THE SPRING FRAMEWORK BEGINNER'S GUIDE: FEATURES, ARCHITECTURE & GETTING STARTED



Java is one of the most popular and widely used programming languages, powering everything from mobile applications to large-scale enterprise systems. Since Java can power almost any kind of development, numerous frameworks and tools have evolved to simplify and streamline Java-based software developments.

When it comes to building Java-based web applications, Spring is one of the most popular frameworks. In this article, we will look at the Spring framework and how to build Java applications using it.

What is the Spring Framework?

The Spring Framework is an open-source framework for building enterprise Java applications. Spring aims to simplify the complex and cumbersome enterprise Java application development process by offering a framework that includes technologies such as:

- Aspect-oriented programming (AOP)
- Dependency injection (DI)
- Plain Old Java Object (POJO)

Even with all these technologies, Spring is a lightweight framework that can be used to create scalable, secure, and robust enterprise web applications.

At a macro-level, we can consider the Spring framework a collection of sub frameworks such as Spring Web Flow, Spring MVC, and Spring ORM. In addition to Java, Spring also supports Kotlin and

Groovy.

The Spring framework is also the base that powers all the other Spring-based projects, such as:

- Spring Boot
- Spring Cloud
- Spring GraphQL

Core Features of Spring Framework

Let's look at the core features of the Spring framework.

IoC (Inversion of Control) Container

IoC container is one of the core features of Spring that provides a streamlined way to configure and manage Java objects. This container is responsible for managing the lifecycle of a defined Java object, significantly increasing the configurability of a Spring-based application.

IoC uses the dependency injection or dependency lookup patterns to provide the object reference during runtime. The container consists of assembler code that is required for configuration management.

Spring provides org.springframework.beans and org.springframework.context packages that can be used to facilitate these functions.

Support for aspect-oriented programing

AOP aims to provide more modularity to the cross-cutting concerns, which are functions that span across the application, such as:

- Logging
- Caching
- Transaction management
- Authentication
- Etc.

Moreover, AOP complements object-oriented programming by providing a different way to structure the program, where OOP modularity is based on classes.

In AOP, the main unit of modularity is an aspect (cross-cutting concern). This enables users to use AOP to create custom aspects and declarative enterprise services. The IoC container does not depend on AOP, offering further freedom for developers to select their preferred programming method.

However, Aspect-Oriented Programming combined with the Spring IoC provides a robust middleware solution.

Data access framework

Database communication issues are one of the common issues developers face when developing applications. Spring simplifies the database communication process by providing direct support for

popular data access frameworks in Java, such as JDBC, Hibernate, Java Persistence API (JPA), etc.

Additionally, it offers features such as resource management, exception handling, and resource wrapping for all the supported data access frameworks, further simplifying the development process.

Transaction management framework

Unlike the Java Transaction API (JTA), the Spring Transaction Management Framework is not limited to global and nested transactions. Spring offers an abstraction mechanism for Java that enables users to:

- Work with local, global, and nested transactions
- Save points
- Simplify transaction management across the application

The Spring Data Access Framework directly integrates with the Transaction Management Framework with support for messaging and caching. This enables developers to create feature-rich transactional systems that span across the applications without depending on EJB or JTA.

Spring MVC framework

The Spring MVC enables developers to create applications using the popular MVC pattern. It is a request-based framework that allows developers to easily create customized MVC implementations that exactly suit their needs.

The core component of Spring MVC is the DispatcherServlet class which handles user requests and then forwards them to the correct controller. This allows the controller to process the request, create the model and then provide the information to the end-user via a specified view.

Spring web service

This Spring Web Service component provides a streamlined way to create and manage web service endpoints in the application. It offers a layered approach that can be managed using XML and can be used to provide mapping for web requests to a specific object.

