Learning Spring Application Development

Develop dynamic, feature-rich, and robust Spring-based applications using the Spring Framework

Ravi Kant Soni
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Introducing the Spring Framework'
- A synopsis of the book’s content
- More information on Learning Spring Application Development

About the Author

**Ravi Kant Soni** is a Java Enterprise and Spring Framework specialist with a bachelor's degree in information science and engineering from the Reva Institute of Technology, Bangalore. He has been involved in software development for many years now. Ravi has worn many hats throughout his tenure, ranging from software development, multitenant application design, and the integration of new technology into an existing system, to his current love of writing a Spring Framework book.

Currently, he is a lead engineer at HCL Technologies Limited. Ravi has focused on web and enterprise development using the Spring Framework for most of his career and has been extensively involved in application design and implementation. He has developed applications for core bank, HR and payroll, and e-commerce systems using the Spring Framework.

Ravi has gained extensive experience in all aspects of software engineering, including software design, systems architecture, application programming, and automation testing. He is backed by strong product development experience in Java, Spring, Hibernate, PostgreSQL, and many other enterprise technologies.

Ravi is skilled in other techniques such as Bootstrap, jQuery, FreeMarker, Maven, CAS (SSO) Security, Git, Selenium WebDriver, and Agile methodology. Ravi loves problem statements and really enjoys brainstorming unique solutions. He can be reached at springframeworkbyravi@gmail.com. You can also get in touch with him at in.linkedin.com/in/november03ravikantsoni/.
Learning Spring Application Development

The Spring Framework is a cutting-edge framework that provides comprehensive infrastructure support for developing Java applications. The Spring Framework handles the infrastructure so that you can focus on your application. It promotes good programming practice by enabling a POJO-based programming model and also provides a good way to structure your application into layers. It is the appropriate time for you to understand how to best leverage the Spring Framework to create high-performing, easily testable, reusable code when architecting, designing, and developing large-scale Java development projects.

Some of you prefer learning by reading, while others prefer learning by coding. I believe that learning by coding results in better learning, which is what I've done in this book. There is plenty of example code and adequate textual description to help you grasp each Spring Framework feature presented. From the very first chapter, you will be able to develop an application using the Spring Framework.

The Spring Framework is an ocean with a number of features. This book covers a lot of commonly used features in applications and has taken care to present code-based examples for every feature. The book is replenished with tons of code and diagrams. Extra effort has been taken to present snapshots of the libraries used in every example and output. For more information about this book, visit http://learningspringapplicationdevelopment.com/

What This Book Covers

Chapter 1, Introducing the Spring Framework, helps you to understand the architecture of Spring Framework and set up the key components of the Spring application development environment. This chapter serves as a roadmap to the rest of the book.

Chapter 2, Inversion of Control in Spring, configures the Spring Container and manages Spring beans using XML. In this chapter, we take a look at the concepts of Inversion of Control (IoC) and Dependency Injection.

Chapter 3, DAO and JDBC in Spring, grants you access to data using the DAO design pattern and Spring. Implement JDBC support and ORM support in the Spring Framework. This chapter discusses how Spring manages data sources and which data sources you can use in your applications.
Chapter 4, *Hibernate with Spring*, covers one of the object-relational mapping (ORM) libraries that has wide support in Spring—Hibernate. It covers mapping configurations to map persistent classes and discusses how to configure Hibernate to work in a Spring application.

Chapter 5, *Spring Web MVC Framework*, lets you leverage the best of Spring web controllers and the Spring form tag library to create a Spring MVC application. It introduces Spring MVC and discusses how we can use the powerful features provided by Spring MVC to develop high-performing web applications.

Chapter 6, *Spring Security*, allows you to secure your applications against malicious intruders using Spring Security. It introduces Acegi Security System and discusses how to secure web applications using Servlet filters.

Chapter 7, *Spring Testing*, implements practical testing strategies using JUnit and TestNG. It explains how unit tests work, focusing in particular on the JUnit framework.

Chapter 8, *Integrating JavaMail and JMS with Spring*, implements the Spring Mail Application programming interface to send and receive e-mails. It introduces Java Messaging Service (JMS) for asynchronous processing.

