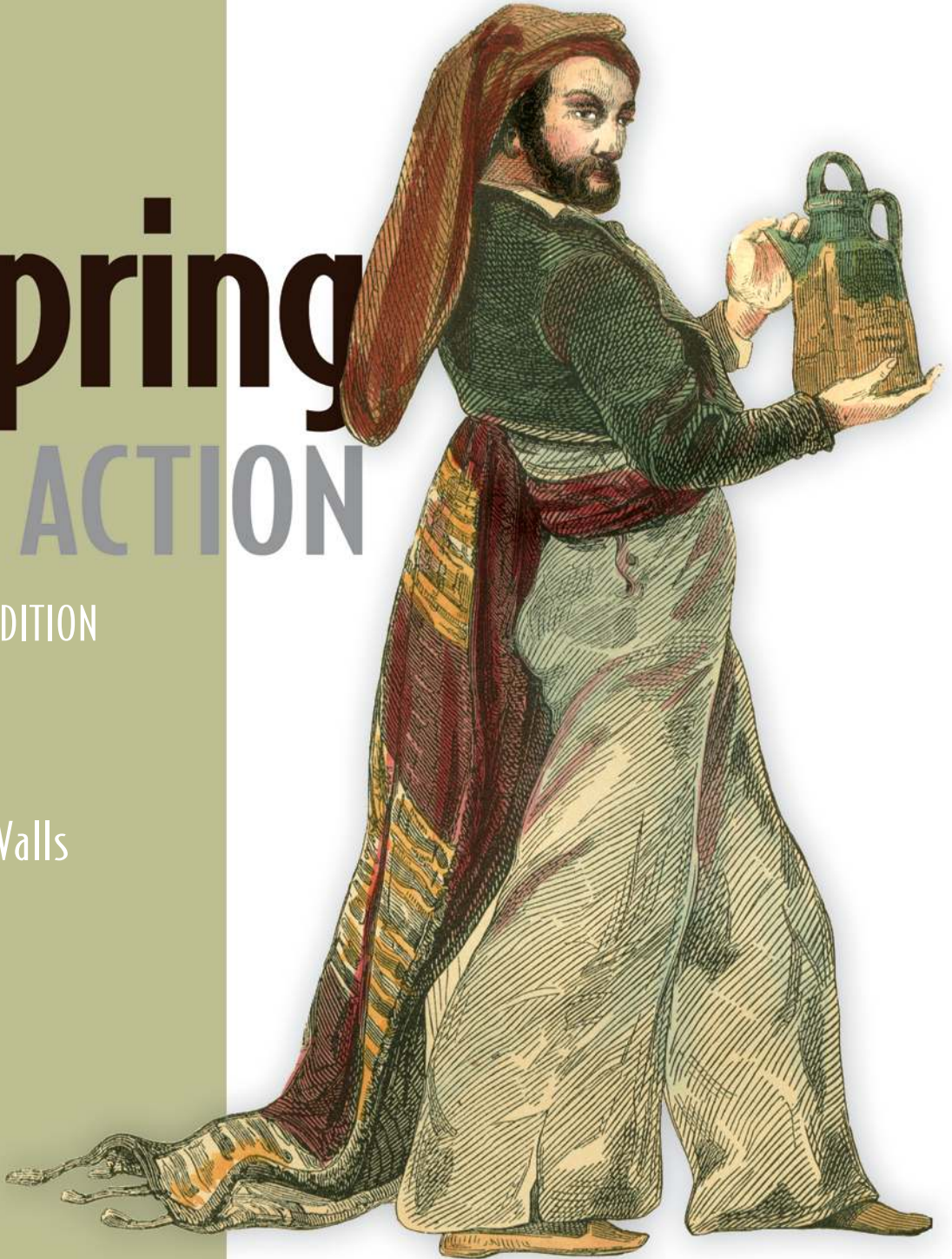


Covers Spring 5.0

Spring IN ACTION

FIFTH EDITION

Craig Walls



 MANNING

Praise for Spring in Action, 4th edition

“The best book for Spring—updated and revised.”

—Gregor Zurowski, Sotheby’s

“The classic, remastered and full of awesomeness.”

—Mario Arias, Cake Solutions Ltd.

“Informative, accurate, and insightful!

—Jeelani Shaik, D3Banking.com

“After ten years, this is still the clearest and most comprehensive introduction to the core concepts of the Spring platform.”

—James Wright, Sword-Apak

“This book is a quick and easy way to get into the Spring Framework Universe. Simply perfect for Java developers.”

—Jens O’Richter, freelance Senior Software Architect

“This book belongs on the bookshelf of any serious Java developer who uses Spring.”

—Jonathan Thoms, Expedia Inc.

“Spring in Action is an excellent travel companion for the huge landscape that is the Spring Framework.”

—Ricardo Lima, Senado Federal do Brasil

“Pragmatic advice for Java’s most important framework.”

