Introduction to Spring Boot

Spring Boot is an enterprise-grade Java framework launched in 2014 by Pivotal (now VMware Tanzu).

Currently at version 3.2.2 (as of Q1 2025), it's used by 60% of Java developers worldwide.

by Saratha Natarajan

The Problem Spring Boot Solves

Traditional Spring

Required extensive XML/Java configuration files

Configuration Hell

Multiple dependencies caused setup nightmares

Slow Setup Time

Average Spring project setup took 2-3 days

Spring Boot Solution

Reduces setup to minutes with auto-configuration

Core Spring Boot Features

Opinionated Defaults

"Convention over configuration" approach simplifies development

Embedded Servers

Includes Tomcat, Jetty, or Undertow with zero configuration

Standalone Applications

Run as self-contained units with minimal setup required

Production-Ready

Enterprise features available out of the box

Spring Boot vs Traditional Spring Framework

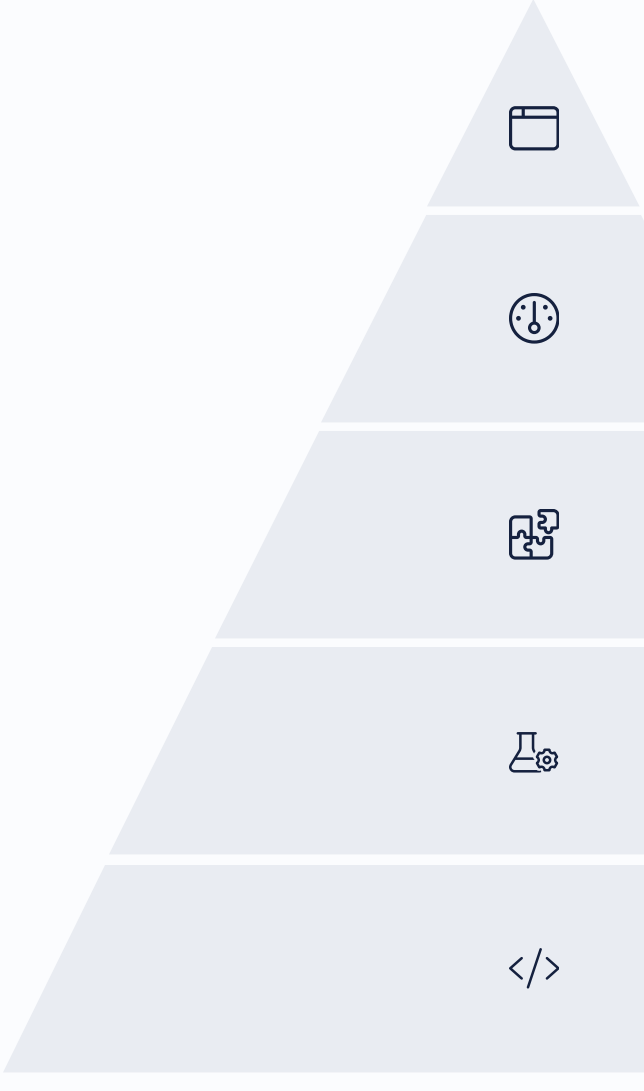
Traditional Spring

- Manual configuration required
- Extensive boilerplate code
- External server deployment
- Complex dependency management

Spring Boot

- Auto-configuration
- 75% less boilerplate code
- Embedded server included
- Starters reduce dependency management by 80%

