for versions: `3.1.0 (SNAPSHOT)`, `3.1.0 (RC1)`, `3.1.0 (M2)`, `3.0.7 (SNAPSHOT)`, `3.0.6` (selected), `2.7.12 (SNAPSHOT)`, and `2.7.11`.
- Project Metadata:** Includes text input fields for `Group` (com.example), `Artifact` (demo), `Name` (demo), `Description` (Demo project for Spring Boot), and `Package name` (com.example.demo).
- Packaging:** Includes radio buttons for `Jar` (selected) and `War`.
- Java:** Includes radio buttons for versions: `20`, `17` (selected), `11`, and `8`.
- Dependencies:** Includes a button `ADD DEPENDENCIES... CTRL + B` and the text `No dependency selected`.
- Footer:** Includes buttons `GENERATE CTRL + G`, `EXPLORE CTRL + SPACE`, and `SHARE...`.

## Generate Spring Boot Download

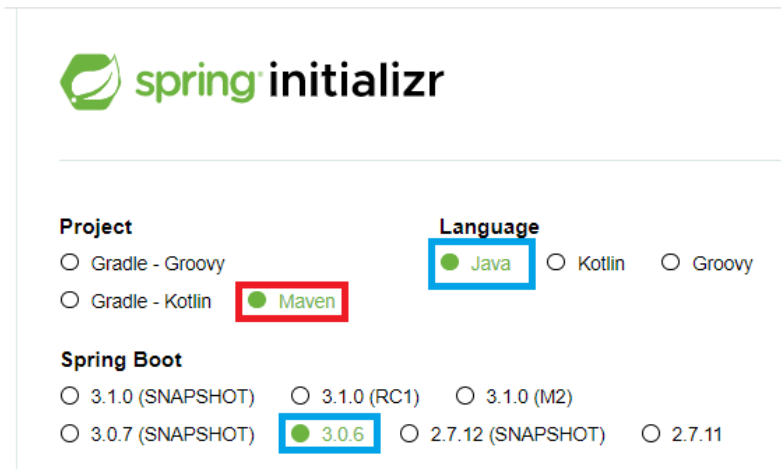
Use [start.spring.io](https://start.spring.io) to create a “web” project. In the “Dependencies” dialog search for and add the “web” dependency as shown in the screenshot. Hit the “Generate” button, download the zip, and unpack it into a folder on your computer.

The following should be selected to use for project documents referencing this appendix:

“Project”: “Maven Project”

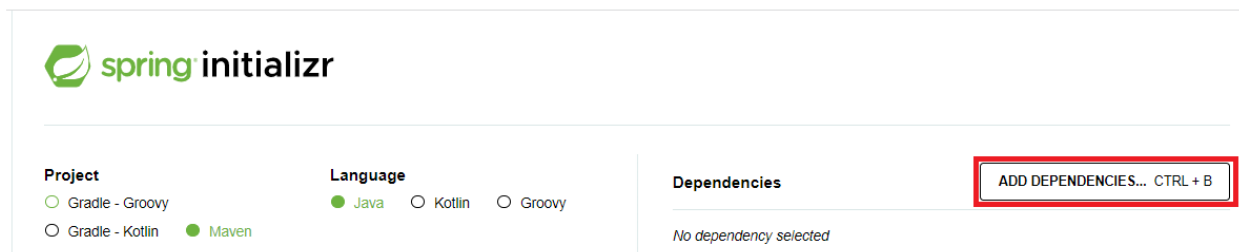
“Language”: “Java”

“Spring Boot”: latest **non-snapshot** version (3.0.6 when this document was updated.)



The screenshot shows the Spring Initializr configuration form. The 'Project' section has 'Maven' selected and highlighted with a red box. The 'Language' section has 'Java' selected and highlighted with a blue box. The 'Spring Boot' section has '3.0.6' selected and highlighted with a blue box.