—Mike Roberts, Information Innovators

Spring in Action
Fifth Edition

COVERS SPRING 5.0

CRAIG WALLS



MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact


Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

©2019 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

© Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

 Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editor: Jennifer Stout
Project manager: Janet Vail
Copy editors: Frances Buran, Andy Carroll
Proofreaders: Melody Dolab, Katie Tennant
Technical proofreader: Joshua White
Typesetter: Dennis Dalinnik
Cover designer: Marija Tudor

ISBN: 9781617294945

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 – DP – 23 22 21 20 19 18

brief contents

PART 1	FOUNDATIONAL SPRING	1
	1 ■ Getting started with Spring	3
	2 ■ Developing web applications	29
	3 ■ Working with data	56
	4 ■ Securing Spring	84
	5 ■ Working with configuration properties	114
PART 2	INTEGRATED SPRING	135
	6 ■ Creating REST services	137
	7 ■ Consuming REST services	169
	8 ■ Sending messages asynchronously	178
	9 ■ Integrating Spring	209
PART 3	REACTIVE SPRING	239
	10 ■ Introducing Reactor	241
	11 ■ Developing reactive APIs	269
	12 ■ Persisting data reactively	296

PART 4	CLOUD-NATIVE SPRING.....	321
13	■ Discovering services	323
14	■ Managing configuration	343
15	■ Handling failure and latency	376
PART 5	DEPLOYED SPRING	393
16	■ Working with Spring Boot Actuator	395
17	■ Administering Spring	429
18	■ Monitoring Spring with JMX	446
19	■ Deploying Spring	454

contents

preface xiii
acknowledgments xv
about this book xvii

PART 1 FOUNDATIONAL SPRING1

1 *Getting started with Spring* 3

- 1.1 What is Spring? 4
- 1.2 Initializing a Spring application 6
 - Initializing a Spring project with Spring Tool Suite* 7
 - Examining the Spring project structure* 11
- 1.3 Writing a Spring application 17
 - Handling web requests* 17 ■ *Defining the view* 19
 - Testing the controller* 20 ■ *Building and running the application* 21 ■ *Getting to know Spring Boot DevTools* 23
 - Let's review* 25
- 1.4 Surveying the Spring landscape 26
 - The core Spring Framework* 26 ■ *Spring Boot* 26
 - Spring Data* 27 ■ *Spring Security* 27 ■ *Spring Integration and Spring Batch* 27 ■ *Spring Cloud* 28

- ## 2 *Developing web applications* 29
- 2.1 Displaying information 30
 - Establishing the domain* 31
 - *Creating a controller class* 32
 - Designing the view* 35
 - 2.2 Processing form submission 40
 - 2.3 Validating form input 45
 - Declaring validation rules* 46
 - *Performing validation at form binding* 48
 - *Displaying validation errors* 49
 - 2.4 Working with view controllers 51
 - 2.5 Choosing a view template library 52
 - Caching templates* 54
- ## 3 *Working with data* 56
- 3.1 Reading and writing data with JDBC 57
 - Adapting the domain for persistence* 59
 - *Working with JdbcTemplate* 60
 - *Defining a schema and preloading data* 64
 - Inserting data* 66
 - 3.2 Persisting data with Spring Data JPA 75
 - Adding Spring Data JPA to the project* 76
 - *Annotating the domain as entities* 76
 - *Declaring JPA repositories* 80
 - Customizing JPA repositories* 81
- ## 4 *Securing Spring* 84
- 4.1 Enabling Spring Security 85
 - 4.2 Configuring Spring Security 86
 - In-memory user store* 88
 - *JDBC-based user store* 89
 - LDAP-backed user store* 92
 - *Customizing user authentication* 96
 - 4.3 Securing web requests 103
 - Securing requests* 104
 - *Creating a custom login page* 106
 - Logging out* 109
 - *Preventing cross-site request forgery* 109
 - 4.4 Knowing your user 110
- ## 5 *Working with configuration properties* 114
- 5.1 Fine-tuning autoconfiguration 115
 - Understanding Spring's environment abstraction* 116
 - Configuring a data source* 117
 - *Configuring the embedded server* 119
 - *Configuring logging* 120
 - *Using special property values* 121

- 5.2 Creating your own configuration properties 122
 - Defining configuration properties holders* 124
 - *Declaring configuration property metadata* 126
- 5.3 Configuring with profiles 129
 - Defining profile-specific properties* 130
 - *Activating profiles* 131
 - Conditionally creating beans with profiles* 132

PART 2 INTEGRATED SPRING 135

6 *Creating REST services* 137

- 6.1 Writing RESTful controllers 138
 - Retrieving data from the server* 140
 - *Sending data to the server* 145
 - *Updating data on the server* 146
 - *Deleting data from the server* 148
- 6.2 Enabling hypermedia 149
 - Adding hyperlinks* 152
 - *Creating resource assemblers* 154
 - Naming embedded relationships* 159
- 6.3 Enabling data-backed services 160
 - Adjusting resource paths and relation names* 162
 - *Paging and sorting* 164
 - *Adding custom endpoints* 165
 - *Adding custom hyperlinks to Spring Data endpoints* 167

7 *Consuming REST services* 169

- 7.1 Consuming REST endpoints with RestTemplate 170
 - GETting resources* 172
 - *PUTting resources* 173
 - DELETEing resources* 174
 - *POSTing resource data* 174
- 7.2 Navigating REST APIs with Traverson 175

8 *Sending messages asynchronously* 178

- 8.1 Sending messages with JMS 179
 - Setting up JMS* 179
 - *Sending messages with JmsTemplate* 181
 - Receiving JMS messages* 188
- 8.2 Working with RabbitMQ and AMQP 192
 - Adding RabbitMQ to Spring* 193
 - *Sending messages with RabbitTemplate* 194
 - *Receiving message from RabbitMQ* 198
- 8.3 Messaging with Kafka 202
 - Setting up Spring for Kafka messaging* 203
 - *Sending messages with KafkaTemplate* 204
 - *Writing Kafka listeners* 206

