Spring Framework

From Wikipedia, the free encyclopedia

The **Spring Framework** is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.

Spring Framework

Developer(s)	Pivotal Software
Initial release	1 October 2002
Stable release	4.3.4 [1] / November 7, 2016
Preview release	5.0.0 M3 / November 8, 2016
Repository	github.com/spring-projects/spring-framework (https://github.com/spring-projects/spring-framework)
Development status	Active
Written in	Java
Operating system	Cross-platform
Platform	Java Virtual Machine
Type	Application framework
License	Apache License 2.0
Website	spring.io (https://spring.io)

Contents

Version history

The first version was written by Rod Johnson, who released the framework with the publication of his book *Expert One-on-One J2EE Design and Development* in October 2002. The framework was first released under the Apache 2.0 license in June 2003. The first milestone release, 1.0, was released in March 2004, with further milestone releases in September 2004 and March 2005. The Spring 1.2.6 framework won a Jolt productivity award and a JAX Innovation Award in 2006.[2][3] Spring 2.0 was released in October 2006, Spring 2.5 in November 2007, Spring 3.0 in December 2009, Spring 3.1 in December 2011, and Spring 3.2.5 in November 2013.[4] Spring Framework 4.0 was released in December 2013.[5] Notable improvements in Spring 4.0 included support for Java SE 8, Groovy 2, some aspects of Java EE7, and WebSocket.

Spring Framework 4.2.0 was released on 31 July 2015 and was immediately upgraded to version 4.2.1, which was released on 01 Sept 2015.[6] It is *"compatible with Java 6, 7 and 8, with a focus on core refinements and modern web capabilities"*.[7]

Spring Framework 4.3 has been released on 10 June 2016. The 4.3.0.RC1 [8] version is available. It *"will be the final generation within the general Spring 4 system requirements (Java 6+, Servlet 2.5+), getting prepared for an extended 4.3.x support life until 2019"*.[9]

Spring 5 is announced to be built upon Reactive Streams compatible Reactor Core.[10]

Modules

The Spring Framework includes several modules that provide a range of services:

- Spring Core Container: this is the base module of Spring and provides spring containers (BeanFactory and ApplicationContext).[11]
- Aspect-oriented programming: enables implementing cross-cutting concerns.
- Authentication and authorization: configurable security processes that support a range of standards, protocols, tools and practices via the Spring Security sub-project (formerly Acegi Security System for Spring).
- Convention over configuration: a rapid application development solution for Spring-based enterprise applications is offered in the Spring Roo module
- Data access: working with relational database management systems on the Java platform using JDBC and object-relational mapping tools and with NoSQL databases
- Inversion of control container: configuration of application components and lifecycle management of Java objects, done mainly via dependency injection
- Messaging: configurative registration of message listener objects for transparent message-consumption from message queues via JMS, improvement of message sending over standard JMS APIs
- Model–view–controller: an HTTP- and servlet-based framework providing hooks for extension and customization for web applications and RESTful Web services.
- Remote access framework: configurative RPC-style marshalling of Java objects over networks supporting RMI, CORBA and HTTP-based protocols including Web services (SOAP)
- Transaction management: unifies several transaction management APIs and coordinates transactions for Java objects
- Remote management: configurative exposure and management of Java objects for local or remote configuration via JMX
- Testing: support classes for writing unit tests and integration tests

Inversion of control container (dependency injection)

Central to the Spring Framework is its inversion of control (IoC) container, which provides a consistent means of configuring and managing Java objects using reflection. The container is responsible for managing object lifecycles of specific objects: creating these objects, calling their initialization methods, and configuring these objects by wiring them together.

Objects created by the container are also called managed objects or beans. The container can be configured by loading XML files or detecting specific Java annotations on configuration classes. These data sources contain the bean definitions that provide the information required to create the beans.

Objects can be obtained by means of either dependency lookup or dependency injection.[12] Dependency lookup is a pattern where a caller asks the container object for an object with a specific name or of a specific type. Dependency injection is a pattern where the container passes objects by name to other objects, via either constructors, properties, or factory methods.

In many cases one need not use the container when using other parts of the Spring Framework, although using it will likely make an application easier to configure and customize. The Spring container provides a consistent mechanism to configure applications and integrates with almost all Java environments, from small-scale applications to large enterprise applications.