Spring test frameworks

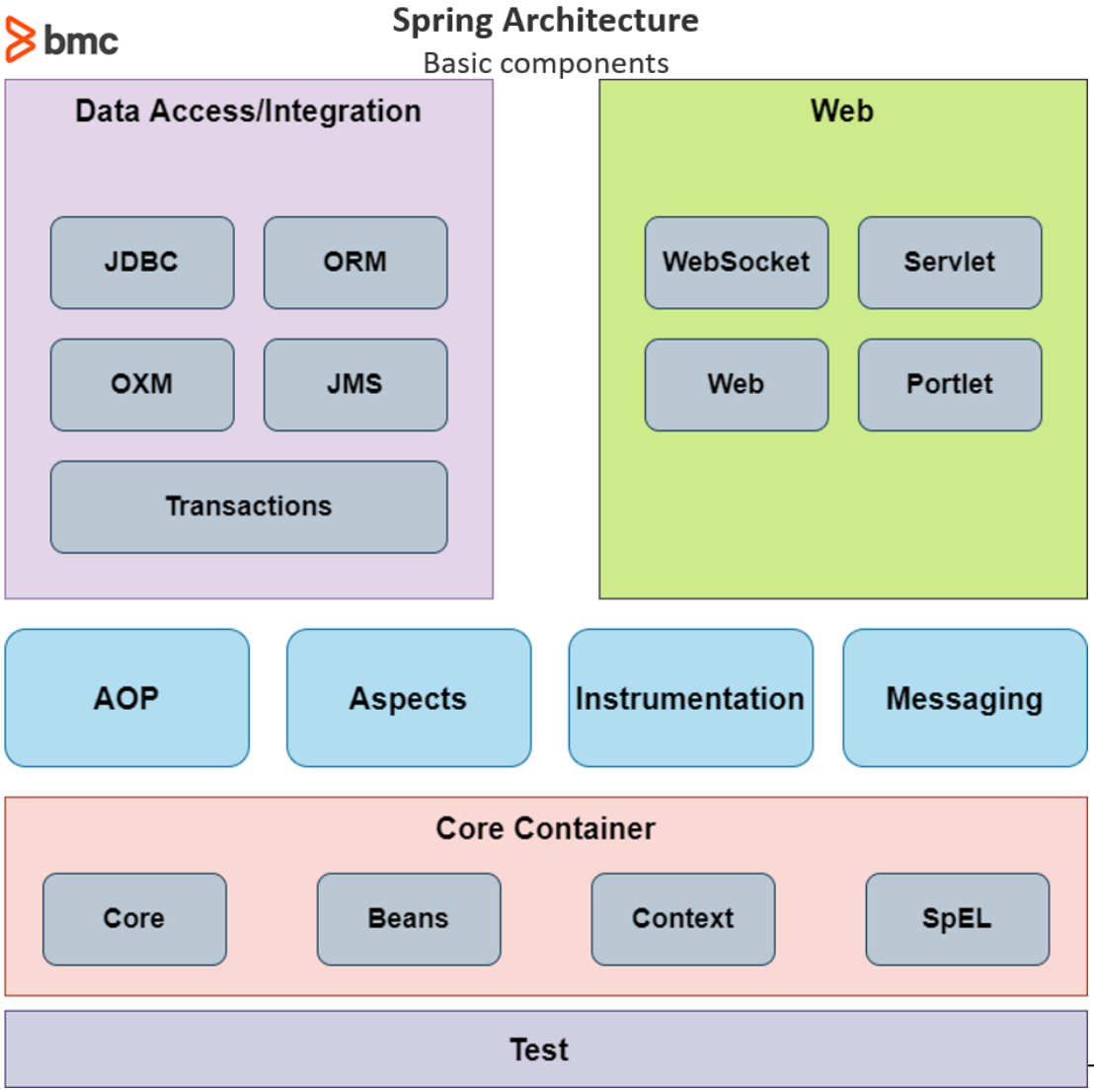
Testing is a core part of any development. Spring simplifies testing within the framework with components like:

- Mock objects
- TestContext framework
- Spring MVC Test
- Etc.

(Read about test automation frameworks.)

Spring Framework Architecture

OK, we've looked at the core features that makes Spring an excellent framework. Now we'll look into the architecture of Spring—the architecture that facilitates all these features.



The above diagram represents the basic components of the Spring architecture. As you can see, Spring is built using different modules that enable different functionality.

Core container

This contains the fundamental modules that are the cornerstone of the Spring framework.

- **Core (spring-core)** is the core of the framework that power features such as Inversion of Control and dependency injection.
- **Beans (spring-beans)** provides Beanfactory, which is a sophisticated implementation of the factory pattern.
- **Context (spring-context)** builds on Core and Beans and provides a medium to access defined objects. ApplicationContext interface is the core part of the Context module, and the spring-context-support provides support for third-party interactions such as caching, mailing, and template engines.
- **SpEL (spring-expression)** enables users to use the Spring Expression Language to query and manipulate the object graph at runtime.