Chapter 9, *Inversion of Control in Spring – Using Annotation*, configures Spring beans and Dependency Injection using annotation. It covers annotation-based Dependency Injection and life cycle annotation. It explains how to reference beans using Spring Expression Language (SpEL), invoke methods using SpEL, and work with operators in SpEL. It also covers the text messages and internationalization provided by Spring, which we will learn to implement in our application. This is an online chapter available at https://www.packtpub.com/sites/default/files/downloads/7368OS_Chapter9.pdf.

Chapter 10, *Aspect-oriented Programming with Spring*, introduces you to aspect-oriented programming. It shows you how and where to apply your aspects in your application using Spring's powerful pointcut mechanism and discusses proxies in the Spring AOP. This is an online chapter available at https://www.packtpub.com/sites/default/files/downloads/7368OS_Chapter10.pdf.
Appendix A, Solutions to Exercises, provides solutions to all the exercises from every chapter of this book.

Appendix B, Setting up the Application Database – Apache Derby, teaches you how to set up the Apache Derby Application Database.

Appendix C, Spring Form Tag Library, shows the Spring form tag library provided by the Spring Web MVC framework. The Spring form tag library is a set of tags in the form of a tag library, which is used to construct views (web pages). This is an online appendix available at https://www.packtpub.com/sites/default/files/downloads/7368OS_AppendixC.pdf.
Introducing the Spring Framework

In this chapter, we'll introduce you to the Spring Framework. We'll also summarize some of the other features of Spring. We'll then discuss the Spring Architecture as well as the benefits of the Spring Framework. We will create your first application in Spring and will look into understanding the packaging structure of the Spring Framework. This chapter serves as a road map to the rest of this book.

The following topics will be covered in this chapter:

- Introducing Spring
- Spring Framework Architecture
- Benefits of the Spring Framework
- Creating a first application in Spring

Spring is an open source framework, which was created by Rod Johnson. He addressed the complexity of enterprise application development and described a simpler, alternative approach in his book *Expert One-on-One J2EE Design and Development*, Wrox.

Spring is now a long-time de-facto standard for Java enterprise software development. The framework was designed with developer productivity in mind, and it makes it easier to work with the existing Java and Java EE APIs. Using Spring, we can develop standalone applications, desktop applications, two-tier applications, web applications, distributed applications, enterprise applications, and so on.
Introducing the Spring Framework

As the title implies, we introduce you to the Spring Framework and then explore Spring’s core modules. Upon finishing this chapter, you will be able to build a sample Java application using Spring. If you are already familiar with the Spring Framework, then you might want to skip this chapter and proceed straight to Chapter 2, Inversion of Control in Spring.

Introducing Spring

Spring is a lightweight Inversion of Control (IoC) and aspect-oriented container framework. Historically, it was created to alleviate the complexity of the then J2EE standard, often giving an alternative model. Any Java EE application can benefit from the Spring Framework in terms of simplicity, loose coupling, and testability.

It remains popular due to its simple approach to building applications. It also offers a consistent programming model for different kinds of technologies, be they for data access or messaging infrastructure. The framework allows developers to target discrete problems and build solutions specifically for them.

The Spring Framework provides comprehensive infrastructure support for developing Java EE applications, where the Spring Framework handles the infrastructure and so developers can focus on application development.

Considering a scenario of JDBC application without using the Spring Framework, we have a lot of boilerplate code that needs to be written over and over again to accomplish common tasks. Whereas in Spring JDBC application, which internally uses plain JDBC, the JdbcTemplate class eliminates boilerplate code and allows the programmer to just concentrate on application-specific logics development.

- For a plain JDBC application without Spring, follow these steps:
  1. Register driver with the DriverManager service.
  2. Establish a connection with the database.
  3. Create a statement object.
  4. Prepare and execute an SQL query.
  5. Gather and process the result.
  6. Perform exception handling.
  7. Perform transaction management.
  8. Close JDBC object.
• For a Spring JDBC application (internally uses plain JDBC), follow these steps:
  1. Get access to `JdbcTemplate`.
  2. Prepare and execute an SQL query.
  3. Gather and process the result.

Spring’s main aim is to promote good programming practice such as coding to interfaces and make Java EE easier to use. It does this by enabling a **Plain Old Java Object (POJO)**-based programming model, which can be applicable in a wide range of development environments.

Technically, a POJO is any ordinary object that should not implement pre-specified interface or extend pre-specified class or contains annotation.