9 Integrating Spring 209

- 9.1 Declaring a simple integration flow 210
 - Defining integration flows with XML* 211
 - *Configuring integration flows in Java* 213
 - *Using Spring Integration's DSL configuration* 215
- 9.2 Surveying the Spring Integration landscape 216
 - Message channels* 217
 - *Filters* 219
 - *Transformers* 220
 - Routers* 221
 - *Splitters* 223
 - *Service activators* 225
 - Gateways* 227
 - *Channel adapters* 228
 - *Endpoint modules* 230
- 9.3 Creating an email integration flow 231

PART 3 REACTIVE SPRING 239

10 Introducing Reactor 241

- 10.1 Understanding reactive programming 242
 - Defining Reactive Streams* 243
- 10.2 Getting started with Reactor 245
 - Diagramming reactive flows* 246
 - *Adding Reactor dependencies* 247
- 10.3 Applying common reactive operations 248
 - Creating reactive types* 249
 - *Combining reactive types* 253
 - Transforming and filtering reactive streams* 257
 - *Performing logic operations on reactive types* 266

11 Developing reactive APIs 269

- 11.1 Working with Spring WebFlux 269
 - Introducing Spring WebFlux* 271
 - *Writing reactive controllers* 272
- 11.2 Defining functional request handlers 276
- 11.3 Testing reactive controllers 279
 - Testing GET requests* 279
 - *Testing POST requests* 282
 - Testing with a live server* 284
- 11.4 Consuming REST APIs reactively 285
 - GETting resources* 285
 - *Sending resources* 287
 - Deleting resources* 288
 - *Handling errors* 289
 - Exchanging requests* 290

- 11.5 Securing reactive web APIs 292
 - Configuring reactive web security* 292
 - *Configuring a reactive user details service* 294

12 *Persisting data reactively* 296

- 12.1 Understanding Spring Data's reactive story 297
 - Spring Data reactive distilled* 297
 - *Converting between reactive and non-reactive types* 298
 - *Developing reactive repositories* 300
- 12.2 Working with reactive Cassandra repositories 300
 - Enabling Spring Data Cassandra* 301
 - *Understanding Cassandra data modeling* 303
 - *Mapping domain types for Cassandra persistence* 304
 - *Writing reactive Cassandra repositories* 309
- 12.3 Writing reactive MongoDB repositories 312
 - Enabling Spring Data MongoDB* 312
 - *Mapping domain types to documents* 314
 - *Writing reactive MongoDB repository interfaces* 317

PART 4 CLOUD-NATIVE SPRING.....321

13 *Discovering services* 323

- 13.1 Thinking in microservices 324
- 13.2 Setting up a service registry 326
 - Configuring Eureka* 330
 - *Scaling Eureka* 333
- 13.3 Registering and discovering services 334
 - Configuring Eureka client properties* 335
 - *Consuming services* 337

14 *Managing configuration* 343

- 14.1 Sharing configuration 344
- 14.2 Running Config Server 345
 - Enabling Config Server* 346
 - *Populating the configuration repository* 349
- 14.3 Consuming shared configuration 352
- 14.4 Serving application- and profile-specific properties 353
 - Serving application-specific properties* 354
 - *Serving properties from profiles* 355
- 14.5 Keeping configuration properties secret 357
 - Encrypting properties in Git* 357
 - *Storing secrets in Vault* 360

- 14.6 Refreshing configuration properties on the fly 364
 - Manually refreshing configuration properties* 365
 - Automatically refreshing configuration properties* 367

15 Handling failure and latency 376

- 15.1 Understanding circuit breakers 376
- 15.2 Declaring circuit breakers 378
 - Mitigating latency* 381
 - *Managing circuit breaker thresholds* 382
- 15.3 Monitoring failures 383
 - Introducing the Hystrix dashboard* 384
 - *Understanding Hystrix thread pools* 387
- 15.4 Aggregating multiple Hystrix streams 389

PART 5 DEPLOYED SPRING393

16 Working with Spring Boot Actuator 395

- 16.1 Introducing Actuator 396
 - Configuring Actuator's base path* 397
 - *Enabling and disabling Actuator endpoints* 398
- 16.2 Consuming Actuator endpoints 399
 - Fetching essential application information* 400
 - *Viewing configuration details* 403
 - *Viewing application activity* 411
 - Tapping runtime metrics* 413
- 16.3 Customizing Actuator 416
 - Contributing information to the /info endpoint* 416
 - Defining custom health indicators* 421
 - *Registering custom metrics* 422
 - *Creating custom endpoints* 424
- 16.4 Securing Actuator 426

17 Administering Spring 429

- 17.1 Using the Spring Boot Admin 430
 - Creating an Admin server* 430
 - *Registering Admin clients* 431
- 17.2 Exploring the Admin server 435
 - Viewing general application health and information* 436
 - Watching key metrics* 437
 - *Examining environment properties* 438
 - *Viewing and setting logging levels* 439
 - Monitoring threads* 440
 - *Tracing HTTP requests* 441

17.3 Securing the Admin server 442

Enabling login in the Admin server 443 ■ *Authenticating with the Actuator* 444

18 *Monitoring Spring with JMX* 446

18.1 Working with Actuator MBeans 446

18.2 Creating your own MBeans 449

18.3 Sending notifications 451

19 *Deploying Spring* 454

19.1 Weighing deployment options 455

19.2 Building and deploying WAR files 456

19.3 Pushing JAR files to Cloud Foundry 458

19.4 Running Spring Boot in a Docker container 461

19.5 The end is where we begin 465

appendix Bootstrapping Spring applications 466

index 487

preface

After nearly 15 years of working with Spring and having written five editions of this book (not to mention *Spring Boot in Action*), you'd think that it'd be hard to come up with something exciting and new to say about Spring when writing the preface for this book. But nothing could be further from the truth!