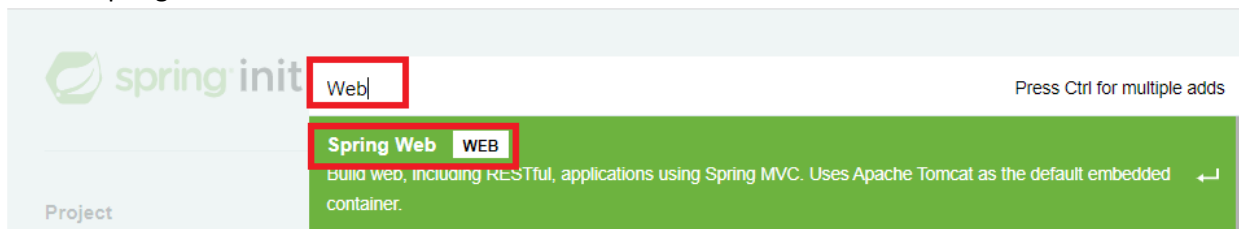
Click “ADD DEPENDENCIES...”



The screenshot shows the Spring Initializr Dependencies dialog. The 'ADD DEPENDENCIES... CTRL + B' button is highlighted with a red box. The 'Project' section has 'Maven' selected. The 'Language' section has 'Java' selected. The 'Dependencies' section shows 'No dependency selected'.

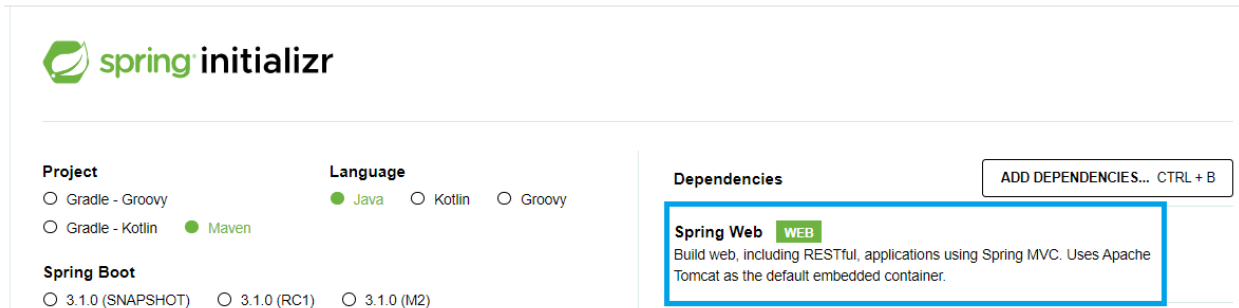
Enter “web” in search bar

Select “Spring Web”



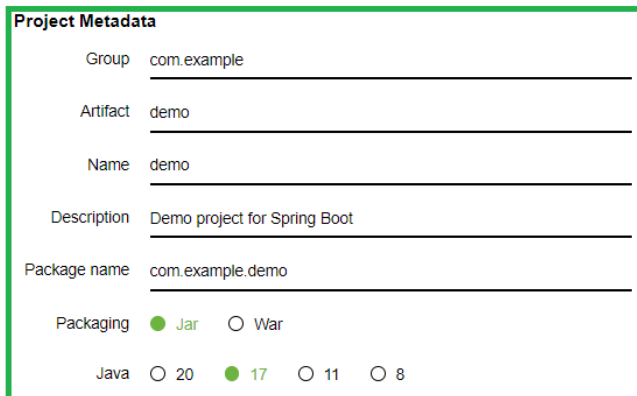
The screenshot shows the Spring Initializr search results. The search bar contains 'Web' and is highlighted with a red box. The search results list 'Spring Web' with a 'WEB' tag, and it is highlighted with a green box. The description for 'Spring Web' is 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'

The dependency is added for the build. Other dependencies might be specified in project document to add and / or replace the “Spring Web” dependency.