The following is the code for the `POJOClass.java` class:

```java
package com.packt.spring.chapter1;

/* This is a simple Java Class – POJO */
public class POJOClass {

  private String message;

  public String getMessage() {
    return this.message;
  }

  public void setMessage(String message) {
    this.message = message;
  }
}
```

In the preceding code snippet, we have `POJOClass` containing a field and corresponding getter and setter methods. This class is a POJO class as it is not extending or implementing any class or predefined interface of Spring API.

Spring is modular, allowing you to use only those parts that you need, without having to bring in extra complexity. The Spring Framework can be used either for all layer implementations or for the development of particular layer of an application.
Features of Spring
The Spring Framework contains the following features:

- **Lightweight**: Spring is described as a lightweight framework when it comes to size and transparency. A lightweight framework helps in reducing complexity in application code. It also helps in avoiding unnecessary complexity in its own functioning. A lightweight framework won't have a high startup time and will run in any environment. A lightweight framework also won't involve huge binary dependencies.

- **Non-intrusive**: This means that your domain logic code has no dependencies on the framework itself. The Spring Framework is designed to be non-intrusive. The object in a Spring-enabled application typically has no dependencies on any predefined interface or class given by Spring API. Thus, Spring can configure application objects that don't import Spring APIs.

- **Inversion of Control (IoC)**: Spring's container is a lightweight container that contains Spring beans and manages their life cycle. The core container of the Spring Framework provides an implementation for IoC supporting injection. IoC is an architectural pattern that describes the Dependency Injection needs to be done by external entity rather than creating the dependencies by the component itself. Objects are passively given their dependencies rather than creating dependent objects for themselves. Here, you describe which components need which service, and you don't directly connect your services and components together in your code. Let's consider an example: we have two classes Zoo and Animal, where Zoo has an object of Animal:

  - **Without Dependency Injection**: This is a common way to instantiate an object is with a new operator. Here, the Zoo class contains the object Animal that we have instantiated using a new operator, as shown in the following screenshot:

    ```java
    Zoo
    Animal animal;
    getAnimal () {
      animal = new Animal;
    }
    ```
° **With Dependency Injection**: Here, we supply the job of instantiating to a third party, as shown in following screenshot. Zoo needs the object of Animal to operate, but it outsources instantiation job to some third party that decides the moment of instantiation and the type to use in order to create the instance. This process of outsourcing instantiation is called dependency injection.

The Spring Framework promotes loose coupling by using the technique known as IoC. We'll talk more about IoC in Chapter 2, *Inversion of Control in Spring*.

- **Aspect-oriented Programming (AOP)**: This refers to the programming paradigm that isolates supporting functions from the main program's business logic. It allows a developer to build the core functionality of a system without being aware of additional requirements. AOP is used in the Spring Framework to provide declarative aspects such as transactions and security. Here, application objects perform business logic and are not responsible for other system concerns such as logging, security, auditing, locking, and event handling. AOP is a method of applying middleware services such as security service, and transaction management service on Spring's application.
Let's consider a payroll management application where there will be **Employee Service**, **HR Service**, and **Payroll Service**, as shown in the following figure, which will perform some functional requirement to the system such as add/update employee details, remove employee, browse employee details, and much more. While implementing business functionality, this type of application would also require nonfunctional capabilities such as role-based access and logging details. AOP leaves an application component to focus on business functionality. Here, the core application implements the business functionality and is covered with layers of functionality provided by AOP for security, logging, and transaction management.

Aspects can be added or removed as needed without changing your code. Spring aspects can be configured using its own IoC container. Spring AOP includes advisors that contain advice and pointcut filtering.

- **JDBC exception handling**: The JDBC abstraction layer of the Spring Framework provides an exception hierarchy. It shortens the error handling strategy in JDBC. This is one of the areas where Spring really helps in reducing the amount of boilerplate code we need to write in the exception handling. We'll talk more on Spring JDBC in Chapter 3, **DAO and JDBC in Spring**.

- **Spring MVC Framework**: This helps in building robust and maintainable web applications. It uses IoC that provides separation of controller logic. Spring MVC Framework, which is a part of the Spring Framework licensed under the term of Apache license, is an open source web application framework. Spring MVC Framework offers utility classes to handle some of the most common tasks in web application development. We'll discuss more about Spring Web MVC Framework in Chapter 5, **Spring Web MVC Framework**.
• Spring Security: This provides a declarative security mechanism for
Spring-based applications, which is a critical aspect of many applications.
We'll add Spring Security to our web applications in Chapter 6, Spring Security.

