Introducing Spring Framework 6

Learning and Building Java-based Applications With Spring

—

Second Edition

—

Felipe Gutierrez
Joseph B. Ottinger

Apress®

Introducing Spring Framework 6

Learning and Building Java-based Applications With Spring

Second Edition

Felipe Gutierrez
Joseph B. Ottinger

Apress®

Introducing Spring Framework 6: Learning and Building Java-based Applications With Spring

Felipe Gutierrez
Cary, NC, USA

Joseph B. Ottinger
YOUNGSVILLE, NC, USA

ISBN-13 (pbk): 978-1-4842-8636-4
ISBN-13 (electronic): 978-1-4842-8637-1
https://doi.org/10.1007/978-1-4842-8637-1

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover designed by eStudioCalamar

Cover image by Mink Mingle on Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress). For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my parents, Rocio Cruz and Felipe Gutierrez.

—Felipe Gutierrez

To my beloved wife and sons, and to the rabbits that keep hounding my yard.

—Joseph B. Ottinger

Table of Contents

About the Authors

Felipe Gutierrez is a solutions software architect, with a bachelor's and a master's degree in computer science from Instituto Tecnologico y de Estudios Superiores de Monterrey Campus Ciudad de Mexico, with over 20 years of IT experience, during which time he developed programs for companies in multiple vertical industries, such as government, retail, healthcare, education, and banking. He is currently working as a principal technical instructor for Pivotal, specializing in Cloud Foundry, Spring Framework, Spring Cloud Native Applications, Groovy, and RabbitMQ, among other technologies. He has worked as a solutions architect for big companies like Nokia, Apple, Redbox, and Qualcomm, among others. He is also the author of *Introducing Spring Framework*, *Pro Spring Boot*, and *Spring Boot Messaging*, all published by Apress.

Joseph B. Ottinger is reputedly an expert software developer, coder, and programmer with experience covering many technologies and platforms. He was the Editor in Chief for both *Java Developer Journal* and TheServerSide.com and has contributed to a large number of publications, open source projects, and commercial products over the years, using many different languages (but primarily Java, Python, and JavaScript). He is the author of *Hibernate Recipes* and *Beginning Hibernate* for Apress and has authored other books as well as a few articles here and there. He's also constantly writing odes to random furry creatures grooving in caves with members of ancient cultures, or something.

About the Technical Reviewer

Manuel Jordan Elera is an autodidactic developer and researcher who enjoys learning new technologies for his own experiments and creating new integrations. Manuel won the Springy Award 2013 Community Champion and Spring Champion. In his little free time, he reads the Bible and composes music on his guitar. Manuel is known as dr_pompeii. He has tech-reviewed numerous books, including *Pro Spring MVC with WebFlux* (Apress, 2020), *Pro Spring Boot 2* (Apress, 2019), *Rapid Java Persistence and Microservices* (Apress, 2019), *Java Language Features* (Apress, 2018), *Spring Boot 2 Recipes* (Apress, 2018), and *Java APIs, Extensions and Libraries* (Apress, 2018). You can read his detailed tutorials on Spring technologies and contact him through his blog at `www.manueljordanelera.blogspot.com`. You can follow Manuel on his Twitter account, @dr_pompeii.

Acknowledgments

I would like to express all my gratitude to the Apress team: first and foremost to Steve Anglin for accepting my proposal; Laura Berendson and Jill Balzaono for helping me out when I needed it; and the rest of the Apress team involved in this project. Thanks to everybody for making this possible.

Thanks to our technical reviewer, Manuel Jordan, and the entire Spring team for making the Spring Framework the best programming and configuration model for modern Java-based enterprise applications.

Thanks to my parents, Rocio Cruz and Felipe Gutierrez, for all their love and support, and to my best friend, my brother Edgar Gerardo Gutierrez. Even though we live far away, we are closer than ever; thanks, "macnitous."

—Felipe Gutierrez

I'm constantly amazed that I'm asked to write these acknowledgments and dedications, mostly because they feel so limiting to point out only specific persons and events to whom I'm grateful. I would like to thank both grammar and spelling, and my sense of restraint in using both proper grammar *and* spelling in doing so, as well as the patience of my wife and family (and friends), without whom I'd hardly ever be able to do anything. This book was written to the tune of many classic '70s tunes, many of which were cheesy soft rock for some reason, and I'd like to thank a whole host of one-hit-wonders without which I'd never wonder what the record industry was thinking back then. It's awesome, and I miss it. May all of us live with as much love in our hearts for each other as we can stand.

—Joseph B. Ottinger

Introduction

This book is an introduction to the well-known Spring Framework that offers an inversion of control container for the Java platform. The Spring Framework is an open source application framework that can be used with any Java application.

