Spring Core

Presentation by

Hung Nguyen Huy

Content

1. What is a Framework?
2. What is Spring Framework?
3. Spring Framework History
4. Spring Framework Architecture
5. Why Spring?
6. Spring Framework Ecosystem

What is a Framework?

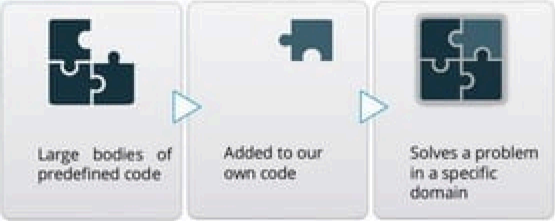
What is a Software Framework?

- A set of libraries and classes, which provides built-in generic functionalities, dealt with standard low-level details of a working system.
- A reusable software environment.
- Can be extended by additional users-written code to provide specific functionalities.
- Enforces the adherence to a coding standards and consistent design approaches which are pre-defined by the Framework itself.

spring

What is a Software Framework?

- Each programming language has at least one universal, reusable framework.
- Frameworks are fully layered workflow environment. More than just code, they defines the flow of control for the application.
- They can includes:
  - Libraries
  - APIs
  - Compilers
  - Tool sets
  - Security
  - Caching
  - ...



Large bodies of predefined code

Added to our own code

Solves a problem in a specific domain

Software Framework Advantages

- Provides built-in generic functionalities
    - For example: Security, request handling, caching, logging,...
- Reduce overall development effort and time
    - Developers can focus on writing code just for their specific application requirements
- Allows the applications to be implemented in a standard structure
    - This improves the maintainability of applications

Software Framework Disadvantages

- It takes time to learn how to use the framework.
- Using a framework increases the size of the program.
- Need to follow some coding standards in order to utilize the framework.
- It is unsuitable for writing small programs which can be written quickly without using any framework.

What is Spring Framework?

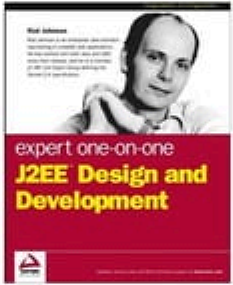
What is Spring Framework?

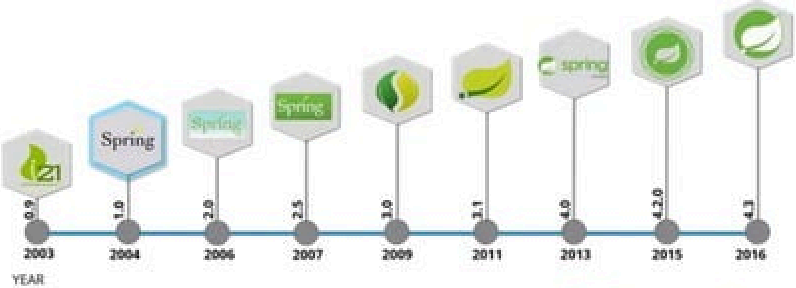
- Open source Java application framework (since 2003)
- Supports any kind of Java Applications
  - Special support for J2EE applications
- Foundation: the Core Container
  - Inversion of Control (IoC) container
  - Dependency Injection pattern
- Spring handles the infrastructure, so you can focus on your application.

Spring Framework History

Spring History

- In October 2002, **Rod Johnson** wrote his famous book.
- First release in June 2003 (Interface21) under Apache license
- First milestone release Spring 1.0 in 2014
- Spring 2.0
  - Simplified XML configuration
- Spring 2.5
  - Annotation configuration
- Spring 3.0
  - Support for Java 7
  - Servlet 3.0 specs
  - Java Configuration
- Spring 4.0
  - Support Java 8
- Spring 5.0 → 2017
  - Featuring a new reactive web framework