Every single release of Spring, Spring Boot, and all of the other projects in the Spring ecosystem unleashes some new amazing capabilities that rekindle the fun in developing applications. With Spring reaching a significant milestone with its 5.0 release and Spring Boot releasing version 2.0, there's so much more Spring to enjoy that it was a no-brainer to write another edition of *Spring in Action*.

The big story of Spring 5 is reactive programming support, including Spring Web-Flux, a brand new reactive web framework that borrows its programming model from Spring MVC, allowing developers to create web applications that scale better and make better use of fewer threads. Moving toward the backend of a Spring application, the latest edition of Spring Data enables the creation of reactive, non-blocking data repositories. And all of this is built on top of Project Reactor, a Java library for working with reactive types.

In addition to the new reactive programming features of Spring 5, Spring Boot 2 now provides even more autoconfiguration support than ever before as well as a completely reimagined Actuator for peeking into and manipulating a running application.

What's more, as developers look to break down their monolithic applications into discrete microservices, Spring Cloud provides facilities that make it easy to configure and discover microservices, as well as fortify them so they're more resilient to failure.

I'm happy to say that this fifth edition of *Spring in Action* covers all of this and more! If you're a seasoned veteran with Spring, *Spring in Action, Fifth Edition* will be your guide to everything new that Spring has to offer. On the other hand, if you're new to Spring, then there's no better time than now to get in on the action and the first few chapters will get you up and running in no time!

It's been an exciting 15 years of working with Spring. And now that I've written this fifth edition of *Spring in Action*, I'm eager to share that excitement with you!

acknowledgments

One of the most amazing things that Spring and Spring Boot do is to automatically provide all of the foundational plumbing for an application, leaving you as a developer to focus primarily on the logic that's unique to your application. Unfortunately, no such magic exists for writing a book. Or does it?

At Manning, there were several people working their magic to make sure that this book is the best it can possibly be. Many thanks in particular to Jenny Stout, my development editor, and to the production team, including project manager Janet Vail, copyeditors Andy Carroll and Frances Buran, and proofreaders Katie Tennant and Melody Dolab. Thanks, too, to technical proofer Joshua White who was thorough and helpful.

Along the way, we got feedback from several peer reviewers who made sure that the book stayed on target and covered the right stuff. For this, my thanks goes to Andrea Barisone, Arnaldo Ayala, Bill Fly, Colin Joyce, Daniel Vaughan, David Witherspoon, Eddu Melendez, Iain Campbell, Jetro Coenradie, John Gunvaldson, Markus Matzker, Nick Rakochy, Nusry Firdousi, Piotr Kafel, Raphael Villela, Riccardo Noviello, Sergio Fernandez Gonzalez, Sergiy Pylypets, Thiago Presa, Thorsten Weber, Waldemar Modzelewski, Yagiz Erkan, and Željko Trogrlić.

As always, there'd be absolutely no point in writing this book if it weren't for the amazing work done by the members of the Spring engineering team. I'm amazed at what you've created and how we continue to change how software is developed.

Many thanks to my fellow speakers on the No Fluff/Just Stuff tour. I continue to learn so much from every one of you. I especially want to thank Brian Sletten, Nate

Schutta, and Ken Kousen for conversations and emails about Spring that have helped shape this book.

Once again, I'd like to thank the Phoenicians. You know what you did.

Finally, to my beautiful wife Raymie, the love of my life, my sweetest dream, and my inspiration: Thank you for your encouragement and for putting up with another book project. And to my sweet and wonderful girls, Maisy and Madi: I am so proud of you and of the amazing young ladies you are becoming. I love all of you more than you can imagine or I can possibly express.

about this book

Spring in Action, Fifth Edition was written to equip you to build amazing applications using the Spring Framework, Spring Boot, and a variety of ancillary members of the Spring ecosystem. It begins by showing you how to develop web-based, database-backed Java applications with Spring and Spring Boot. It then expands on the essentials by showing how to integrate with other applications, program using reactive types, and then break an application into discrete microservices. Finally, it discusses how to ready an application for deployment.

Although all of the projects in the Spring ecosystem provide excellent documentation, this book does something that none of the reference documents do: provide a hands-on, project-driven guide to bringing the elements of Spring together to build a real application.

Who should read this book

Spring in Action, 5th edition is for Java developers who want to get started with Spring Boot and the Spring Framework as well as for seasoned Spring developers who want to go beyond the basics and learn the newest features of Spring.

How this book is organized: a roadmap

The book has 5 parts spanning 19 chapters. Part 1 covers the foundational topics of building Spring applications:

- Chapter 1 introduces Spring and Spring Boot and how to initialize a Spring project. In this chapter, you'll take the first steps toward building a Spring application that you'll expand upon throughout the course of the book.

- Chapter 2 discusses building the web layer of an application using Spring MVC. In this chapter, you'll build controllers that handle web requests and views that render information in the web browser.
- Chapter 3 delves into the backend of a Spring application where data is persisted to a relational database.
- In chapter 4, you'll use Spring Security to authenticate users and prevent unauthorized access to an application.
- Chapter 5 reveals how to configure a Spring application using Spring Boot configuration properties. You'll also learn how to selectively apply configuration using profiles.

Part 2 covers topics that help integrate your Spring application with other applications:

- Chapter 6 expands on the discussion of Spring MVC started in chapter 2 by looking at how to write REST APIs in Spring.
- Chapter 7 turns the tables on chapter 6 to show how a Spring application can consume a REST API.
- Chapter 8 looks at using asynchronous communication to enable a Spring application to both send and receive messages using the Java Message Service, RabbitMQ, or Kafka.
- Chapter 9 discusses declarative application integration using the Spring Integration project.