The image shows the Spring Initializr web interface. It has a header with the Spring logo and 'spring initializr' text. Below the header, there are three main sections: 'Project', 'Language', and 'Dependencies'. The 'Project' section has radio buttons for 'Gradle - Groovy', 'Gradle - Kotlin', and 'Maven' (which is selected). The 'Language' section has radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Spring Boot' section has radio buttons for '3.1.0 (SNAPSHOT)', '3.1.0 (RC1)', and '3.1.0 (M2)'. The 'Dependencies' section has a button 'ADD DEPENDENCIES... CTRL + B' and a list of dependencies. One dependency, 'Spring Web', is highlighted with a blue box. It has a green 'WEB' tag and a description: 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'

### Enter “Project Metadata”



The image shows the 'Project Metadata' form. It has a green border. The form contains the following fields: 'Group' with the value 'com.example', 'Artifact' with the value 'demo', 'Name' with the value 'demo', 'Description' with the value 'Demo project for Spring Boot', and 'Package name' with the value 'com.example.demo'. There are also radio buttons for 'Packaging' with 'Jar' selected and 'War' unselected. At the bottom, there are radio buttons for 'Java' with '20' unselected, '17' selected, '11' unselected, and '8' unselected.

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”

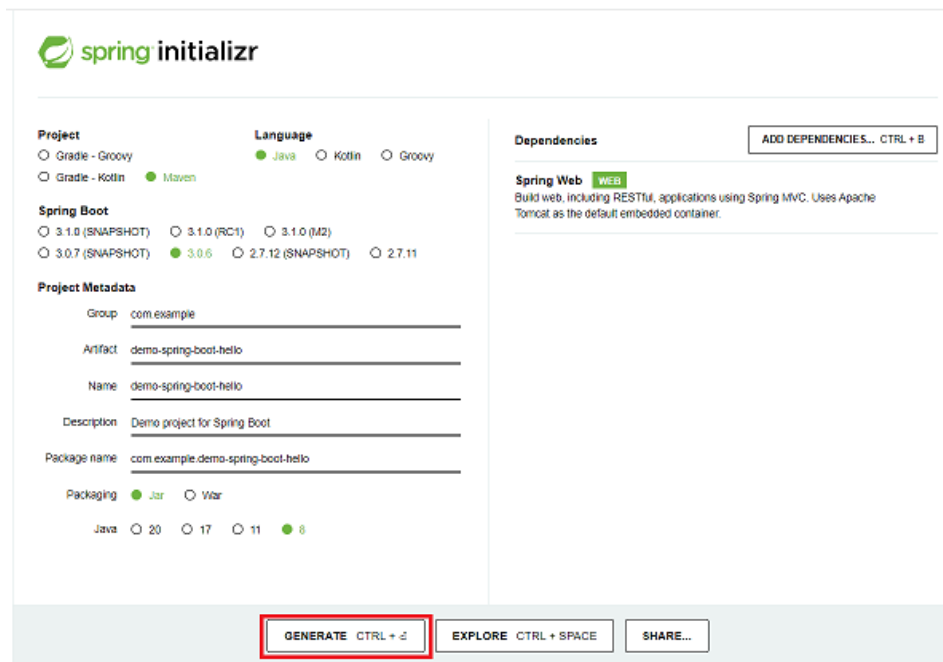
**Select “Java”: “8”** – Java version 8 is used in various project demo documents.

### Project Metadata

Group	<input type="text" value="com.example"/>
Artifact	<input type="text" value="demo-spring-boot-hello"/>
Name	<input type="text" value="demo-spring-boot-hello"/>
Description	<input type="text" value="Demo project for Spring Boot"/>
Package name	<input type="text" value="com.example.demo-spring-boot-hello"/>
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input type="radio"/> 20 <input type="radio"/> 17 <input type="radio"/> 11 <input checked="" type="radio"/> 8

**Note:** There appears to be a bug with this version of the Spring Initializr. Selecting Java version 8 (for 1.8) will build the project with the default Java 17. The document [“Appendix 03 Import Spring Tool Suite Project”](#) details how to switch to Java version 1.8. Java version 8 should still be selected here for when the issue with the Spring Initializr is resolved.