expert one-on-one
J2EE Design and Development



Spring History

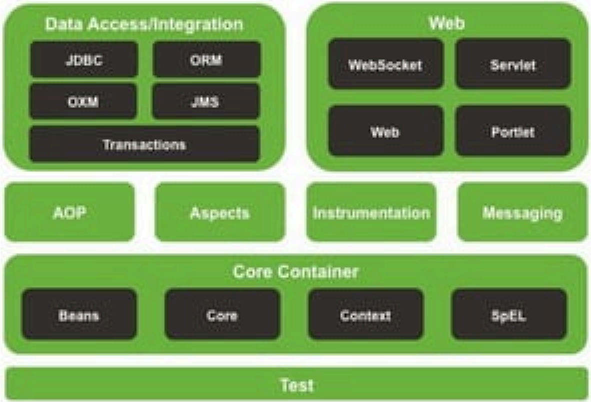
Spring Boot → 2014

- Makes it particularly easy to create and config a Spring application.
- Automatically configure Spring and 3rd party libraries whenever possible.
- Spring Boot hide so much of what going on under the hood.
- Spring Boot **is still Spring** under the hood, it bases on the same **core** of Spring.

Spring Framework Architecture

Spring Framework Architecture

- Spring Framework consists of features organized into about 20 modules.
- These modules are generalized into following layers:
    - Core Container
    - Data Access/ Integration
    - Web
    - AOP (Aspect Oriented Programming)
    - Instrumentation
    - Test



Data Access/Integration

JDBC

ORM

OXM

JMS

Transactions

Web

WebSocket

Servlet

Web

Portlet

AOP

Aspects

Instrumentation

Messaging

Core Container

Beans

Core

Context

SpEL

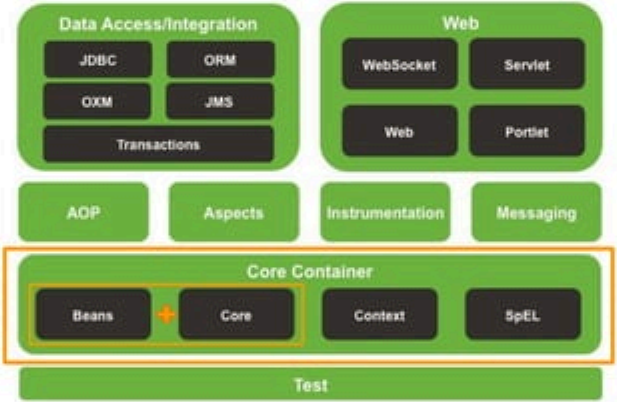
Test

spring

Core Container

Core and Beans modules

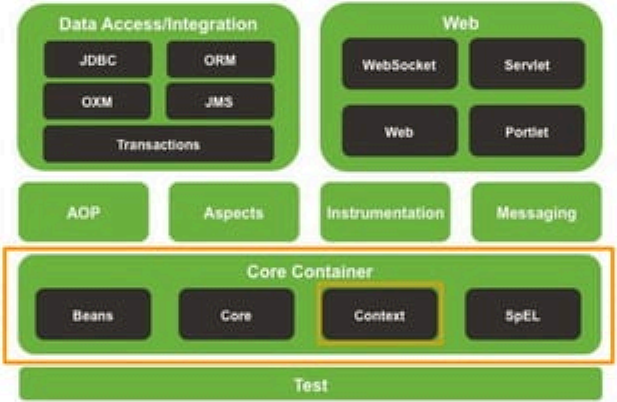
- Provide the fundamental parts of the framework, including the **IoC** and **Dependency Injection** features.
- The **Bean** module provides **BeanFactory**, which is a sophisticated implementation of the factory pattern.
- Removes the need for programmatic singletons
- Allows to decouple the configuration and specification of dependencies from your actual program logic.



Core Container

Context module

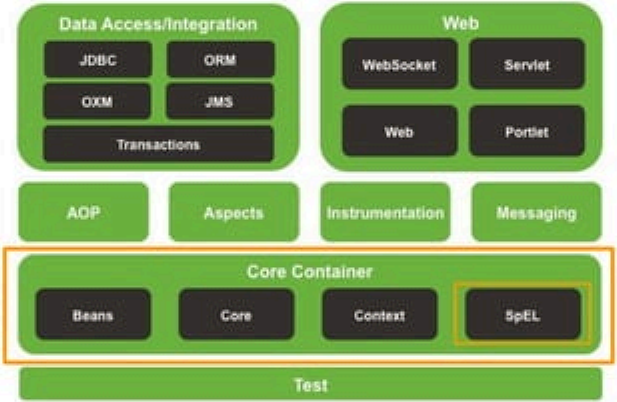
- Builds on the solid base provided by the Core and Beans modules.
- It is a medium to access any objects defined and configured.
- The **ApplicationContext** interface is the focal point of the Context module.