Spring Boot Architecture



Your Application

Business logic and custom code

Spring Boot Actuator

Monitoring and metrics

Spring Boot Starters

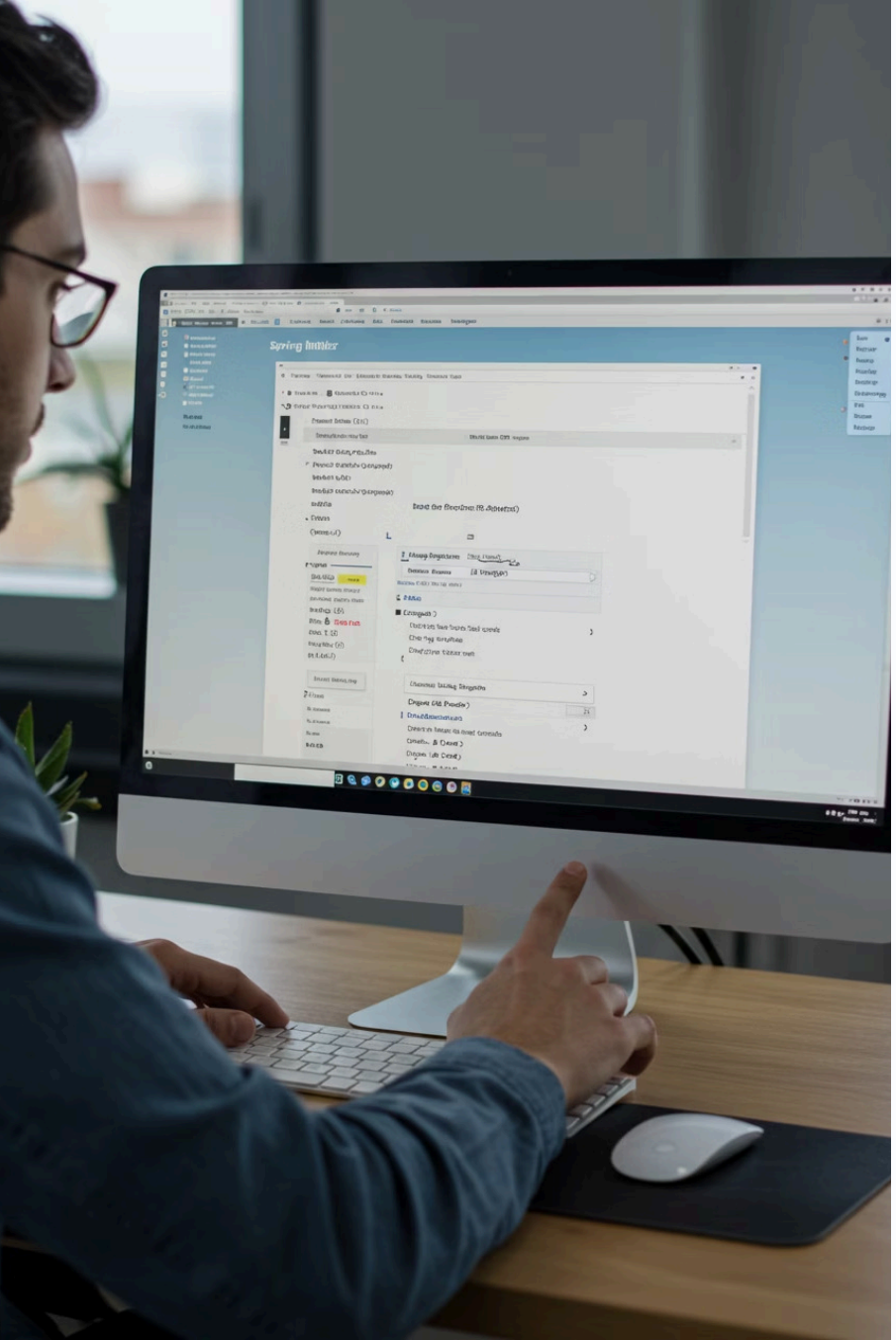
Curated dependency collections

Autoconfigure Module

Automatic configuration system

Core Module

Foundation of Spring Boot



Getting Started with Spring Boot

Spring Initializr

Visit start.spring.io to generate your project structure

Over 10 million projects generated annually

Configure Dependencies

Select required starters based on your project needs

Import into IDE

Use Spring Tools or your preferred IDE integration

Start Coding

Begin with minimal configuration using Maven or Gradle

Spring Boot Starters