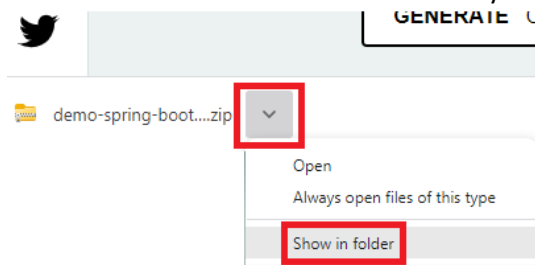
Click “GENERATE”



The image shows the Spring Initializr web interface. It has a header with the Spring logo and 'spring initializr' text. The main content is divided into three columns. The left column contains 'Project' (radio buttons for Gradle - Groovy, Gradle - Kotlin, and Maven), 'Spring Boot' (radio buttons for various versions, with 3.0.6 selected), and 'Project Metadata' (text input fields for Group, Artifact, Name, Description, and Package name, and radio buttons for Packaging and Java version). The middle column contains 'Language' (radio buttons for Java, Kotlin, and Groovy). The right column contains 'Dependencies' (a button to add dependencies) and 'Spring Web' (a button to add the Spring Web dependency). At the bottom, there are three buttons: 'GENERATE' (highlighted with a red box), 'EXPLORE', and 'SHARE...'. The 'GENERATE' button has the text 'CTRL + G' next to it.

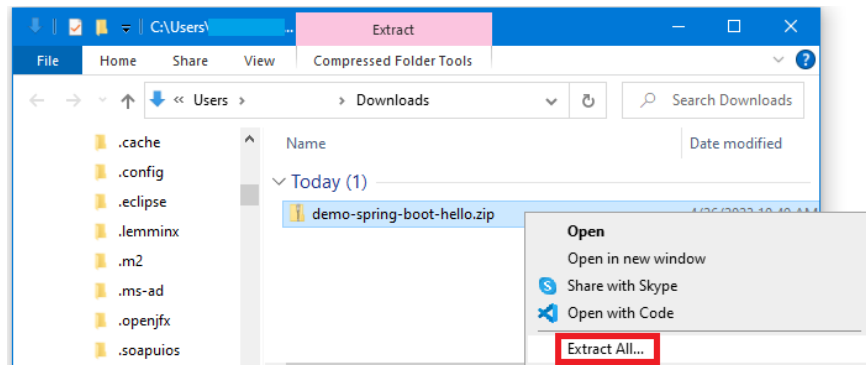
## Extract the Generated Project from the Zip File

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”.

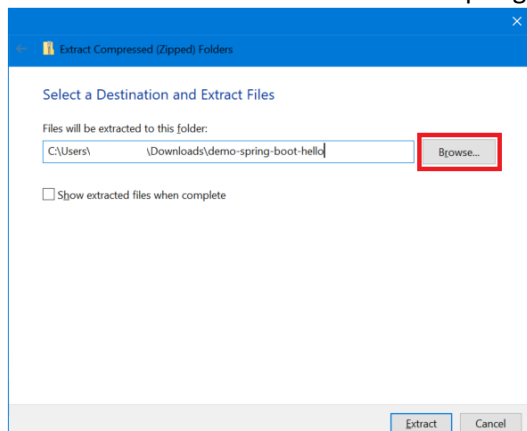


As with the browser, your unzip program might be different than what is shown below. Extract the files from the download location to your project file folder.

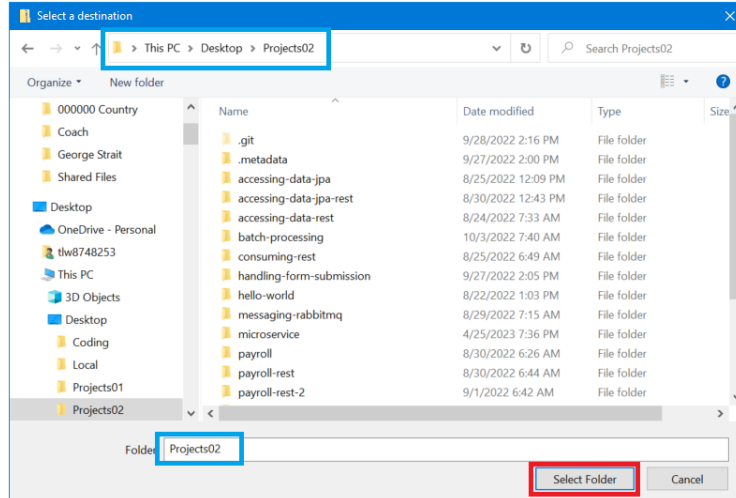
Right Click zip file: your project name. This example “demo-spring-boot.zip”.  
**Select “Extract All...”**