Part 3 explores the exciting new support for reactive programming in Spring:

- Chapter 10 introduces Project Reactor, the reactive programming library that underpins Spring 5's reactive features.
- Chapter 11 revisits REST API development, introducing Spring WebFlux, a new web framework that borrows much from Spring MVC while offering a new reactive model for web development.
- Chapter 12 takes a look at writing reactive data persistence with Spring Data to read and write data to Cassandra and Mongo databases.

Part 4 breaks down the monolithic application model, introducing you to Spring Cloud and microservice development:

- Chapter 13 dives into service discovery, using Spring with Netflix's Eureka registry to both register and discover Spring-based microservices.
- Chapter 14 shows how to centralize application configuration in a configuration server that shares configuration across multiple microservices.
- Chapter 15 introduces the circuit breaker pattern with Hystrix, enabling microservices that are resilient in the face of failure.

In part 5, you'll ready an application for production and see how to deploy it:

- Chapter 16 introduces the Spring Boot Actuator, an extension to Spring Boot that exposes the internals of a running Spring application as REST endpoints.

- In chapter 17 you'll see how to use the Spring Boot Admin to put a user-friendly browser-based administrative application on top of the Actuator.
- Chapter 18 discusses how to expose and consume Spring beans as JMX MBeans.
- Finally, in chapter 19 you'll see how to deploy your Spring application in a variety of production environments.

In general, developers new to Spring should start with chapter 1 and work through each chapter sequentially. Experienced Spring developers may prefer to jump in at any point that interests them. Even so, each chapter builds upon the previous chapter, so there may be some context missing if you dive into the middle of the book.

About the code

This book contains many examples of source code both in numbered listings and inline with normal text. In both cases, source code is formatted in a fixed-width font like this to separate it from ordinary text. Sometimes code is also in **bold** to highlight code that has changed from previous steps in the chapter, such as when a new feature adds to an existing line of code.

In many cases the original source code has been reformatted; we've added line breaks and reworked indentation to accommodate the available page space in the book. In rare cases, even this was not enough, and listings include line-continuation markers (➡). Additionally, comments in the source code have often been removed from the listings when the code is described in the text. Code annotations accompany many of the listings, highlighting important concepts.

Source code for the examples in this book is available for download from the publisher's website at www.manning.com/books/spring-in-action-fifth-edition as well as from the author's GitHub account at github.com/habuma/spring-in-action-5-samples.

Book forum

Purchase of *Spring in Action*, 5th edition, includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the author and from other users. To access the forum, go to <https://forums.manning.com/forums/spring-in-action-fifth-edition>. You can also learn more about Manning's forums and the rules of conduct at <https://forums.manning.com/forums/about>.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the author can take place. It is not a commitment to any specific amount of participation on the part of the author, whose contribution to the forum remains voluntary (and unpaid). We suggest you try asking the author some challenging questions lest his interest stray! The forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

Other online resources

Need additional help?

- The Spring website has several useful getting-started guides (some of which were written by the author of this book) at <https://spring.io/guides>.
- The Spring tag at StackOverflow (<https://stackoverflow.com/questions/tagged/spring>) as well as the Spring Boot tag at StackOverflow are great places to ask questions and help others with Spring. Helping someone else with their Spring questions is a great way to learn Spring!

About the author

CRAIG WALLS is a principal engineer with Pivotal. He's a zealous promoter of the Spring Framework, speaking frequently at local user groups and conferences and writing about Spring. When he's not slinging code, Craig is planning his next trip to Disney World or Disneyland and spending as much time as he can with his wife, two daughters, two birds, and three dogs.

About the cover illustration

The figure on the cover of *Spring in Action*, 5th edition, is “Le Caraco,” or an inhabitant of the province of Karak in southwest Jordan. Its capital is the city of Al-Karak, which boasts an ancient hilltop castle with magnificent views of the Dead Sea and surrounding plains. The illustration is taken from a French travel book, *Encyclopédie des Voyages* by J. G. St. Sauveur, published in 1796. Travel for pleasure was a relatively new phenomenon at the time and travel guides such as this one were popular, introducing both the tourist as well as the armchair traveler to the inhabitants of other regions of France and abroad.

The diversity of the drawings in the *Encyclopédie des Voyages* speaks vividly of the distinctiveness and individuality of the world's towns and provinces just two hundred years ago. This was a time when the dress codes of two regions separated by a few dozen miles identified people uniquely as belonging to one or the other. The travel guide brings to life a sense of isolation and distance of that period, and of every other historic period except our own hyperkinetic present.

Dress codes have changed since then and the diversity by region, so rich at the time, has faded away. It is now often hard to tell the inhabitants of one continent from another. Perhaps, trying to view it optimistically, we have traded a cultural and visual diversity for a more varied personal life—or a more varied and interesting intellectual and technical life. We at Manning celebrate the inventiveness, the initiative, and the fun of the computer business with book covers based on the rich diversity of regional life two centuries ago brought back to life by the pictures from this travel guide.

Part 1

Foundational Spring

Part 1 of this book will get you started writing a Spring application, learning the foundations of Spring along the way.