spring-boot-starter-web

For web applications with Tomcat and Spring MVC

spring-boot-starter-data-jpa

For database access with JPA and Hibernate

spring-boot-starter-security

For authentication and authorization

spring-boot-starter-test

For comprehensive testing support

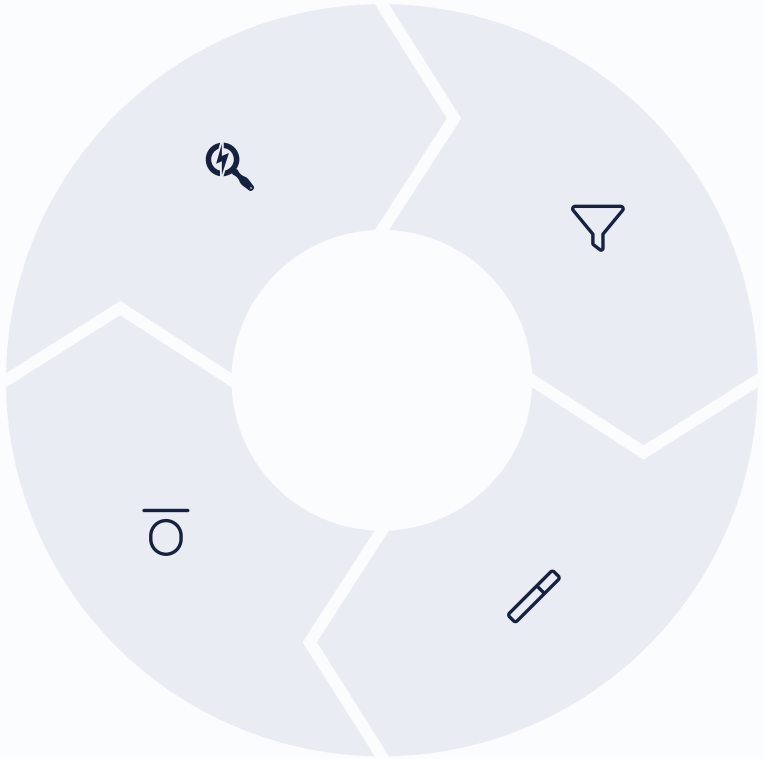
Auto-Configuration Magic

Classpath Scanning

Detects libraries on classpath

Conditional Evaluation

Analyzes using @ConditionalOn annotations

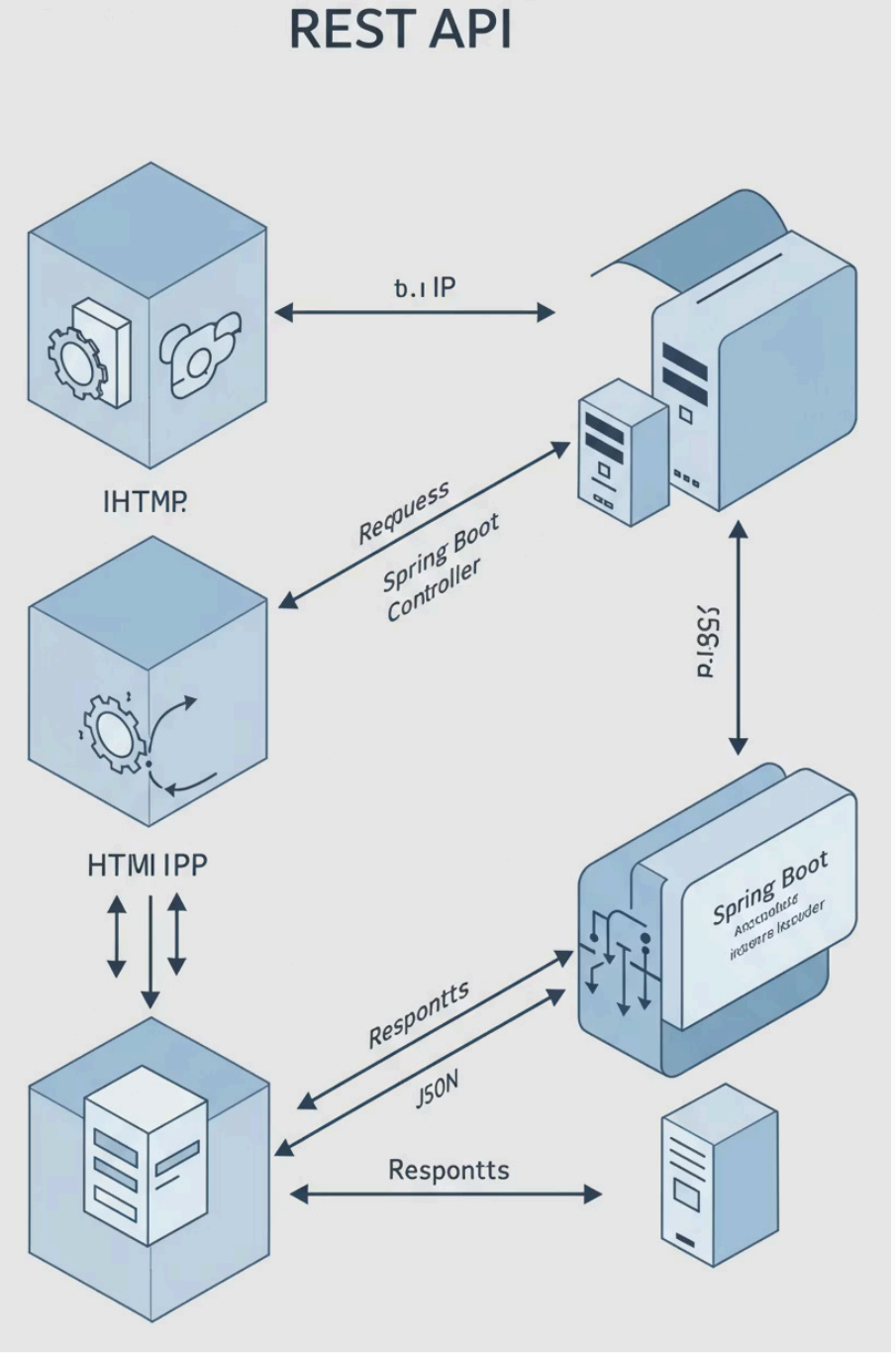


Custom Override

Default settings can be replaced when needed

Auto-Configuration

Activates appropriate beans automatically



Creating RESTful APIs with Spring Boot

@RestController

Simplifies creating endpoints that return JSON/XML responses