Core Container

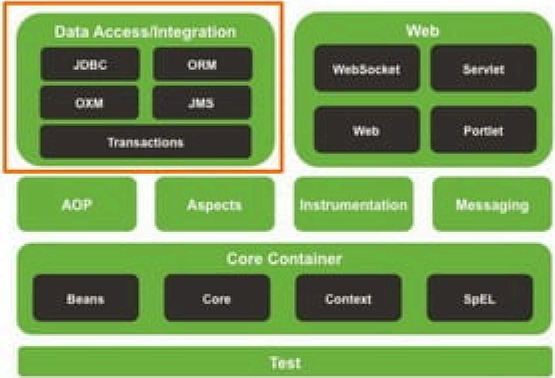
SpEL (Expression Language) module

- Provides a powerful expression language for **querying** and **manipulating** an object graph at runtime.
- Supports **setting** and **getting** property values, method invocation, logical and arithmetic operators, named variables, and retrieval of objects by name from Spring's IoC container, etc...



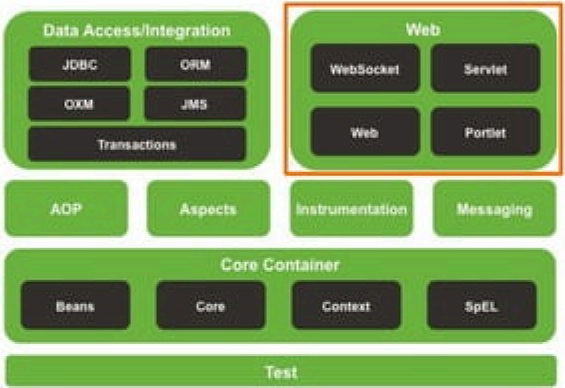
Data Access/Integration

- **JDBC** module provides a **JDBC-abstraction layer** that removes the need to do tedious JDBC coding.
- **ORM** module provides **integration layers** for popular **object-relational mapping APIs**, including JPA, JDO, and Hibernate.
- **OXM** module provides an abstraction layer that supports **Object/XML mapping** implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
- **JMS** (Java Messaging Service) module contains features for producing and consuming messages.
- **Transaction** module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.



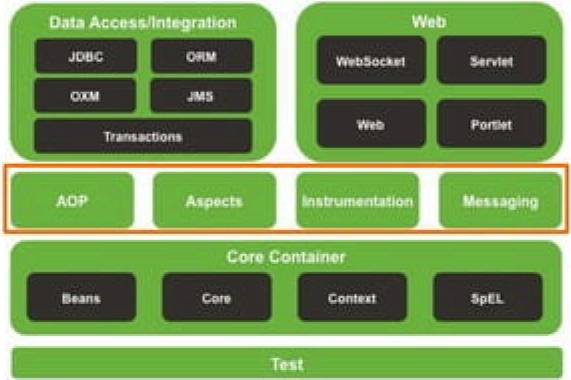
Web

- **Web** module provides basic web-oriented integration features and the initialization of the IoC container using servlet listeners and a web application context.
- **Servlet** module contains Spring MVC implementation for web applications.
- **Web-Socket** module provides support for WebSocket-based, 2-way communication between the client and the server in web applications.
- **Web-Portlet** module provides the MVC implementation to be used in a portlet environment.



Miscellaneous

- **AOP** module provides an aspect-oriented programming implementation allowing you to cleanly decouple code that implements functionality that should be separated.
- **Aspects** module provides integration with AspectJ (a powerful AOP framework).
- **Instrumentation** module provides class instrumentation support and class loader implementations.
- **Messaging** module provides support for STOMP as the WebSocket sub-protocol.
- **Test** module supports the unit testing and integration testing of Spring components with JUnit or TestNG.



Why Spring?

Why Spring?