In chapter 1, I'll give you a quick overview of Spring and Spring Boot essentials and show you how to initialize a Spring project as you work on building Taco Cloud, your first Spring application. In chapter 2, you'll dig deeper into the Spring MCV and learn how to present model data in the browser and how to process and validate form input. You'll also get some tips on choosing a view template library. You'll add data persistence to the Taco Cloud application in chapter 3. There, we'll cover using Spring's JDBC template, how to insert data, and how to declare JPA repositories with Spring Data. Chapter 4 covers security for your Spring application, including autoconfiguring Spring Security, defining custom user storage, customizing the login page, and securing against cross-site request forgery (CSRF) attacks. To close out part 1, we'll look at configuration properties in chapter 5. You'll learn how to fine-tune autoconfigured beans, apply configuration properties to application components, and work with Spring profiles.

Getting started with Spring

This chapter covers

- Spring and Spring Boot essentials
- Initializing a Spring project
- An overview of the Spring landscape

Although the Greek philosopher Heraclitus wasn't well known as a software developer, he seemed to have a good handle on the subject. He has been quoted as saying, "The only constant is change." That statement captures a foundational truth of software development.

The way we develop applications today is different than it was a year ago, 5 years ago, 10 years ago, and certainly 15 years ago, when an initial form of the Spring Framework was introduced in Rod Johnson's book, *Expert One-on-One J2EE Design and Development* (Wrox, 2002, <http://mng.bz/oVjy>).

Back then, the most common types of applications developed were browser-based web applications, backed by relational databases. While that type of development is still relevant, and Spring is well equipped for those kinds of applications, we're now also interested in developing applications composed of microservices destined for the cloud that persist data in a variety of databases. And a new interest in reactive programming aims to provide greater scalability and improved performance with non-blocking operations.

As software development evolved, the Spring Framework also changed to address modern development concerns, including microservices and reactive programming. Spring also set out to simplify its own development model by introducing Spring Boot.

Whether you're developing a simple database-backed web application or constructing a modern application built around microservices, Spring is the framework that will help you achieve your goals. This chapter is your first step in a journey through modern application development with Spring.

1.1 **What is Spring?**

I know you're probably itching to start writing a Spring application, and I assure you that before this chapter ends, you'll have developed a simple one. But first, let me set the stage with a few basic Spring concepts that will help you understand what makes Spring tick.

Any non-trivial application is composed of many components, each responsible for its own piece of the overall application functionality, coordinating with the other application elements to get the job done. When the application is run, those components somehow need to be created and introduced to each other.

At its core, Spring offers a *container*, often referred to as the *Spring application context*, that creates and manages application components. These components, or *beans*, are wired together inside the Spring application context to make a complete application, much like bricks, mortar, timber, nails, plumbing, and wiring are bound together to make a house.

The act of wiring beans together is based on a pattern known as *dependency injection* (DI). Rather than have components create and maintain the lifecycle of other beans that they depend on, a dependency-injected application relies on a separate entity (the container) to create and maintain all components and inject those into the beans that need them. This is done typically through constructor arguments or property accessor methods.

For example, suppose that among an application's many components, there are two that you'll address: an inventory service (for fetching inventory levels) and a product service (for providing basic product information). The product service depends on the inventory service to be able to provide a complete set of information about products. Figure 1.1 illustrates the relationships between these beans and the Spring application context.

On top of its core container, Spring and a full portfolio of related libraries offer a web framework, a variety of data persistence options, a security framework, integration with other systems, runtime monitoring, microservice support, a reactive programming model, and many other features necessary for modern application development.

Historically, the way you would guide Spring's application context to wire beans together was with one or more XML files that described the components and their relationship to other components. For example, the following XML declares two

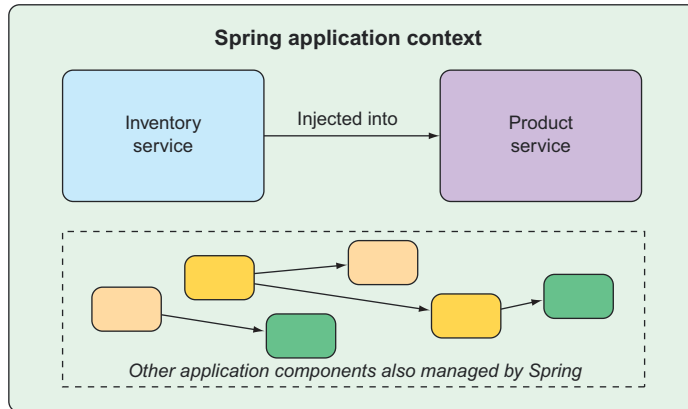


Figure 1.1 Application components are managed and injected into each other by the Spring application context.

beans, an `InventoryService` bean and a `ProductService` bean, and wires the `InventoryService` bean into `ProductService` via a constructor argument:

```
<bean id="inventoryService"
      class="com.example.InventoryService" />

<bean id="productService"
      class="com.example.ProductService" />
  <constructor-arg ref="inventoryService" />
</bean>
```

In recent versions of Spring, however, a Java-based configuration is more common. The following Java-based configuration class is equivalent to the XML configuration:

```
@Configuration
public class ServiceConfiguration {
    @Bean
    public InventoryService inventoryService() {
        return new InventoryService();
    }

    @Bean
    public ProductService productService() {
        return new ProductService(inventoryService());
    }
}
```

The `@Configuration` annotation indicates to Spring that this is a configuration class that will provide beans to the Spring application context. The configuration's class methods are annotated with `@Bean`, indicating that the objects they return should be added as beans in the application context (where, by default, their respective bean IDs will be the same as the names of the methods that define them).

Java-based configuration offers several benefits over XML-based configuration, including greater type safety and improved refactorability. Even so, explicit configuration with either Java or XML is only necessary if Spring is unable to automatically configure the components.