Request Mapping

@GetMapping, @PostMapping for HTTP method handling

Serialization

Built-in conversion between Java objects and HTTP responses

Validation

Bean Validation API ensures data integrity

Data Access with Spring Boot

Define Entity Classes

Create POJOs with JPA annotations

Create Repositories

Extend JpaRepository interface

Let Spring Generate Implementation

Spring Data creates queries automatically

4

Use Repository in Services

Inject repositories to access data

Spring Boot Security

Basic Security

Authentication in 10 lines of code

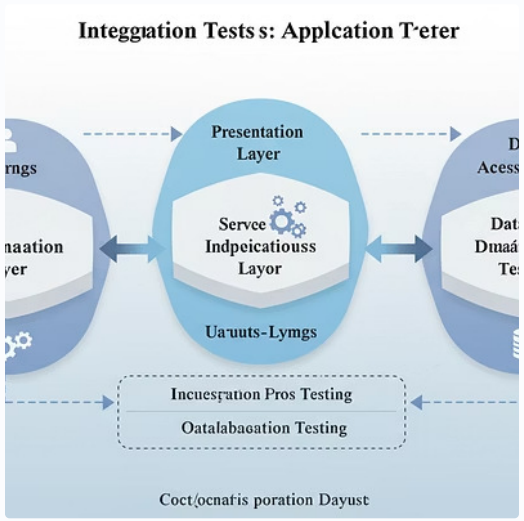
Advanced Features

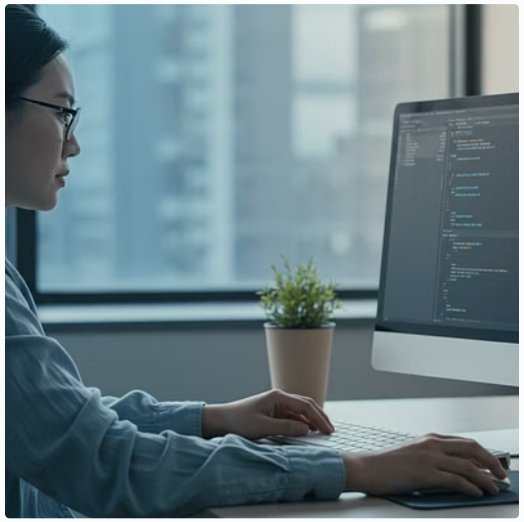
OAuth2 and OpenID Connect support