The container can be turned into a partially compliant EJB 3.0 container by means of the Pitchfork project. Some criticize the Spring Framework for not complying with standards.[13] However, SpringSource doesn't see EJB 3 compliance as a major goal, and claims that the Spring Framework and the container allow for more powerful programming models.[14] You do not create an object, but describe how they should be created, by defining it in the Spring configuration file. You do not call services and components, but tell which services and components must be called, by defining them in the Spring configuration files. This makes the code easy to maintain and easier to test through IoC.

Aspect-oriented programming framework

The Spring Framework has its own Aspect-oriented programming (AOP) framework that modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it would be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The Spring AOP framework also takes full advantage of the Spring container.

The Spring AOP framework is proxy pattern-based, and is configured at run time. This removes the need for a compilation step or load-time weaving. On the other hand, interception only allows for public method-execution on existing objects at a join point.

Compared to the AspectJ framework, Spring AOP is less powerful, but also less complicated. Spring 1.2 includes support to configure AspectJ aspects in the container. Spring 2.0 added more integration with AspectJ; for example, the pointcut language is reused and can be mixed with Spring AOP-based aspects. Further, Spring 2.0 added a Spring Aspects library that uses AspectJ to offer common Spring features such as declarative transaction management and dependency injection via AspectJ compile-time or load-time weaving. SpringSource also uses AspectJ AOP in other Spring projects such as Spring Roo and Spring Insight, with Spring Security also offering an AspectJ-based aspect library.

Spring AOP has been designed to make it able to work with cross-cutting concerns inside the Spring Framework. Any object which is created and configured by the container can be enriched using Spring AOP.

The Spring Framework uses Spring AOP internally for transaction management, security, remote access, and JMX.

Since version 2.0 of the framework, Spring provides two approaches to the AOP configuration:

- schema-based approach[15] and
- @AspectJ-based annotation style.[16]

```
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:mvc="http://www.springframework.org/schema/mvc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:aop="http://www.springframework.org/schema/aop"
   xmlns:context="http://www.springframework.org/schema/context"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">
```

The Spring team decided not to introduce new AOP-related terminology; therefore, in the Spring reference documentation and API, terms such as aspect, join point, advice, pointcut, introduction, target object (advised object), AOP proxy, and weaving all have the same meanings as in most other AOP frameworks (particularly AspectJ).

Data access framework

Spring's data access framework addresses common difficulties developers face when working with databases in applications. Support is provided for all popular data access frameworks in Java: JDBC, iBatis/MyBatis, Hibernate, JDO, JPA, Oracle TopLink, Apache OJB, and Apache Cayenne, among others.

For all of these supported frameworks, Spring provides these features

- Resource management - automatically acquiring and releasing database resources
- Exception handling - translating data access related exception to a Spring data access hierarchy
- Transaction participation - transparent participation in ongoing transactions
- Resource unwrapping - retrieving database objects from connection pool wrappers
- Abstraction for BLOB and CLOB handling

All these features become available when using template classes provided by Spring for each supported framework. Critics have said these template classes are intrusive and offer no advantage over using (for example) the Hibernate API directly.[17] In response, the Spring developers have made it possible to use the Hibernate and JPA APIs directly. This however requires transparent transaction management, as application code no longer assumes the responsibility to obtain and close database resources, and does not support exception translation.

Together with Spring's transaction management, its data access framework offers a flexible abstraction for working with data access frameworks. The Spring Framework doesn't offer a common data access API; instead, the full power of the supported APIs is kept intact. The Spring Framework is the only framework available in Java that offers managed data access environments outside of an application server or container.

While using Spring for transaction management with Hibernate, the following beans may have to be configured:

- A Data Source like `com.mchange.v2.c3p0.ComboPooledDataSource` or `org.apache.commons.dbcp.BasicDataSource`
- A SessionFactory like `org.springframework.orm.hibernate3.LocalSessionFactoryBean` with a DataSource attribute
- A HibernateProperties like `org.springframework.beans.factory.config.PropertiesFactoryBean`
- A TransactionManager like `org.springframework.orm.hibernate3.HibernateTransactionManager` with a SessionFactory attribute

Other points of configuration include:

- An AOP configuration of cutting points.
- Transaction semantics of AOP advice.