Automatic configuration has its roots in the Spring techniques known as *autowiring* and *component scanning*. With component scanning, Spring can automatically discover components from an application's classpath and create them as beans in the Spring application context. With autowiring, Spring automatically injects the components with the other beans that they depend on.

More recently, with the introduction of Spring Boot, automatic configuration has gone well beyond component scanning and autowiring. Spring Boot is an extension of the Spring Framework that offers several productivity enhancements. The most well-known of these enhancements is *autoconfiguration*, where Spring Boot can make reasonable guesses of what components need to be configured and wired together, based on entries in the classpath, environment variables, and other factors.

I'd like to show you some example code that demonstrates autoconfiguration. But I can't. You see, autoconfiguration is much like the wind. You can see the effects of it, but there's no code that I can show you and say "Look! Here's an example of autoconfiguration!" Stuff happens, components are enabled, and functionality is provided without writing code. It's this lack of code that's essential to autoconfiguration and what makes it so wonderful.

Spring Boot autoconfiguration has dramatically reduced the amount of explicit configuration (whether with XML or Java) required to build an application. In fact, by the time you finish the example in this chapter, you'll have a working Spring application that has only a single line of Spring configuration code!

Spring Boot enhances Spring development so much that it's hard to imagine developing Spring applications without it. For that reason, this book treats Spring and Spring Boot as if they were one and the same. We'll use Spring Boot as much as possible, and explicit configuration only when necessary. And, because Spring XML configuration is the old-school way of working with Spring, we'll focus primarily on Spring's Java-based configuration.

But enough of this chitchat, yakety-yak, and flimflam. This book's title includes the phrase *in action*, so let's get moving, and you can start writing your first application with Spring.

1.2 *Initializing a Spring application*

Through the course of this book, you'll create Taco Cloud, an online application for ordering the most wonderful food created by man—tacos. Of course, you'll use Spring, Spring Boot, and a variety of related libraries and frameworks to achieve this goal.

You'll find several options for initializing a Spring application. Although I could walk you through the steps of manually creating a project directory structure and

defining a build specification, that's wasted time—time better spent writing application code. Therefore, you're going to lean on the Spring Initializr to bootstrap your application.

The Spring Initializr is both a browser-based web application and a REST API, which can produce a skeleton Spring project structure that you can flesh out with whatever functionality you want. Several ways to use Spring Initializr follow:

- From the web application at <http://start.spring.io>
- From the command line using the `curl` command
- From the command line using the Spring Boot command-line interface
- When creating a new project with Spring Tool Suite
- When creating a new project with IntelliJ IDEA
- When creating a new project with NetBeans

Rather than spend several pages of this chapter talking about each one of these options, I've collected those details in the appendix. In this chapter, and throughout this book, I'll show you how to create a new project using my favorite option: Spring Initializr support in the Spring Tool Suite.

As its name suggests, Spring Tool Suite is a fantastic Spring development environment. But it also offers a handy Spring Boot Dashboard feature that (at least at the time I write this) isn't available in any of the other IDE options.

If you're not a Spring Tool Suite user, that's fine; we can still be friends. Hop over to the appendix and substitute the Initializr option that suits you best for the instructions in the following sections. But know that throughout this book, I may occasionally reference features specific to Spring Tool Suite, such as the Spring Boot Dashboard. If you're not using Spring Tool Suite, you'll need to adapt those instructions to fit your IDE.

1.2.1 Initializing a Spring project with Spring Tool Suite

To get started with a new Spring project in Spring Tool Suite, go to the File menu and select New, and then Spring Starter Project. Figure 1.2 shows the menu structure to look for.



Figure 1.2 Starting a new project with the Initializr in Spring Tool Suite

Once you select Spring Starter Project, a new project wizard dialog (figure 1.3) appears. The first page in the wizard asks you for some general project information, such as the project name, description, and other essential information. If you're familiar with the

contents of a Maven pom.xml file, you'll recognize most of the fields as items that end up in a Maven build specification. For the Taco Cloud application, fill in the dialog as shown in figure 1.3, and then click Next.