Enterprise Grade

Used by 74% of Fortune 500 companies

Testing in Spring Boot





@SpringBootTest

Loads the full application context for integration testing

@WebMvcTest

Tests only the controller layer for faster execution

@DataJpaTest

Tests repository components in isolation

MockMvc

Simulates HTTP requests without a running server

Spring Boot Actuator

15+

Built-in Endpoints

Ready-made monitoring points

1

Dependency

Single addition to enable all features

80%

Production Usage

Percentage of apps using Actuator

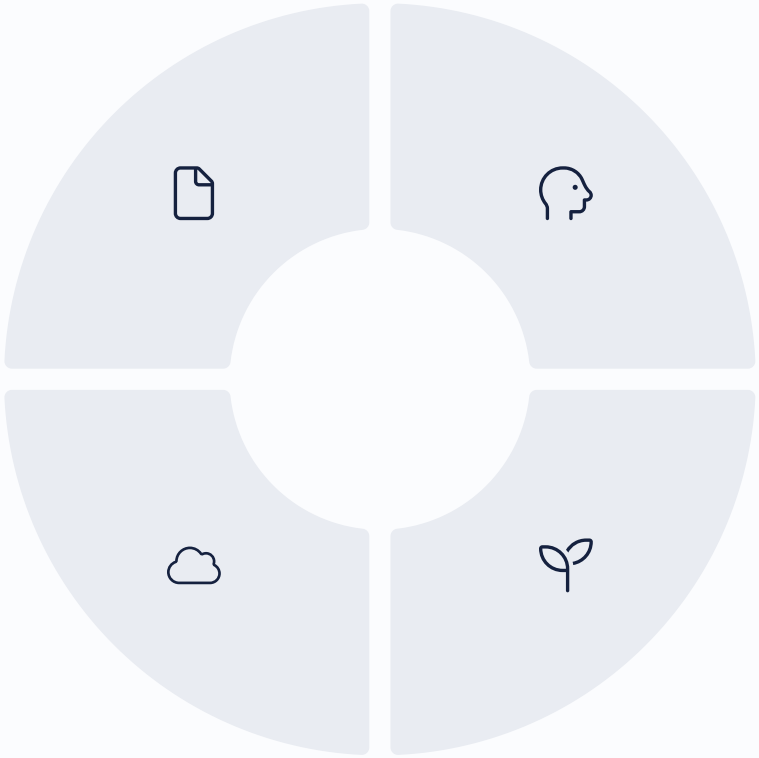
Configuration Management

Properties Files

application.properties/yml for basic settings

Profiles

Environment-specific configs (dev, test, prod)