Data access/integration

This includes the modules that are used to handle data access and transaction processing in an application.

- **JDBC (spring-jdbc)** provides a JDBC abstraction layer that eliminates the need to separate JDBC coding when dealing with databases.
- **ORM (spring-orm)** are integration layers for popular object-relational mapping API such as JPA, JDO Hibernate.
- **OXM (spring-oxm)** is the abstraction layer that supports Object/XML mapping implementations like JAXB, XStream.
- **JMS (spring-jms)** is the Java Messaging Service module that creates and consumes messages that directly integrate with the Spring messaging module.
- **Transaction (spring-tx)** offers programmatic and declarative transaction management for classes that include special interfaces and POJOs.

Web

The Web layer relates to modules that power web-based functions in Spring.

- **WebSocket (spring-websocket)** powers the web socket-based communication for clients and servers.
- **Servlet (spring-webmvc)** is the Spring WebMVC module that contains the MVC and REST implementations.
- **Web (spring-web)** provides all the basic web-oriented features and contains an HTTP client and web-related parts of the Spring remoting.
- **Portlet (spring-webmvc-portlet)** provides the MVC implementation to be used in a portlet environment.

Other Modules

- **AOP (spring-aop)** provides an aspect-oriented programming implementation that can be used when creating applications.
- **Aspects (spring-aspects)** enables direct integration with the AspectJ programming extension by the eclipse foundation.
- **Instrumentation (spring-instrument)** is the class instrumentation support and class loader implementations for application servers.
- **Messaging (spring-messaging)** provides a robust platform to manage messaging in applications.
- **Test (spring-test)** is the Spring test module that supports unit and integration testing with JUnit and TestNG.

Getting started with the Spring framework

Now that we understand what makes Spring a truly great framework, let's see how to start developing a Spring project.

We will be using a Windows environment and Visual Studio Code as our IDE for the following example. First, we need Java installed on our PC. You can get the latest prebuilt open-source Java

(OpenJDK) binaries from the Adoptium project. We will be using the latest LTS release at the time of writing this article which is OpenJDK 17 (LTS).

```
> java --version
openjdk 17 2021-09-14
OpenJDK Runtime Environment Temurin-17+35 (build 17+35)
OpenJDK 64-Bit Server VM Temurin-17+35 (build 17+35, mixed mode, sharing)
```

While you can always create a Spring project manually, the spring initializer (https://start.spring.io/) provides users with an easy way to configure the project using spring boot. Spring Boot is an extension of the Spring framework that enables users to create standalone Spring-based applications with minimal dependencies.

The spring initializer provides a centralized platform to easily create a Spring project with different options, including project type (Maven/Gradle), language (Java, Kotlin, Groovy), Project metadata, Packaging, Java version, and dependencies.

We will create a simple Maven Project named ***test_app*** in ***com.myproject*** group using Java and Spring Boot 2.5.5 targeting Java 17 (OpenJDK) with Spring Web as the dependency.



The above action will generate a zip file. Extract that zip file and open the folder in the IDE. Since we are using Visual Studio Code as our IDE, make sure that you have installed the appropriate VSCode extensions to support Java and Spring. Following are the recommended extensions.

- Extension Pack for Java - Visual Studio Marketplace
- Spring Boot Extension Pack - Visual Studio Marketplace

Navigate to the primary java file (Example :<Name>\src\main\java\com\myproject\test_app\TestAppApplication.java) and change the code