Other features of Spring
The following are the other features provided by the Spring Framework:

• Spring Web Services: This provides a contract-first web services model,
whereby service implementations are written to satisfy the service contract.
For more information, check out http://static.springsource.org/
spring-ws/sites/2.0.

• Spring Batch: This is useful when it's necessary to perform bulk operations
on data. For more information, refer to http://static.springsource.org/
spring-batch.

• Spring Social: Social networking, nowadays, is a rising trend on the
Internet, and more and more applications such as Facebook and Twitter
are being outfitted with integration into social-networking sites. To know
more, have a look at http://www.springsource.org/spring-social.

• Spring Mobile: Mobile applications are another significant area of
software development. Spring Mobile supports development of mobile
web applications. More information about Spring Mobile can be found
at http://www.springsource.org/spring-mobile.

Evolution of the Spring Framework
The Spring Framework is an open source framework that has multiple versions
released with the latest one being 4.x. The different versions of the Spring Framework
are as follows:

• Spring Framework 1.0: This version was released on March 2004, and the
first release was Spring Framework 1.0 RC4. The final and stable release was
Spring Framework 1.0.5. Spring 1.0 was a complete Java/J2EE application
framework, which covered the following functionalities:
  ° Spring Core: This is a lightweight container with various setter
    and constructor injection
  ° Spring AOP: This is an Aspect-oriented Programming (AOP)
    interception framework integrated with the core container
  ° Spring Context: This is an application context concept to provide
    resource loading
Introducing the Spring Framework

- **Spring DAO**: This is a generic DAO support that provides access to a generic data exception hierarchy with any data access strategy.
- **Spring JDBC**: This is a JDBC abstraction shorten error and resource handling.
- **Spring ORM**: This is a hibernate support `SessionFactory` management.
- **Spring Web**: This web MVC Framework integrates various view technologies.

- **Spring Framework 2.X**: The Spring Framework 2.0 was released in October 2006 and Spring 2.5 was released in November 2007. The Spring Framework 2.x release was based around two themes: simplicity and power. This provides you with the following features:
  - Improvements in the IoC container and AOP, including the `@AspectJ` annotation support for AOP development.
  - Introduction of bean configuration dialects.
  - XML-based configuration is reduced and XML schema support and custom namespace is introduced.
  - Annotation-driven configuration that requires component scanning to auto-detect annotated components in the classpath using annotations such as `@Component` or specialized annotations such as `@Repository`, `@Service`, and `@Controller`.
  - Introduces annotations such as `@RequestMapping`, `@RequestParam`, and `@ModelAttribute` for MVC controllers.

- **Spring Framework 3.0**: This version was released in December 2009. It makes the entire Spring code base to take advantage of the Java 5.0 technology. This provides you with the following features:
  - Supports REST in Spring MVC, which is one of the beautiful additions to the Spring Framework itself.
  - Introduces new annotations `@CookieValue` and `@RequestHeader` for pulling values from cookies and request headers, respectively. It also supports new XML namespace that makes easier to configure Spring MVC.
  - Task scheduling and asynchronous method execution with annotation support is introduced to this version.
• Spring Framework 3.0.5 is the latest update release, which was released on October 20, 2010. The Hibernate version 3.6 final is supported by this Spring release.

• **Spring Framework 3.1**: This version was released in December 2011. This release introduced many new exciting features that are related to cache abstraction, bean definition profiles, environment abstraction, PropertySource abstraction, and a lot more. This provides you with the following features:
  - Introduces Cache Abstraction to add caching concept to any existing application using `@Cacheable` annotation.
  - Introduces annotation called `@Profile`, which is used while applying configuration classes.
  - Introduces `PropertySource` that is an abstraction performed over any different source of the key-value pairs. In `DefaultEnvironment`, there are two configured `PropertySource` objects: `System.getProperties()` and `System.getenv()`.
  - Hibernate 4.x is supported by this release through Java Persistence API (JPA). With this release, the JPA `EntityManagerFactory` can be bootstrapped without `persistence.xml` or other metadata files.
  - Introduces `@RequestPart` annotation to provide access to multipart form-data content on the controller method arguments.
  - Introduces the `c:namespace` to support constructor injection.