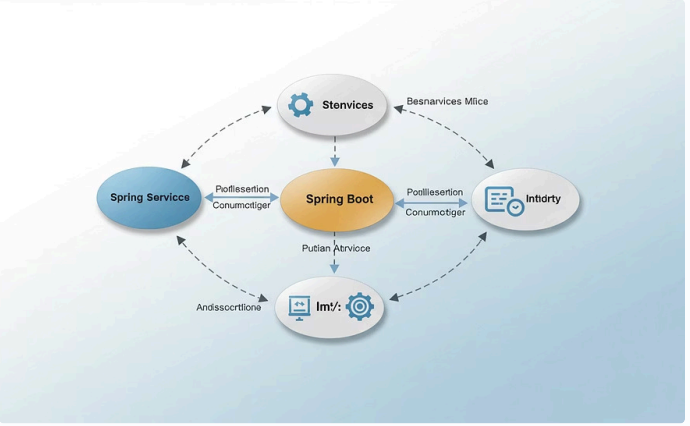
Cloud Config

Centralized configuration server integration

External Config

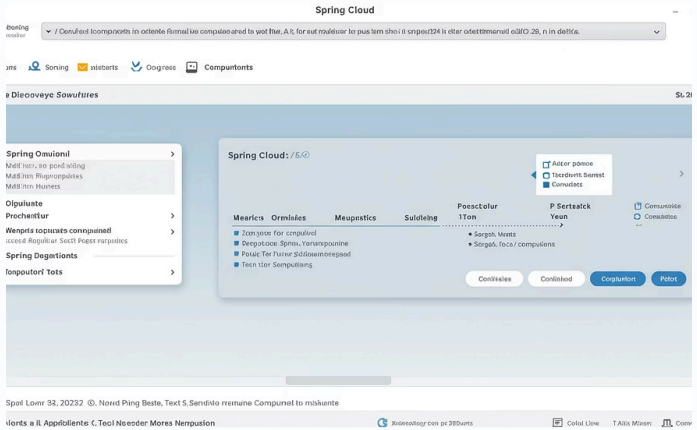
Override via environment variables

Microservices with Spring Boot



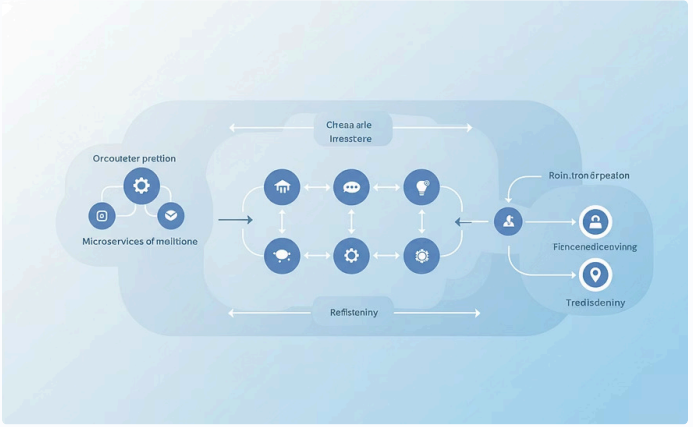
Lightweight Services

Typical size: 10-15MB, perfect for containerization



Service Discovery

Spring Cloud integration for service registration



Resilience Patterns

Circuit breakers protect from cascading failures

Deployment Options for Spring Boot

Executable JAR

Self-contained deployment with embedded server (30% faster)

