Introduction to Spring Boot

Last Updated : 28 Aug, 2025

Spring is one of the most popular frameworks for building enterprise applications, but traditional Spring projects require heavy XML configuration, making them complex for beginners.

Spring Boot solves this problem by providing a ready-to-use, production-grade framework on top of Spring. It eliminates boilerplate configuration, comes with an embedded server and focuses on rapid development.

Features of Spring Boot

Spring Boot is built on top of the conventional Spring framework, providing all the features of Spring while being significantly easier to use. Here are its key features:

1. Auto-Configuration: Automatically configures components (like Hibernate, JPA) based on dependencies. No need for manual XML setup.

2. Easy Maintenance and Creation of REST Endpoints: With annotations like @RestController, @GetMapping and @PostMapping, creating REST endpoints is straightforward.

Example:

```
@RestController
@RequestMapping("/api")
public class MyController {
    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```

3. Embedded Tomcat Server: Embedded Servers: Spring Boot includes an embedded Tomcat server, eliminating the need for manual setup. It also supports Jetty and Undertow, offering flexibility for different application needs.

4. Easy Deployment: Spring Boot applications can be packaged as JAR or WAR files and deployed directly to servers or cloud environments. By 2025, it offers seamless integration with Docker and Kubernetes for easier cloud-native deployment and scaling.

5. Microservice-Based Architecture: Spring Boot is well-suited for microservice architecture, where applications are split into independent, modular services. Unlike monolithic systems, this approach improves scalability, maintainability and deployment flexibility.

Evolution of Spring Boot

Spring Boot was introduced in 2013 following a JIRA request by Mike Youngstrom to simplify Spring bootstrapping.

Major Spring Boot Versions

- Spring Boot 1.0 (April 2014)
- Spring Boot 2.0 (March 2018)
- Spring Boot 3.0 (November 2022) (Introduced Jakarta EE 9+, GraalVM support)
- Latest Version (2025): Spring Boot 3.x (Enhancements in observability, native images and containerization)

Applications of Spring Boot

Spring Boot is widely used in modern software development because of its simplicity, scalability and production-ready features. Here are some of its major applications:

1. Enterprise Applications: Build complex applications such as Hospital Management Systems, Banking Systems or ERP solutions with minimal

configuration.

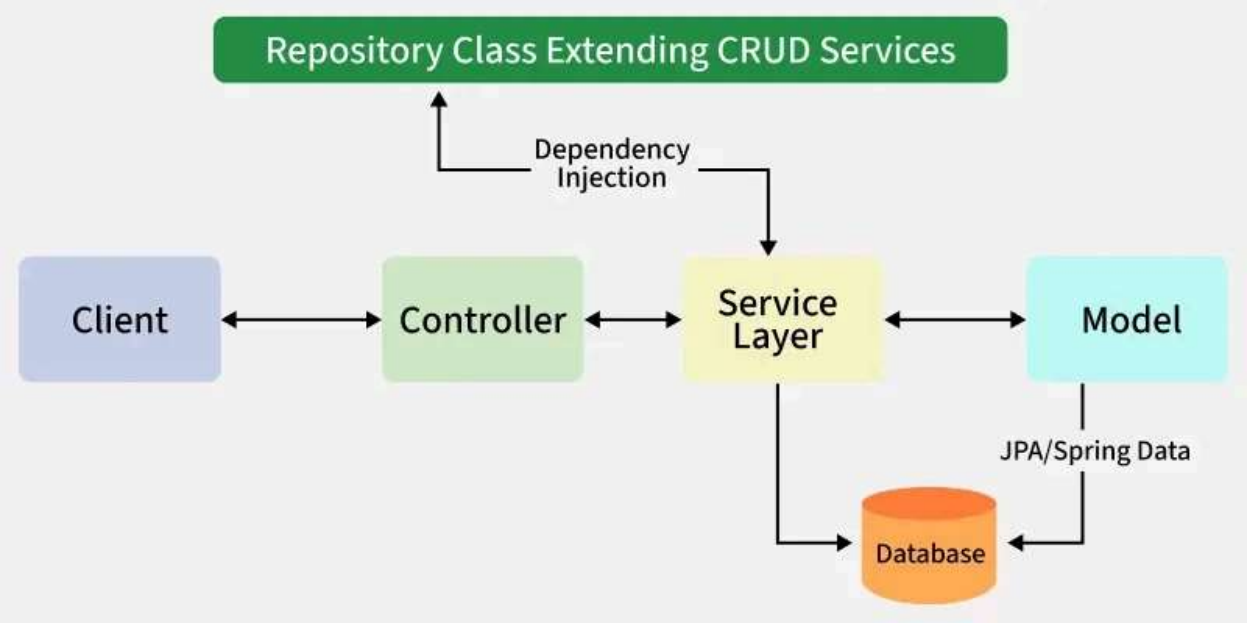
2. Cloud-Native Applications: Seamless integration with Docker, Kubernetes and cloud platforms (AWS, Azure, GCP) for deployment and scaling.

3. Real-Time Applications: Supports Reactive Programming for chat applications, streaming platforms, IoT systems and event-driven systems.

4. Batch Processing Applications: With Spring Batch, Spring Boot is used for ETL (Extract, Transform, Load), report generation and large data processing.

Spring Boot Architecture

To understand the architecture of Spring Boot, let's examine its different layers and components.



Spring Boot Flow Architecture

1. Client Layer

- This represents the external system or user that interacts with the application by sending HTTPS requests.

2. Controller Layer (Presentation Layer)

- Handles incoming HTTP requests from the client.
- Processes the request and sends a response.
- Delegates business logic processing to the Service Layer.

3. Service Layer (Business Logic Layer)

- Contains business logic and service classes.
- Communicates with the Repository Layer to fetch or update data.
- Uses Dependency Injection to get required repository services.

4. Repository Layer (Data Access Layer)

- Handles CRUD (Create, Read, Update, Delete) operations on the database.
- Extends Spring Data JPA or other persistence mechanisms.

5. Model Layer (Entity Layer)

- Represents database entities and domain models.
- Maps to tables in the database using JPA/Spring Data.

6. Database Layer

- The actual database that stores application data.
- Spring Boot interacts with it through JPA/Spring Data.

Request Flow in Spring Boot

Client ->Controller ->Service ->Repository ->Database ->Response

- A Client makes an HTTPS request (GET/POST/PUT/DELETE).
- The request is handled by the Controller, which is mapped to the corresponding route.
- If business logic is required, the Controller calls the Service Layer.

- The Service Layer processes the logic and interacts with the Repository Layer to retrieve or modify data in the Database.
- The data is mapped using JPA with the corresponding Model/Entity class.
- The response is sent back to the client. If using Spring MVC with JSP, a JSP page may be returned as the response if no errors occur.

Comment More info Advertise with us

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

GET IT ON Google Play Download on the App Store

Advertise with us

Company
About Us
Legal
Privacy Policy
Careers
Contact Us
Corporate Solution

Explore
POTD
Job-A-Thon
Connect
Community
Videos
Blogs