After reading this book, you will know how to do the following:

- Use the Spring Framework efficiently.
- Add persistence through JDBC databases, with an easy migration to NoSQL.
- Do unit and integration testing.
- Create web applications and expose RESTful APIs.
- Send messages via JMS, a model that extends to AMQP, RabbitMQ, and MQTT.
- Use dynamic languages like Groovy, Ruby, and Bean Shell for Spring.
- Use Groovy with Spring.
- Use the new Spring Boot and Spring XD technologies.

Who This Book Is For

Introducing Spring Framework 6 is a hands-on guide for any developer who is new to the Spring Framework and wants to learn how to build applications with it. Within this book you will find all the necessary elements to create enterprise-ready applications by using the Spring Framework and all its features and modules.

How This Book Is Organized

This book uses a simple **My Documents** application that you will develop incrementally over the course of the book. The book consists of the following four parts:

- Part 1: Spring Framework Basics: You will learn about the dependency injection design pattern, and Spring's container implementation and how it will help you create a better design by programming toward interfaces. You'll learn the different configurations that you can apply to the Spring Framework. You will also learn how to use bean scopes, work with collections and resource files, and how to test your Spring applications.
- Part 2: The Spring Framework: You will learn to add persistence and integrate your Spring application with other systems. And you will be able to add your Spring application to the Web.
- Part 3: Advanced Techniques with Spring Framework: You will learn how to use message brokers with Spring for massive scalability and distributed architecture, how to expose your application using a RESTful API, and how to send emails and schedule events in your application.
- Part 4: The New Spring I/O: You will learn how to integrate Spring and Groovy into your Spring application. You'll also get pointers on where to look for further technologies in your journey with Spring.

Source Code

All source code used in this book can be found at github.com/apress/introducing-spring-framework6. Included in this download will be the following online-only appendixes:

- Installing Java and Gradle for your Operating System
- Other Recommended Tools

With all that said, let's go ahead and start with the Spring Framework!

PART I

Spring Framework Basics

CHAPTER 1

Your First Spring Application

Most books start with a very long explanation about the technology they are using, the history of it, and often a small example that you can't run until you reach later chapters. In this book, I am going to take a different approach. I am going to start with some basic examples, and I will explain in detail what they do and how to use them so that you get to know Spring quickly. The examples in this chapter will show you how easy it is to integrate the Spring Framework into any existing project or how to start one from scratch and modify it without any effort.

Figure 1-1 shows the Spring portfolio website (`https://spring.io`). In this website, you can find all of the Spring Extensions, guides, and documentation to help you understand better the Spring ecosystem. The Spring Framework itself can be found at `https://spring.io/projects/spring-framework`, if you're interested in drilling down already.