• **Spring Framework 3.2.x**: This version was released in November 2013. This release introduced the following new features and enhancements to earlier features:
  - Servlet 3-based asynchronous request processing is supported in this release.
  - Supports Java 7 features.
  - Testing of Spring MVC applications without a Servlet container is supported in this release. Here, `DispatcherServlet` is used for server-side REST tests and `RestTemplate` for client-side REST tests.
  - `ContentNegotiationStrategy` is introduced to resolve the requested media types from an incoming request. It also supports Jackson JSON 2 library.
Introducing the Spring Framework

- Method annotated with @ExceptionHandler, @InitBinder, and @ModelAttribute can be added to a class annotated with the @ControllerAdvice annotation.
- The @MatrixVariable annotation for extracting matrix variables from the request URI is introduced.
- The @DateTimeFormat annotation to remove dependency on the Joda-Time library is introduced.

- **Spring Framework 4.x**: This version supports a few new features. Improvements in Spring 4.X include support for Java SE 8, Groovy 2, and a few aspects of Java EE7. This provides you with the following features:
  - Supports external bean configuration using a Groovy DSL
  - Auto-wiring is based on generic types
  - Introduces the @Description annotation
  - Introduces @Conditional that can conditionally filter the beans.
  - Introduces the @Jms annotation to support annotation-driven endpoint
  - Catching support is revisited, provided CacheResolver to resolve caches at runtime
  - Added new testing features such as SQL Script execution, bootstrap strategy, and so on
  - Added lightweight messaging and WebSocket-style architectures

**Spring Framework Architecture**

Spring packaging is modular, allowing you to pick and choose the modules that are applicable to you, without any need to bring in the rest. The following section gives you a detailed explanation about different modules available in the Spring Framework. The following figure shows you a complete overview of the framework and modules supported by the Spring Framework:
Spring Core Container

Spring Core Container consists of the core, beans, context, and expression language modules, as shown in the preceding figure. Let’s discuss these in detail as follows:

- **Core module**: This module of Spring Core Container is the most important component of the Spring Framework. It provides features such as IoC and Dependency Injection. The idea behind IoC is similar to the Hollywood principle: "Don't call me, I'll call you." Dependency Injection is the basic design principle in Spring Core Container that removes explicit dependence on container APIs.
Introducing the Spring Framework

- **Beans module**: The bean module in Spring Core Container provides BeanFactory, which is a generic factory pattern that separates the dependencies such as initialization, creation, and access of the objects from your actual program logic. BeanFactory in Spring Core Container supports the following two scopes modes of object:
  - **Singleton**: In singleton, only one shared instance of the object with a particular name will be retrieved on lookup. Spring singleton returns a same bean instance per Spring IoC container. Each time you call `getBean()` on `ApplicationContext`, Spring singleton returns the same bean instance.
  - **Prototype or non-singleton**: In prototype, each retrieval results in the creation of a brand new instance. Each time you call `getBean()` on `ApplicationContext`, Spring prototype creates a separate bean instance.

- **Context module**: An `ApplicationContext` container loads Spring bean definitions and wires them together. The `ApplicationContext` container is the focal point of the Context module. Hierarchical context is also one of the focal points of this API. `ApplicationContext` supports the Message lookup, supporting internationalization (i18N) messages.

- **Expression language**: Spring Expression Language (SpEL) is a powerful expression language supporting the features for querying and manipulating an object graph at runtime. SpEL can be used to inject bean or bean property in another bean. SpEL supports method invocation and retrieval of objects by name from IoC container in Spring.

**The AOP module**

Spring’s Aspect-oriented Programming (AOP) module is one of the main paradigms that provide an AOP implementation. Spring AOP module is a proxy-based framework implemented in Java. The Spring Framework uses AOP for providing most of the infrastructure logic in it.

AOP is a mechanism that allows us to introduce new functionalities into an existing code without modifying it design. AOP is used to weave cross-cutting aspects into the code. The Spring Framework uses AOP to provide various enterprise services, such as security in an application. The Spring AOP framework is configured at runtime.

Spring integrates with AspectJ, which is an extension of AOP. AspectJ lets programmers define special constructs called Aspects, which contains several entities unavailable to standard classes.
Data access/integration

Spring’s data access addresses common difficulties developers face while working with databases in applications.

- **JDBC module**: The Spring Framework provides solution for various problems identified using JDBC as low-level data access. The JDBC abstraction framework provided under the Spring Framework removes the need to do tedious JDBC-related coding. The central class of Spring JDBC abstraction framework is the `JdbcTemplate` class that includes the most common logic in using the JDBC API to access data such as handling the creation of connection, statement creation, statement execution, and release of resource. The `JdbcTemplate` class resides inside the `org.springframework.jdbc.core` package.