Transaction management framework

Spring's transaction management framework brings an abstraction mechanism to the Java platform. Its abstraction is capable of:

- working with local and global transactions (local transaction does not require an application server)
- working with nested transactions
- working with savepoints
- working in almost all environments of the Java platform

In comparison, JTA only supports nested transactions and global transactions, and requires an application server (and in some cases also deployment of applications in an application server).

The Spring Framework ships a PlatformTransactionManager for a number of transaction management strategies:

- Transactions managed on a JDBC Connection
- Transactions managed on Object-relational mapping Units of Work
- Transactions managed via the JTA TransactionManager and UserTransaction
- Transactions managed on other resources, like object databases

Next to this abstraction mechanism the framework also provides two ways of adding transaction management to applications:

- Programmatically, by using Spring's TransactionTemplate
- Configuratively, by using metadata like XML or Java annotations (@Transactional, etc.)

Together with Spring's data access framework — which integrates the transaction management framework — it is possible to set up a transactional system through configuration without having to rely on JTA or EJB. The transactional framework also integrates with messaging and caching engines.

Model–view–controller framework

The Spring Framework features its own MVC web application framework, which wasn't originally planned. The Spring developers decided to write their own Web framework as a reaction to what they perceived as the poor design of the (then) popular Jakarta Struts Web framework,[18] as well as deficiencies in other available frameworks. In particular, they felt there was insufficient separation between the presentation and request handling layers, and between the request handling layer and the model.[19]

Like Struts, Spring MVC is a request-based framework. The framework defines strategy interfaces for all of the responsibilities that must be handled by a modern request-based framework. The goal of each interface is to be simple and clear so that it's easy for Spring MVC users to write their own implementations, if they so choose. MVC paves the way for cleaner front end code. All interfaces are tightly coupled to the Servlet API. This tight coupling to the Servlet API is seen by some as a failure on the part of the Spring developers to offer a high-level abstraction for Web-based applications. However, this coupling makes sure that the features of the Servlet API remain available to developers while offering a high abstraction framework to ease working with said API.

The DispatcherServlet class is the front controller[20] of the framework and is responsible for delegating control to the various interfaces during the execution phases of an HTTP request.

The most important interfaces defined by Spring MVC, and their responsibilities, are listed below:

- Controller: comes between Model and View to manage incoming requests and redirect to proper response. It acts as a gate that directs the incoming information. It switches between going into model or view.
- HandlerAdapter: execution of objects that handle incoming requests

- HandlerInterceptor: interception of incoming requests comparable, but not equal to Servlet filters (use is optional and not controlled by DispatcherServlet).
- HandlerMapping: selecting objects that handle incoming requests (handlers) based on any attribute or condition internal or external to those requests
- LocaleResolver: resolving and optionally saving of the locale of an individual user
- MultipartResolver: facilitate working with file uploads by wrapping incoming requests
- View: responsible for returning a response to the client. Some requests may go straight to view without going to the model part; others may go through all three.
- ViewResolver: selecting a View based on a logical name for the view (use is not strictly required)

Each strategy interface above has an important responsibility in the overall framework. The abstractions offered by these interfaces are powerful, so to allow for a set of variations in their implementations, Spring MVC ships with implementations of all these interfaces and together offers a feature set on top of the Servlet API. However, developers and vendors are free to write other implementations. Spring MVC uses the Java `java.util.Map` interface as a data-oriented abstraction for the Model where keys are expected to be string values.

The ease of testing the implementations of these interfaces seems one important advantage of the high level of abstraction offered by Spring MVC. DispatcherServlet is tightly coupled to the Spring inversion of control container for configuring the web layers of applications. However, web applications can use other parts of the Spring Framework—including the container—and choose not to use Spring MVC.

Remote access framework

Spring's Remote Access framework is an abstraction for working with various RPC-based technologies available on the Java platform both for client connectivity and marshalling objects on servers. The most important feature offered by this framework is to ease configuration and usage of these technologies as much as possible by combining inversion of control and AOP.

The framework also provides fault-recovery (automatic reconnection after connection failure) and some optimizations for client-side use of EJB remote stateless session beans.