**Browse** to the folder location for the Spring Tool Suite.



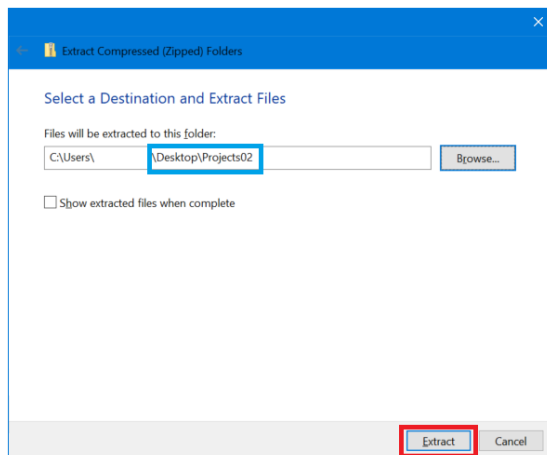
Select the top level folder where the Spring Tool Suite projects are located.  
For this system it is on the desktop in “Project02” folder.



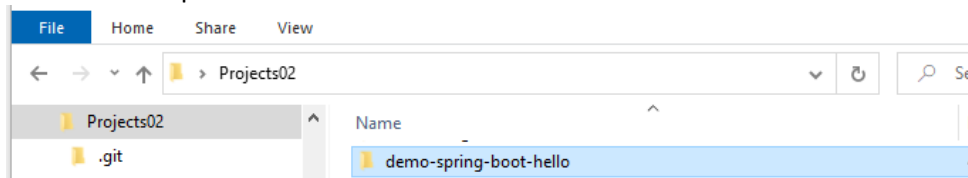
Verify / change extract folder to your IDE workspace top level folder. By selecting the top level folder “Project02” in this case, the demo project folder will be created only once. If extract was used in the download folder, then two layer deep demo project folders would have been created.

Extract the files to the top level project folder.

Click “Extract”



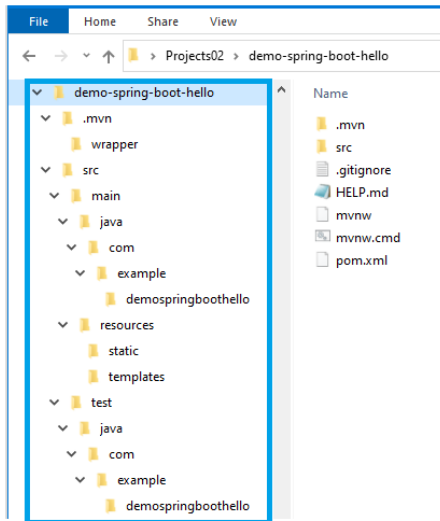
If the “Show extracted files ...” was left checked, a file explorer window will open showing the folder created. The zip file can be deleted.





## Spring Boot Zip File Folder Structure

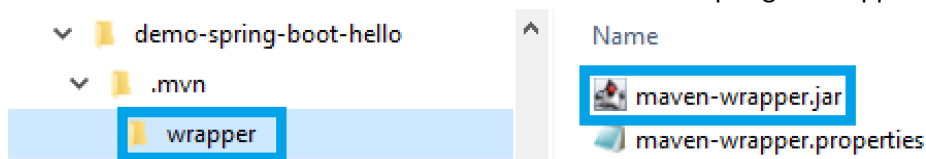
Once the zip file is extracted the folder structure aligns with a Spring Boot IDE project with Maven. The top-level folder is the name we gave our project: “demo-spring-boot-hello”