- **ORM module**: The **Object-relational mapping (ORM)** module of the Spring Framework provides a high-level abstraction for ORM APIs, including JPA and Hibernate. Spring ORM module reduces the complexity by avoiding the boilerplate code from application.

- **OXM module**: Spring OXM module stands for Spring Object XML Mappers, which supports Object/XML mapping. It also supports integration with Castor, JAXB, XmlBeans, and the XStream framework.

Most applications need to integrate or provide services to other applications. One common requirement is to exchange data with other systems, either on a regular basis or in real time. In terms of the data format, XML is the most commonly used format. As a result, there exists a common need to transform a JavaBean into XML format and vice versa.

Spring supports many common Java-to-XML mapping frameworks and, as usual, eliminates the need for directly coupling to any specific implementation. Spring provides common interfaces for marshalling (transforming JavaBeans into XML) and unmarshalling (transforming XML into Java objects) for DI into any Spring beans. Spring also has modules to convert data to and from JSON, in addition to OXM.

- **JMS module**: The **Java Messaging Service (JMS)** module comprises features to produce and consume messages. It is a **Java Message Oriented Middleware (MOM)** API for sending messages between two or more clients. JMS is a specification that describes a common way for Java program to create, send, and read distributed enterprise messages.

  - **Spring Java mail**: The `org.springframework.mail` package is the root package that provides mail support in the Spring Framework. It handles electronic mail.
Introducing the Spring Framework

- **Transaction module**: The Spring transaction module provides abstraction mechanism to supports programmatic and declarative transaction management for classes.

The Web module
The Web module consists of the Web, Servlet, Struts, and Portlet modules.

- **Web module**: The Spring Web module builds on the application context module and includes all the support for developing robust and maintainable web application in a simplified approach. It also supports multipart file-upload functionality.
- **Servlet module**: In Spring, the Servlet module contains Model-View-Controller (MVC) implementation that helps to build enterprise web applications. In Spring Framework, the MVC provides clean separation between binding request parameter, business objects, and controller logic.
- **Struts module**: The Web Struts module supports integration of Struts Web tier within a Spring application. It also supports configuration of Struts Actions using Spring Dependency Injection.
- **Portlet module**: Spring Portlet supports for easier development of web application using Spring. Portlet is managed by the Portlet container, similar to the web container. Portlet is used in the UI layer for displaying contents from data source for end user.

The Test module
In the Spring Framework, the Test module helps to test applications developed using the Spring Framework, either using JUnit or TestNG. It also helps in creating mock object to perform unit testing in isolation. It supports running integration tests outside the application server. We'll look at Spring's Test module in *Chapter 7, Spring Testing*.

Benefits of the Spring Framework
The following is the list of a few great benefits of using the Spring Framework:

- Spring is a powerful framework, which address many common problems in Java EE. It includes support for managing business objects and exposing their services to presentation tier component.
- It facilitates good programming practice such as programming using interfaces instead of classes. Spring enables developers to develop enterprise applications using POJO and POJI model programming.
• It is modular, allowing you to use only those parts that you need. It allows us to just choose any part of it in isolation.
• It supports both XML- and annotation-based configuration.
• Spring provides a lightweight container that can be activated without using web server or application server software.
• It gives good support for IoC and Dependency Injection results in loose coupling.
• The Spring Framework supports JDBC framework that improves productivity and reduces the error.
• It provides abstraction on ORM software to develop the ORM persistence logic.
• The Spring Web MVC framework provides powerful and a flexible Web framework as an alternative to Struts and other framework.
• The Spring Test module provides support for an easy-to-test code.

Creating an application in Spring

Before we create an application in Spring, first we need to obtain Spring Library. We can download the Spring distribution ZIP files that are available in the Spring Maven Repository. Else, we can simply add the dependencies for Spring into project’s pom.xml file whenever we use Maven for application development.

Spring packaging is modular, allowing you to pick and choose the component you want to use in your application. Spring comes with a large selection of sample applications that can be referred while building your application.

Obtaining Spring JAR files

• Downloading Spring distribution ZIP files: The complete Spring Framework library can be downloaded by opening the link http://repo.spring.io/release/org/springframework/spring/ and selecting the appropriate subfolder for the version needed for your application development. Distribution ZIP files end with dist.zip, for example, spring-framework-4.1.4.RELEASE-dist.zip.