F. Gutierrez and J. B. Ottinger, *Introducing Spring Framework 6*, https://doi.org/10.1007/978-1-4842-8637-1_1

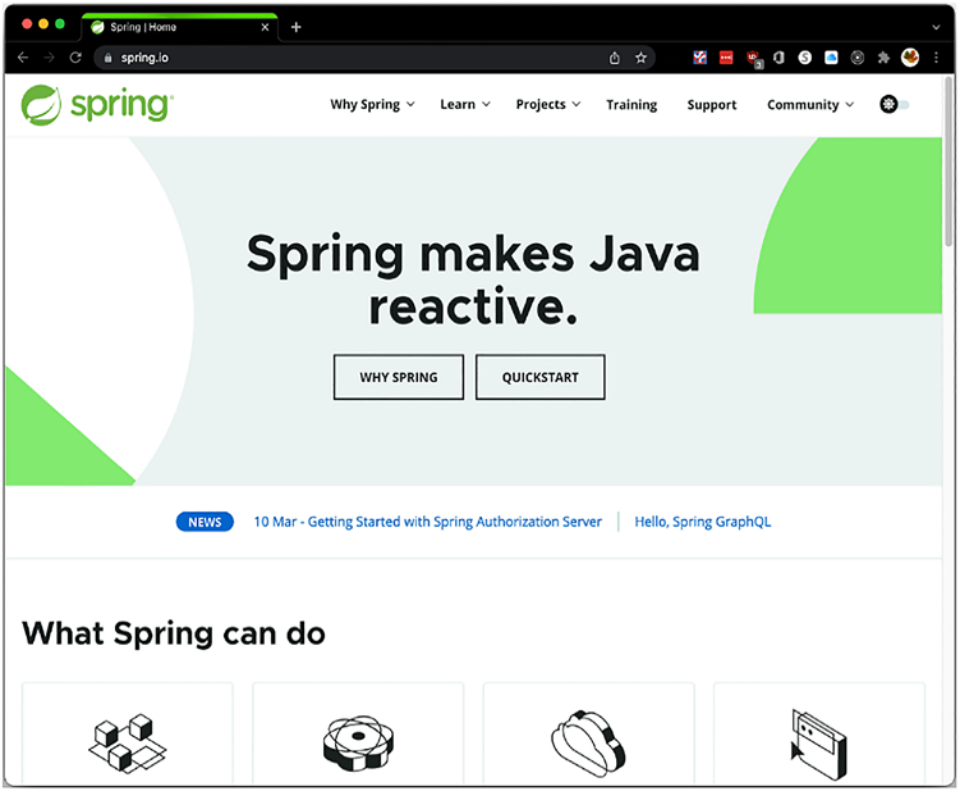

Figure 1-1. *https://spring.io*

Pre-requirements

In order to start with the first Spring Framework example, we need to have some tools installed.

- You need the Java Development Kit (JDK) installed and configured, accessible on the command line. Unlike previous versions of Spring, which supported a wide range of Java versions, Spring 6 requires Java 17 or higher. There are many sites that provide Java 17, but the most appropriate will be either Adoptium's distribution page, at `https://adoptium.net/releases.html`, or Oracle's site, at `www.oracle.com/java/technologies/downloads/`, either of which are appropriate. You can also install Java via SDKMan! if you're on OSX (see `https://sdkman.io`).

- We're going to use the Gradle Build Tool, found at https://gradle.org, for the book (See Figure 1-2). Gradle is one of the two most popular and capable build tools for the JVM; choosing Gradle here is done mostly because it has very simple and short build scripts. We use a build tool because it's very much standard practice and gives us easy dependency management and a full build lifecycle, including the ability to run tests as part of our build; Gradle is also the tool of choice of the Spring team itself.

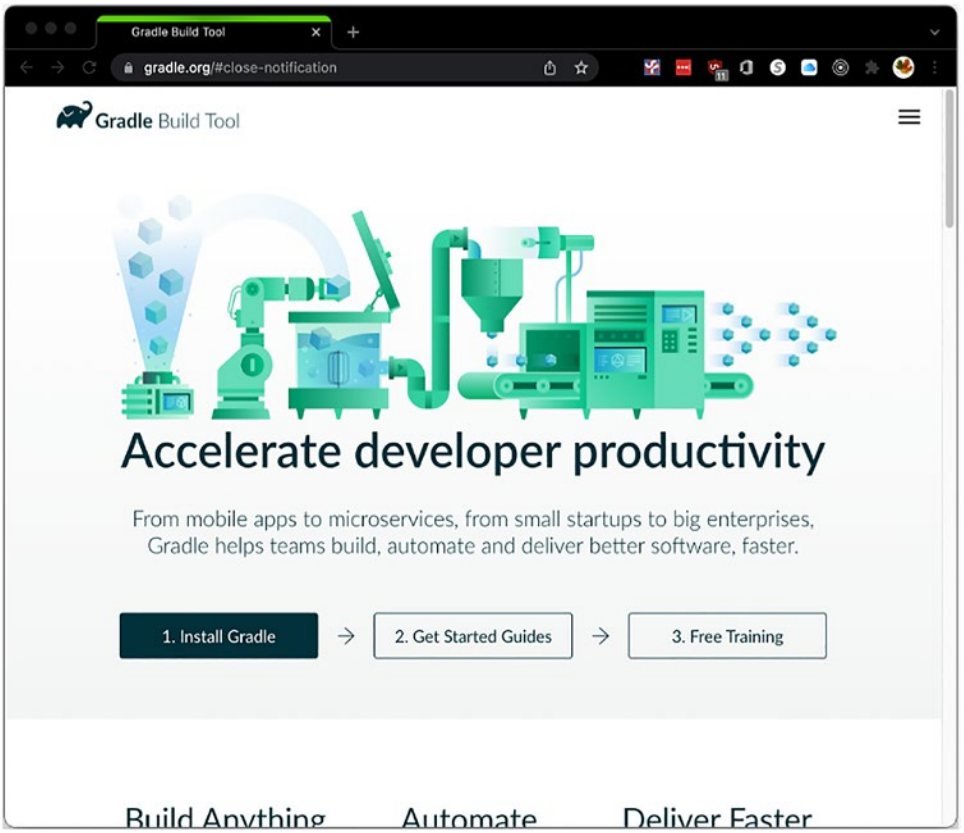

Figure 1-2. *The Gradle Website*

Source Code Organization

We're going to be looking at a lot of source code in this book.

This part will describe how it's laid out so it's easy to understand what is going where.

The book is going to be organized as a Gradle project, with various subdirectories.[1] Most of the subdirectories will be "modules" of the project, although there will be exceptions.

The top-level project needs its own directory; for the authors, it's the isf6 directory (for "Introducing Spring 6"). Most of the chapters' code will be in a clearly named directory mapping to the chapter number: thus, chapter02, chapter03, and so forth.