Spring provides support for these protocols and products out of the box

- HTTP-based protocols
  - Hessian: binary serialization protocol, open-sourced and maintained by CORBA-based protocols
  - RMI (1): method invocations using RMI infrastructure yet specific to Spring
  - RMI (2): method invocations using RMI interfaces complying with regular RMI usage
  - RMI-IIOP (CORBA): method invocations using RMI-IIOP/CORBA
- Enterprise JavaBean client integration
  - Local EJB stateless session bean connectivity: connecting to local stateless session beans
  - Remote EJB stateless session bean connectivity: connecting to remote stateless session beans
- SOAP
  - Integration with the Apache Axis Web services framework

Apache CXF provides integration with the Spring Framework for RPC-style exporting of object on the server side.

Both client and server setup for all RPC-style protocols and products supported by the Spring Remote access framework (except for the Apache Axis support) is configured in the Spring Core container.

There is alternative open-source implementation (Cluster4Spring) of a remoting subsystem included into Spring Framework that is intended to support various schemes of remoting (1-1, 1-many, dynamic services discovering)…

Convention-over-configuration rapid application development

Spring Boot

Spring Boot is Spring's convention-over-configuration solution for creating stand-alone, production-grade Spring-based Applications that you can "just run".[21] It takes an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration. Features:

- Create stand-alone Spring applications
- Embed Tomcat or Jetty directly (no need to deploy WAR files)
- Provide opinionated 'starter' POMs to simplify your Maven configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

Spring Roo

Spring Roo provides an alternative, code-generation based approach at using convention-over-configuration to rapidly build applications in Java. It currently supports Spring Framework, Spring Security and Spring Web Flow. Roo differs from other rapid application development frameworks by focusing on:

- Extensibility (via add-ons)
- Java platform productivity (as opposed to other languages)
- Lock-in avoidance (Roo can be removed within a few minutes from any application)
- Runtime avoidance (with associated deployment advantages)
- Usability (particularly via the shell features and usage patterns)

Batch framework

Spring Batch is a framework for batch processing that provides reusable functions that are essential in processing large volumes of records, including:

- logging/tracing
- transaction management
- job processing statistics
- job restart

It also provides more advanced technical services and features that will enable extremely high-volume and high performance batch jobs through optimizations and partitioning techniques. Spring Batch is a framework for batch processing – execution of a series of jobs. In Spring Batch, A job consists of many steps and each step consists of a READ-PROCESS-WRITE task or single operation task (tasklet).

For "READ-PROCESS-WRITE" process, it means "read" data from the resources (csv, xml or database), "process" it and "write" it to other resources (csv, xml and database). For example, a step may read data from a CSV file, process it and write it into the database. Spring Batch provides many made Classes to read/write CSV, XML and database.

For "single" operation task (tasklet), it means doing single task only, like clean up the resources after or before a step is started or completed.

And the steps can be chained together to run as a job.

Integration framework

Spring Integration is a framework for Enterprise application integration that provides reusable functions that are essential in messaging, or event-driven architectures.

- routers - routes a message to a message channel based on conditions
- transformers - converts/transforms/changes the message payload and creates a new message with transformed payload
- adapters - to integrate with other technologies and systems (HTTP, AMQP, JMS, XMPP, SMTP, IMAP, FTP (as well as FTPS/SFTP), file systems, etc.)
- filters - filters message based on criteria. If the criteria are not met, message is dropped
- service activators - invoke an operation on a service object
- management and auditing

Spring Integration supports pipe-and-filter based architectures.

Criticisms

The Spring Framework has received some criticism for what some developers perceive to be an over-reliance on XML by Spring's container. Since version 3.0.0, however, developers have been able to specify all or part of an application context through annotations or Java code. Spring Boot makes heavy use of this to minimize the amount of configuration that must be written. Furthermore, the Spring Tool Suite (STS), built on top of Eclipse, provides code-completion, validation, contextual information, and graphical visualizations when editing Spring XML configuration files.