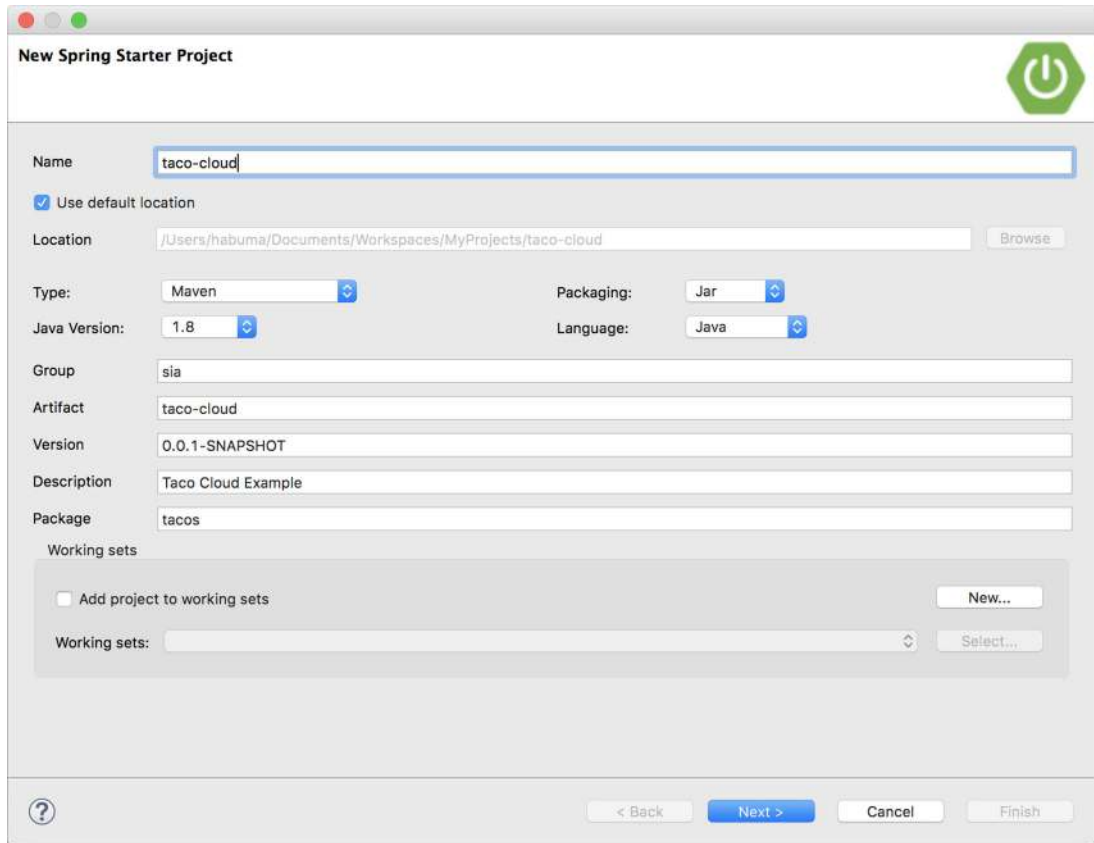


Figure 1.3 Specifying general project information for the Taco Cloud application

The next page in the wizard lets you select dependencies to add to your project (see figure 1.4). Notice that near the top of the dialog, you can select which version of Spring Boot you want to base your project on. This defaults to the most current version available. It's generally a good idea to leave it as is unless you need to target a different version.

As for the dependencies themselves, you can either expand the various sections and seek out the desired dependencies manually, or search for them in the search box at the top of the Available list. For the Taco Cloud application, you'll start with the dependencies shown in figure 1.4.

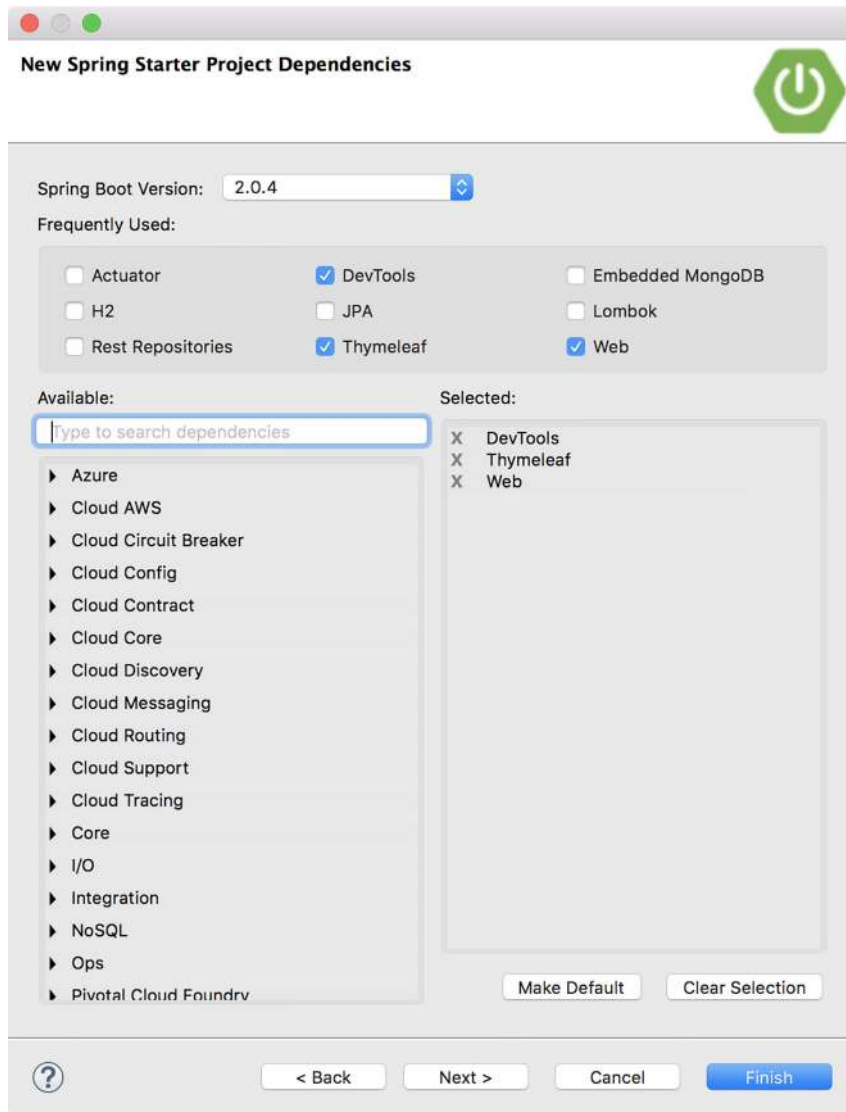


Figure 1.4 Choosing starter dependencies

At this point, you can click Finish to generate the project and add it to your workspace. But if you're feeling slightly adventurous, click Next one more time to see the final page of the new starter project wizard, as shown in figure 1.5.

By default, the new project wizard makes a call to the Spring Initializr at <http://start.spring.io> to generate the project. Generally, there's no need to override this default, which is why you could have clicked Finish on the second page of the