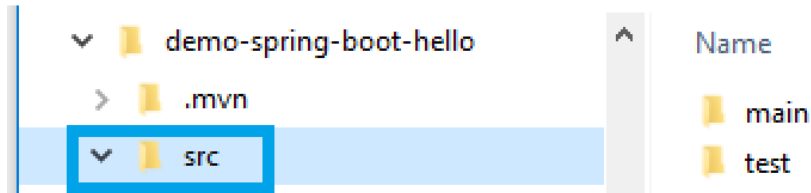
## Spring Boot Sub-Folders

### 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 the 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:

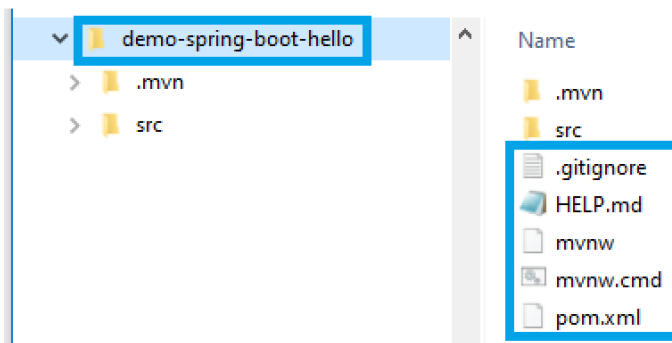
**.gitignore** – This file is related to using Git Hub. This file tells Git which files to ignore when committing your project to the GitHub repository.

**.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.

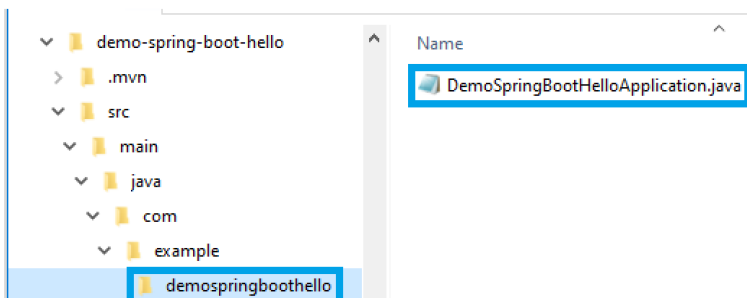


## Spring Boot Package Folders

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

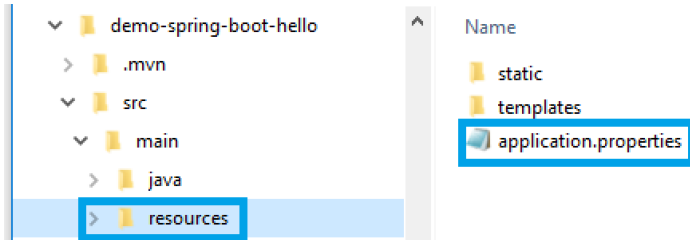
In the src folder structure is the package we defined when creating the zip file: "com.example".

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



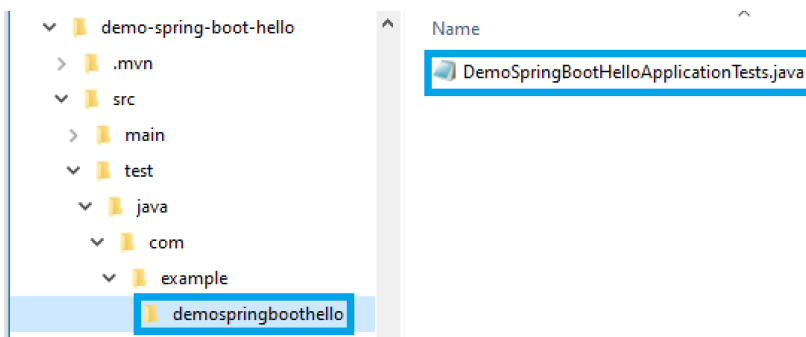
In the resource folder there is a property file shell, it is created as a placeholder, an empty file.

**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 Spring Tool Suite Project](#)” covers the import process for Spring Tool Suites IDE used in various coaching documents created for Spring Boot projects.