See also

- Apache Tapestry
- Google Guice
- Hibernate (framework)

References

1. http://projects.spring.io/spring-framework Spring Framework Projects
2. Jolt winners 2006 (http://www.ddj.com/architect/187900423?pgno=10)
3. JAX Innovation Award Gewinner 2006 (http://jax-award.de/jax_award06/gewinner_de.php)
4. "Spring Framework 3.2.5 Released". *Official Spring website*. 7 Nov 2013. Retrieved 16 October 2016.
5. SpringSource.org (https://spring.io/blog/2013/12/12/announcing-spring-framework-4-0-ga-release/)
6. Spring Official Blog (http://spring.io/blog/2015/07/31/spring-framework-4-2-goes-ga)
7. Spring Official Blog (http://spring.io/blog/2015/07/31/spring-framework-4-2-goes-ga)
8. Spring release blog (https://repo.spring.io/milestone/)
9. Spring Official Blog (http://spring.io/blog/2015/07/31/spring-framework-4-2-goes-ga)
10. Reactive Spring (http://spring.io/blog/2016/02/09/reactive-spring)
11. Spring Framework documentation for the Core Container (http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/beans.html#beans-introduction)
12. What is the difference between the depencylookup and dependency injection - Spring Forum (http://forum.springsource.org/showthread.php?79731-What-is-the-difference-between-the-depencylookup-and-dependency-injection). Forum.springsource.org (2009-10-28). Retrieved on 2013-11-24.
13. Spring VS EJB3 (http://www.andygibson.net/blog/index.php/2008/08/28/is-spring-between-the-devil-and-the-ejb)
14. "Pitchfork FAQ". Retrieved 2006-06-06.
15. Spring AOP XML Configuration (http://howtodoinjava.com/spring/spring-aop/spring-aop-aspectj-xml-configuration-example/)
16. AspectJ Annotation Configuration (http://howtodoinjava.com/spring/spring-aop/spring-aop-aspectj-example-tutorial-using-annotation-config/)
17. Hibernate VS Spring (http://houseofhaug.wordpress.com/2005/08/12/hibernate-hates-spring)

18. Introduction to the Spring Framework (http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework)
19. Johnson, Expert One-on-One J2EE Design and Development, Ch. 12. et al.
20. Patterns of Enterprise Application Architecture: Front Controller (http://www.martinfowler.com/eaaCatalog/frontController.html)
21. Spring Boot (http://projects.spring.io/spring-boot/)

Bibliography

- Mak, Gary (September 1, 2010). *Spring Recipes: A Problem-Solution Approach* (Second ed.). Apress. p. 1104. ISBN 1-4302-2499-1.
- Walls, Craig (November 28, 2010). *Spring in Action* (Third ed.). Manning. p. 700. ISBN 1-935182-35-8.
- Walls, Craig; Breidenbach, Ryan (August 16, 2007). *Spring in Action* (Second ed.). Manning. p. 650. ISBN 1-933988-13-4.
- Johnson, Rod; Höller, Jürgen; Arendsen, Alef; Risberg, Thomas; Sampaleanu, Colin (July 8, 2005). *Professional Java Development with the Spring Framework* (First ed.). Wrox Press. p. 672. ISBN 0-7645-7483-3.
- Harrop, Rob; Machacek, Jan (January 31, 2005). *Pro Spring* (First ed.). Apress. p. 832. ISBN 1-59059-461-4.
- Johnson, Rod; Jürgen, Höller (October 23, 2002). *J2EE Development without EJB* (First ed.). Wrox Press. p. 768. ISBN 0-7645-5831-5.
- Johnson, Rod (October 2002). *Expert One-on-one J2EE Design and Development* (First ed.). Wrox Press. p. 750. ISBN 0-7645-4385-7.
- Pollack, Mark; Gierke, Oliver; Risberg, Thomas; Brisbin, Jon; Hunger, Michael (October 31, 2012). *Spring Data* (First ed.). O'Reilly. p. 316. ISBN 978-1449323950.
- Sarin, Ashish (December 10, 2012). *Getting started with Spring Framework* (First ed.). Self-published. p. 324. ISBN 978-1480013971.
- Long, Josh (August 27, 2013). *Spring Framework LiveLessons* (First ed.). Addison-Wesley Professional. pp. 4+ Hours. ISBN 978-0-13-346307-1.

External links

- Official website (https://spring.io)
- Spring Tutorials & Code Examples (http://memorynotfound.com/category/spring-framework/)
- Spring MVC Samples (http://www.kscodes.com/category/spring-mvc/)

The Wikibook *Java Programming* has a page on the topic of: ***Spring framework***

Retrieved from "https://en.wikipedia.org/w/index.php?title=Spring_Framework&oldid=753355402"

Categories: Aspect-oriented programming | Java enterprise platform | Web frameworks

---