These 3 are basically the main reasons for Spring's popularity:

1. Simplicity
2. Testability
3. Loose Coupling

spring

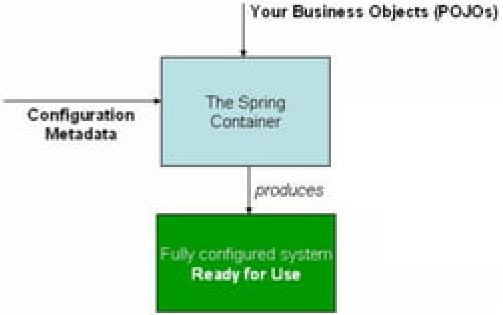
Simplicity

Spring Framework is simple because its non-invasive as it uses POJO and POJI models.

- **POJO** (Plain Old Java Objects): A Java class not coupled with any technology or any framework is called POJO.
- **POJI** (Plain Old Java Interfaces): A Java interface not coupled with any technology or any framework is called POJI.

Testability

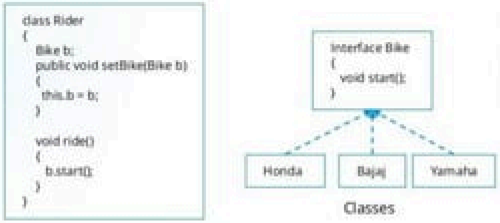
Actually for writing the Spring application, server (Container) is not mandatory because it has it own container to run the applications.



Loose Coupling

Spring Framework is loosely coupled because it has concepts like Dependency Injection, AOP etc.

These features help in reducing dependency and increasing the modularity within the code.



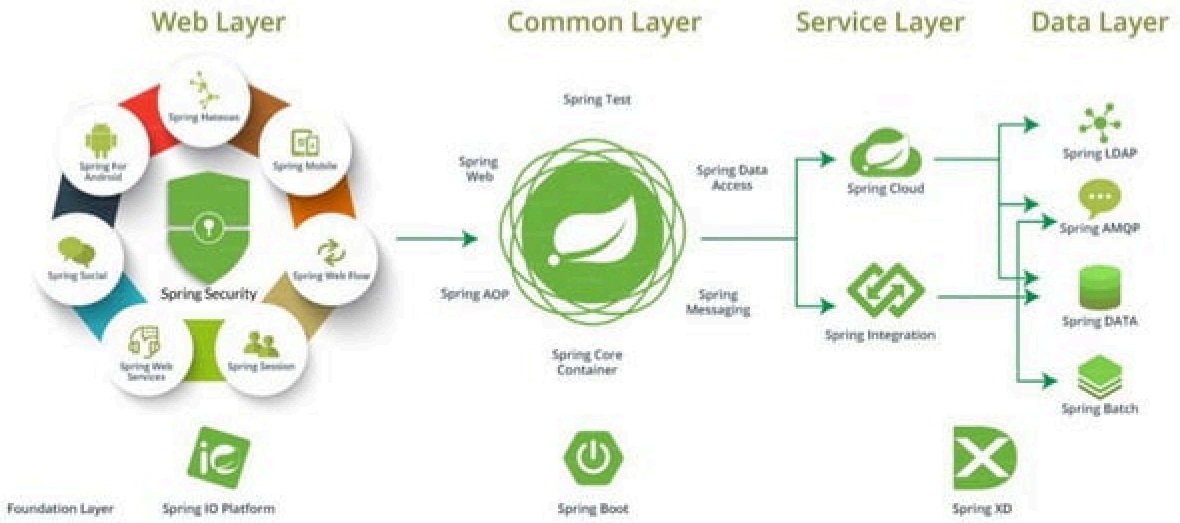
```
class Rider
{
   Bike b;
   public void setBike(Bike b)
   {
    this.b = b;
   }

   void ride()
   {
     b.start();
   }
}
```

```
interface Bike
{
  void start();
}
```

Spring container will inject either Honda object or Bajaj object or Yamaha object into the Rider class by calling setter method

Spring Framework Ecosystem



Web Layer

Common Layer

Service Layer

Data Layer

Spring Hateoas

Spring For Android

Spring Mobile

Spring Security

Spring Social

Spring Web Flow

Spring Web Services

Spring Session

Spring Test

Spring Web

Spring Data Access

Spring AOP

Spring Messaging

Spring Core Container

Spring Cloud

Spring Integration

Spring LDAP

Spring AMQP

Spring DATA

Spring Batch

Foundation Layer

Spring IO Platform

Spring Boot

Spring XD

Summary