While writing this book, the latest version was Spring Framework 4.1.4.

Download the package and extract it. Under the lib folder, you will find a list of Spring JAR files that represents each Spring module.
Introducing the Spring Framework

- **Checking Spring out of GitHub:** You can check out the latest version of code from Spring’s GitHub repository at https://github.com/spring-projects/spring-framework.

To check out the latest version of the Spring code, first install Git from http://git-scm.com/, open the Git Bash tool, and run the following command:

```bash
git clone https://github.com/spring-projects/spring-framework
```

**Understanding Spring packaging**

After extracting the downloaded Spring Framework ZIP file, you will get the directory structure, as shown in the following screenshot:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs</td>
<td>30-Dec-14 2:48 AM</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>libs</td>
<td>30-Dec-14 2:48 AM</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>schema</td>
<td>30-Dec-14 2:48 AM</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>license.txt</td>
<td>30-Dec-14 2:33 AM</td>
<td>Text Document</td>
<td>15 KB</td>
</tr>
<tr>
<td>notice.txt</td>
<td>30-Dec-14 2:33 AM</td>
<td>Text Document</td>
<td>1 KB</td>
</tr>
<tr>
<td>readme.txt</td>
<td>30-Dec-14 2:33 AM</td>
<td>Text Document</td>
<td>1 KB</td>
</tr>
</tbody>
</table>
```

The `spring-framework-4.14.RELEASE` folder, as shown in the preceding screenshot, contains `docs`, `libs`, and `schema` subfolders. The `libs` folder contains the Spring JAR files, as shown in the following screenshot:
Shown in the preceding screenshot is a list of JAR files required while developing applications using Spring. You can find more details on these JAR files at http://www.learnr.pro/content/53560-pro-spring/40 and http://agile-hero.iteye.com/blog/1684338.

**SpringSource Tool Suite**

SpringSource Tool Suite (STS) is a powerful Eclipse-based development environment for developing Spring application. The latest version of STS can be downloaded from http://spring.io/tools/sts. We will use STS IDE for all our examples in this book. The following screenshot shows a snapshot of an STS dashboard:

Let's now create a simple Spring application using Spring STS.
Introducing the Spring Framework

The Spring application

With basic understanding of the Spring Framework, we can now create a simple Spring application example. All the examples in this book have been written using the STS IDE.

We will write a simple Spring application that will print a greeting message to user. Do not worry if you do not fully understand all the code in this section; we'll go into much more detail on all the topics as we proceed through this book.

Creating a Spring project

The following steps will help you create your Spring project in STS:

1. The first step in creating a Spring application is to create a new Spring project in the STS IDE. Navigate to File | New | Spring Project, as shown in the following screenshot:

2. Name your Spring project SimpleSpringProject and select the Simple Java template, which creates a Simple Spring project using the Java build without a top-level package and with default Spring configuration and project natures, as shown in the following screenshot:
3. Then, click on **Finish**, which will create the project in a workspace.
Introducing the Spring Framework

Adding required libraries
Let's add the basic Spring JAR files to the build path of this Spring project:

1. Add the Spring Framework libraries and common logging API libraries to your project. The common login library can be downloaded from http://commons.apache.org/proper/commons-logging/download_logging.cgi. To add required libraries, right-click on the project named SimpleSpringProject and then click on the available options in the context menu, that is, Build Path | Configure Build Path to display the Java Build Path window, as shown in the following screenshot:

2. Now, use the Add External JARs button from the Libraries tab in order to include the following core JARs from the Spring Framework and common logging installation directories:
   - spring-aop-4.1.4.RELEASE
   - spring-aspects-4.1.4.RELEASE
   - spring-beans-4.1.4.RELEASE
   - spring-context-4.1.4.RELEASE
The Libraries tab is as shown in the following screenshot:
Now, you will have the content in your Project Explorer, as shown in the following screenshot:

Creating source files
Now let's create the actual source files under the SimpleSpringProject project:

1. First, create the packages named org.springframework.chapter1.service and org.springframework.chapter1.main, as shown in the following screenshot. To do this, right-click on src in package explorer section and navigate to New | Package.
2. Create a class called `MainClass.java` inside the `org.springframework.chapter1.main` package. Then, create an interface named `GreetingMessageService.java` and its implementation class `GreetingMessageServiceImpl.java` inside the package `org.springframework.chapter1.service`, as shown in the following screenshot:
The following is the content of interface `GreetingMessageService.java` and its implementation `GreetingMessageServiceImpl.java`:

- **GreetingMessageService.java**:
  ```java
  package org.springframework.chapter1.service;

  public interface GreetingMessageService {
      public String greetUser();
  }
  ``