If a chapter has two separate projects, they'll be named with the chapter and then a description of the project, like `chapter01-hello-world` and `chapter01-hello-boot`.

Let's take a look at the "top-level" project and set it up, and then we'll get to writing some actual code that uses Spring.

The first thing we need to do, after installing Java and Gradle, is run a command to initialize Gradle, `gradle init`.

That will give us a short interaction where Gradle asks us what build options we want.

The defaults are fine, because we'll be overwriting them; the main result we want from this process is the setup of the Gradle wrapper. We want to replace two files in this directory, `build.gradle` and `settings.gradle`.

The top-level `build.gradle` file – in your "project working directory," so all relative paths start with "." – should look like this:

Listing 1-1. `build.gradle`

```
apply plugin: 'java'

sourceCompatibility = 17
targetCompatibility = 17

ext {
    springFrameworkVersion = "6.0.0-M4"
    testNgVersion = "7.6.1"
}
```

[1] Gradle uses a wrapper that downloads a copy of Gradle for each project. This is very useful for repeatable builds, but it creates a copy of Gradle on your hard drive for each project. As a result, it's far more efficient for us to have a single project for the book, rather than a project for each chapter, although some chapters will have separate projects, such as the `chapter-01-hello-boot` project.

```
allprojects {
    apply plugin: 'java'

    repositories {
        maven {
            url "https://repo.spring.io/milestone"
        }
        maven {
            url "https://repo.spring.io/snapshot"
            mavenContent {
                snapshotsOnly()
            }
        }
        mavenCentral()
    }

    dependencies {
        implementation \
        "ch.qos.logback:logback-classic:1.2.11"
        testImplementation \
        "org.testng:testng:$testNgVersion"
    }

    test {
        useTestNG()
    }
}
```

This file looks like a lot – but it's also one of the longer build files we'll have.

What it's doing is telling Gradle that we have a Java project (therefore, use the Java compiler), what level of Java to require (17, the current long-term support release as of writing and a requirement for Spring 6), a set of properties to provide version information to the build, and then a set of things that all projects should do: they're all Java projects, they all should pull dependencies from a set of external locations, and they should all rely on the TestNG testing framework.

The next file we'll want to overwrite is the `settings.gradle` file.

What this one does is describe the subprojects to include in the "main build." This one will change as we develop the content in the book.

Our `settings.gradle` needs to do two things:

1. Configure our project so it can find the plug-ins we'll want.
2. Configure the submodules for our book's content.

As this book is being written, Spring 6 has *not* been released. As a result, we need to use repositories that aren't "official releases." We already see the references to milestone and snapshot repositories in our `build.gradle`, but Gradle resolves dependencies and plug-ins through different mechanisms; thus, we need to configure them separately.

After the preamble for plug-in resolution – the bit that starts with `pluginManagement` – we have a simple way to name the project as a whole (`isf6`), and then we include our first subproject, the `chapter01-hello-spring` module. We'll be adding to this as we proceed through the book.[2]

We're going to start off with a simple "Hello, world" project, with a project name of `chapter01-hello-world`, so our `settings.gradle` file will be simple to start off with.

Listing 1-2. `settings.gradle`

```
pluginManagement {
    repositories {
        maven {
            url "https://repo.spring.io/milestone"
        }
        maven {
            url "https://repo.spring.io/snapshot"
            mavenContent {
                snapshotsOnly()
            }
        }
        gradlePluginPortal()
    }
}
```

[2] If you're looking at the source download, you'll see that the `settings.gradle` has a lot more to it and starts with a `//tag::chapter01[]` comment. These comments are to help format the book, which used the *actual source code* for listings instead of copied source.

```
rootProject.name = 'isf6'
include 'chapter01-hello-world'
```

There are two ways chapters will be represented in the source tree.

If the chapter's code is able to be represented cleanly with a single build, the chapter will be named in its own subdirectory, like `chapter02`, `chapter03`, and so forth.

However, if the chapter actually has two build processes, each build will get its own directory, with the chapter name being part of the name.

Oddly enough, this chapter will have two builds, so it will end up with `chapter01-hello-world` and `chapter01-hello-boot`, as an example.

The source organization for each chapter will follow the Gradle source code standard.

All source files go in a subdirectory called `src`, with (usually) two directories under that: `main` and `test`.

In each of those, there will be a directory corresponding to a source language (like `java`) or named `resources`, to map to source files that don't need compilation.

Under those directories, you'll see source files organized by Java packaging conventions.

Thus, after we create the structure for `chapter01-hello-world`, we have this directory tree as shown in Figure 1-3, assuming we have some Java source files in place already:

Figure 1-3. *The Directory Tree So Far*