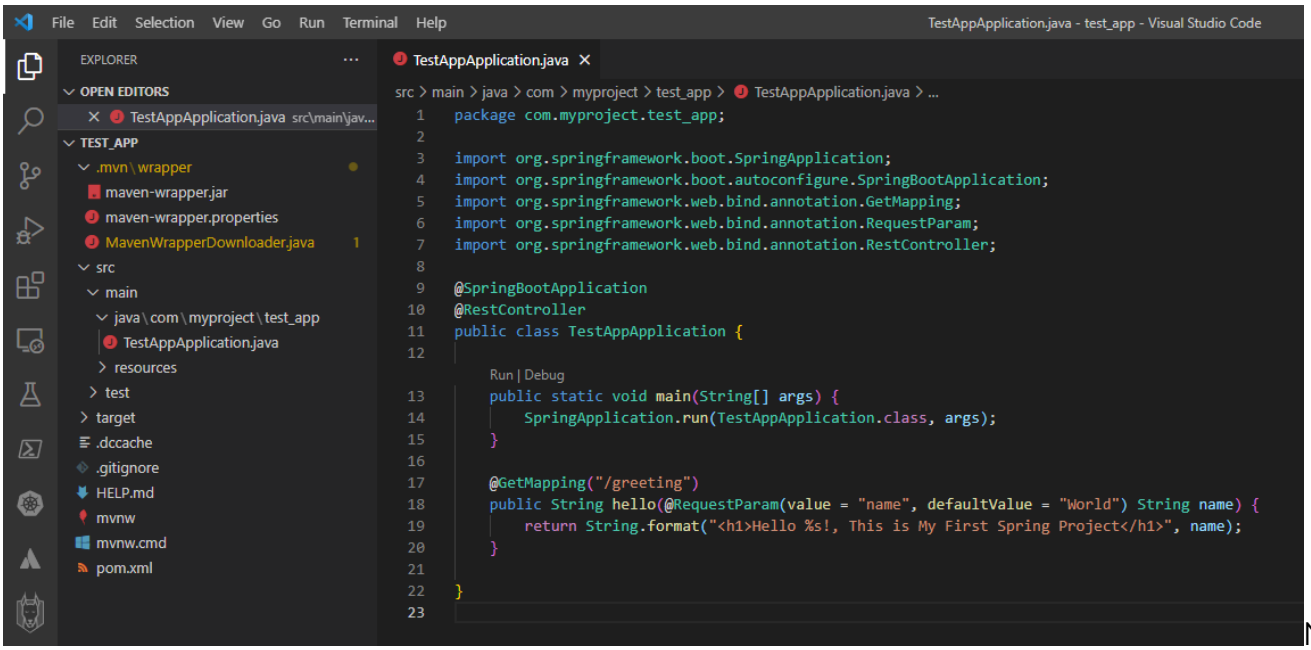
as follows. It will create a simple web server that returns a greeting when a user navigates to the greetings URL.

```
package com.myproject.test_app;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class TestAppApplication {
public static void main(String[] args) {
SpringApplication.run(TestAppApplication.class, args);
}
@GetMapping("/greeting")
public String hello(@RequestParam(value = "name", defaultValue = "World")
String name) {
return String.format("<h1>Hello %s!, This is My First Spring Project</h1>",
name);
}
}
```

VS Code view:



Now we can build and execute the project by opening the terminal in the root of the project folder and running the following command.

```
/mvnw spring-boot:run
```

Result:

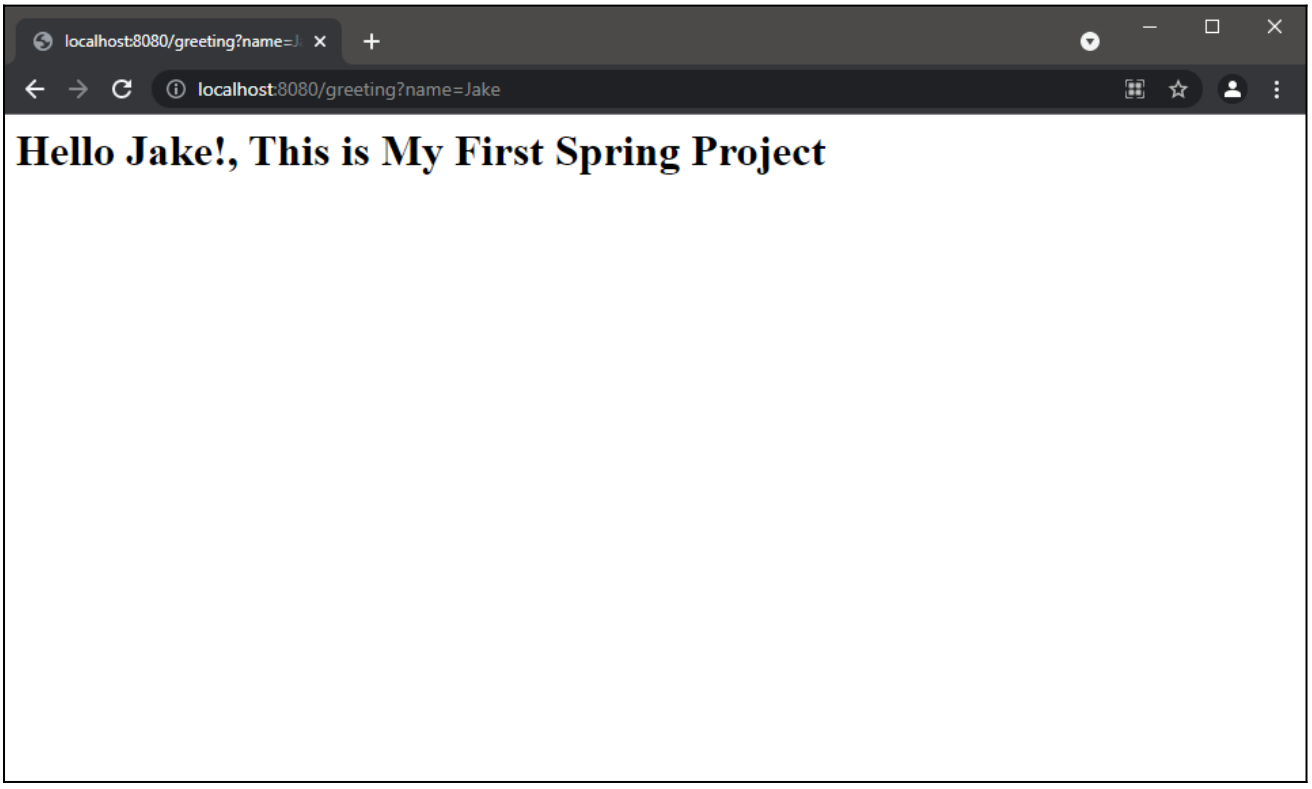
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.6)

2021-09-30 05:40:06.513  INFO 26928 --- [           main] c.myproject.test_app.TestAppApplication  : Starting TestAppApp
lication using Java 17 on BISI-PC01M with PID 26928 (F:\Bisina\Downloads\test_app\target\classes started by bisin in F:\
Bisina\Downloads\test_app)
2021-09-30 05:40:06.515  INFO 26928 --- [           main] c.myproject.test_app.TestAppApplication  : No active profile s
et, falling back to default profiles: default
2021-09-30 05:40:07.070  INFO 26928 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized
with port(s): 8080 (http)
2021-09-30 05:40:07.078  INFO 26928 --- [           main] o.apache.catalina.core.StandardService   : Starting service [T
omcat]
2021-09-30 05:40:07.079  INFO 26928 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet en
gine: [Apache Tomcat/9.0.53]
2021-09-30 05:40:07.126  INFO 26928 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
 embedded WebApplicationContext
2021-09-30 05:40:07.126  INFO 26928 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplication
Context: initialization completed in 578 ms
2021-09-30 05:40:07.342  INFO 26928 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on p
ort(s): 8080 (http) with context path ''
2021-09-30 05:40:07.350  INFO 26928 --- [           main] c.myproject.test_app.TestAppApplication  : Started TestAppAppl
ication in 1.086 seconds (JVM running for 1.307)
```

It will start up the web server, and we can navigate to the greetings page using the following URL format.

`http://localhost:8080/greeting?name=<Value>`

For example, if we navigate to http://localhost:8080/greeting?name=Jake URL, it will provide a greeting for Jake.



Spring can power enterprise Java apps

Spring is a powerful Java framework that can be used to power enterprise-grade Java applications at any scale. Even though it can sometimes become complex and daunting, you can easily get the

hang of this framework and build the next great software if you start from the ground up.

Spring Framework, alongside all the other Spring Projects, provides all the necessary tools and features to cater to any user requirement while simplifying the development process.

Related reading

- BMC DevOps Blog
- Python vs Java: What's The Difference?
- Java Developer Roles & Responsibilities
- React JavaScript Library: Concepts & Tutorials for Getting Started
- GitHub, GitLab, Bitbucket & Azure DevOps: What's The Difference?