- **GreetingMessageServiceImpl.java**:
  ```java
  package org.springframework.chapter1.service;
  import org.springframework.stereotype.Service;
  @Service
  public class GreetingMessageServiceImpl implements GreetingMessageService {
      public String greetUser() {
          return "Welcome to Chapter-1 of book Learning Spring Application Development";
      }
  }
  ``

The `GreetingMessageService` interface has a `greetUser()` method. The `GreetingMessageServiceImpl` class implements the `GreetingMessageService` interface and provides definition to the `greetUser()` method. This class is annotated with the `@Service` annotation, which will define this class as service class.

**Downloading the example code**

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

The following is the content of the file `MainClass.java`:

```java
package org.springframework.chapter1.main;

import org.springframework.chapter1.service.GreetingMessageService;
```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(
            "beans.xml");
        GreetingMessageService greetingMessageService = context.getBean(
            "greetingMessageServiceImpl", GreetingMessageService.class);
        System.out.println(greetingMessageService.greetUser());
    }
}

In MainClass.java, we are creating ApplicationContext using framework API, as shown in the following:

ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

This API loads Spring beans configuration file named beans.xml, which takes care of creating and initializing all the bean objects. We use the getBean() method of the created ApplicationContext to retrieve required Spring bean from the application context, as shown in the following:

GreetingMessageService greetingMessageService = context.getBean("greetingMessageServiceImpl", GreetingMessageService.class);

The getBean() method uses bean ID and bean class to return a bean object.

Creating the Spring bean configuration file

The Spring bean configuration file is used to configure the Spring beans in the Spring IoC container. As we have annotated the GreetingMessageServiceImpl class with @Service annotation, the next step is to add  <context:component-scan> in the bean configuration file. To do this, follow these steps:

1. Create a Spring Bean Configuration file under the src directory. To do this, right-click on src in package explorer section and then navigate to New | Spring Bean Configuration File.
2. Enter the bean name `beans` and click on **Next**, as shown in the following screenshot:
3. Select the context option and click on Finish, as shown in the following screenshot:

![Create a new Spring Bean Definition file](image-url)

4. Now the Spring bean configuration file is created. Add the following code to create an entry. The contents of the beans.xml file are as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">...
```

[27]
Introducing the Spring Framework

```xml
<beans
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.1.xsd">
  <context:component-scan base-package="org.springframework.chapter1.service"/>
</beans>
```

When the Spring application gets loaded into the memory, in order to create all the beans, the framework uses the preceding configuration file, as shown in the following screenshot:

The Spring bean configuration file can be named anything, but developers usually keep the name `beans.xml`. This Spring bean configuration file should be available in classpath.

```
<beans
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-4.1.xsd">
  <context:component-scan base-package="org.springframework.chapter1.service"/>
</beans>
```

The S in the upper-right corner of the project icon indicates it is a Spring Project.

Running the program

Once you are done with creating source files and beans configuration files, you are ready for the next step, that is, compiling and running your program.
To execute the example, run the `MainClass.java` file. Right-click on `MainClass.java` and navigate to **Run As | Java Application**. If everything goes fine, then it will print the following message in STS IDE's console, as shown in the following screenshot:

We have successfully created our first Spring application, where you learned how to create the Spring project and executed it successfully. We will see detailed examples in the next chapter.

**Exercise**
Q1. What is Spring?
Q2. List some of the features of Spring.
Q3. Explain different modules in the Spring Framework.

The answers to these are provided in *Appendix A, Solution to Exercises*.

**Summary**
In this chapter, you were introduced the Spring Framework and acquainted with its features. You took a look at the versions of Spring. Then, you studied the architecture, and different modules in the Spring Framework such as the Spring Core Container, Spring AOP, Spring data access/integration, and the Spring Web module and Test module. You also understood the benefits of the Spring Framework. Finally, you created an application in Spring and took a look on package structure of Spring.

In the next chapter, we'll explore IoC, Dependency Injection, and Spring Core Container service. We'll also see bean's life cycle and bean's scope.
Where to buy this book

You can buy Learning Spring Application Development from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.