Traditional WAR

Deploy to existing application servers when required

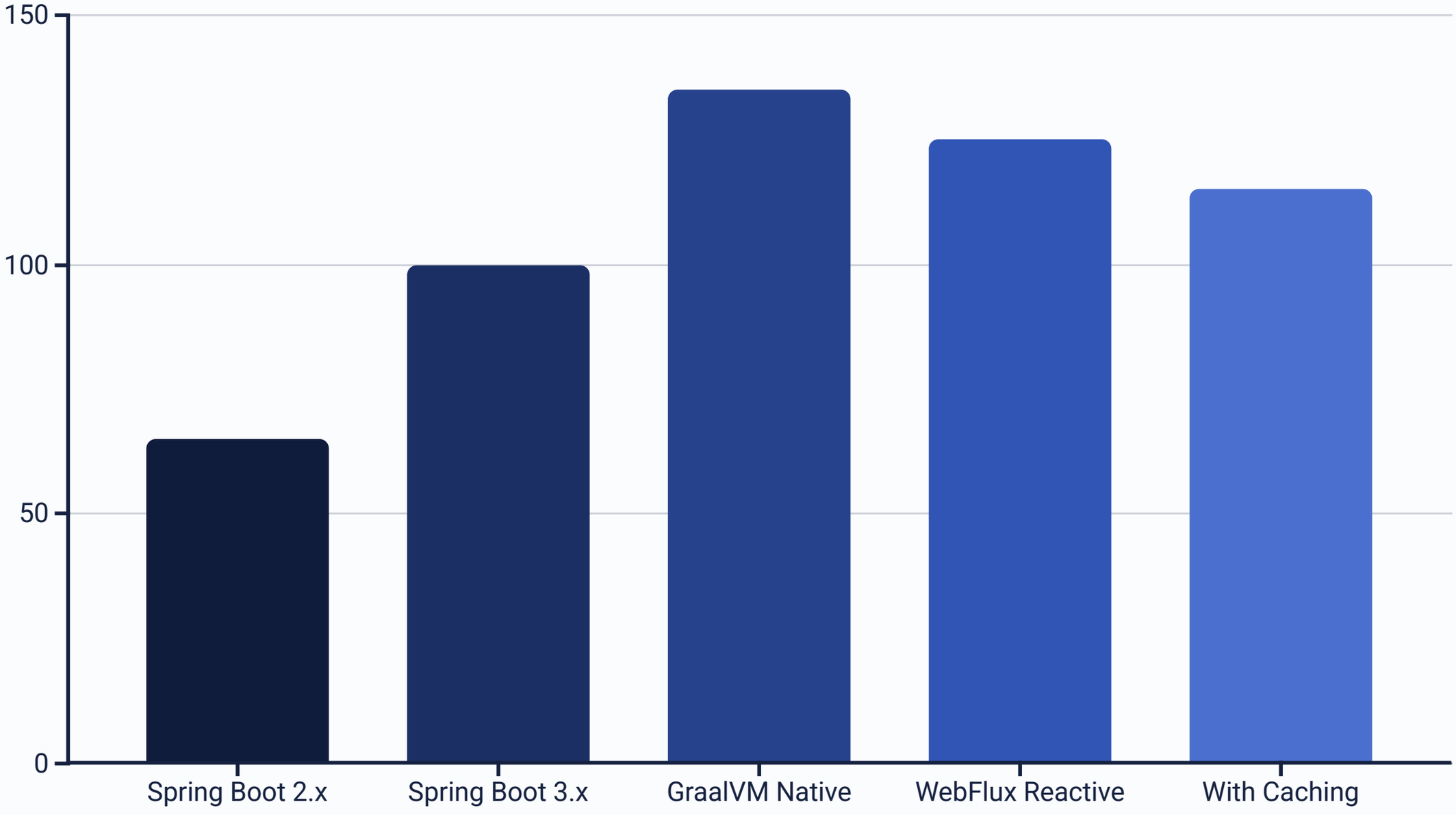
Docker Containers

75% of production deployments use containerization

Cloud Platforms

Seamless deployment to AWS, Azure, GCP

Performance Optimization



Real-World Spring Boot Use Cases

Netflix

90% of microservices built on Spring Boot

Alibaba

Processes 325,000 orders/second with Spring Boot

Financial Services

80% of new applications use Spring Boot

Healthcare

Patient management systems with 99.99% uptime

Spring Boot Roadmap (2025-2026)

1

AOT Compilation

Improvements for faster application startup

2

Enhanced Observability

OpenTelemetry integration for better monitoring

3

Native Image Support

40% size reduction for GraalVM native images

4

Reactive Programming

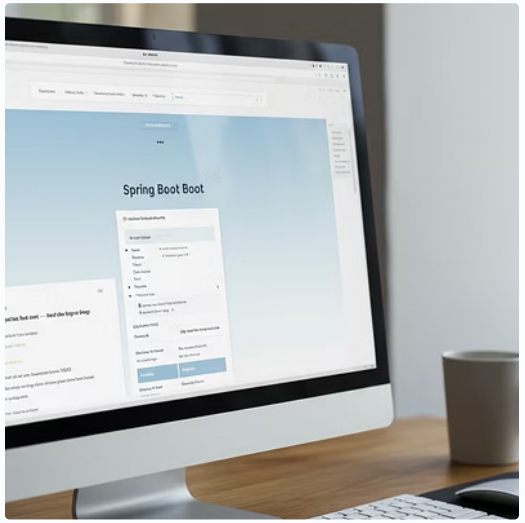
Simplified model for async applications

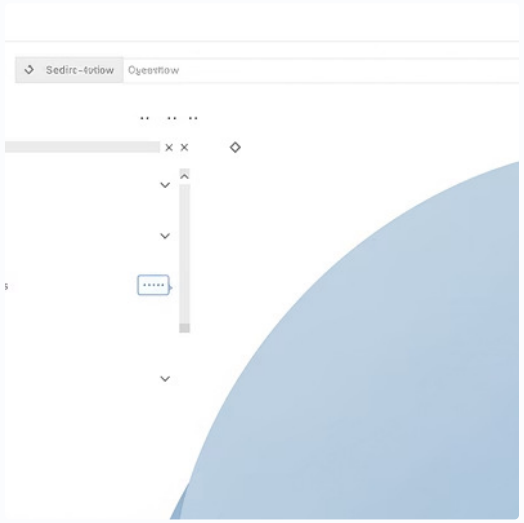
5

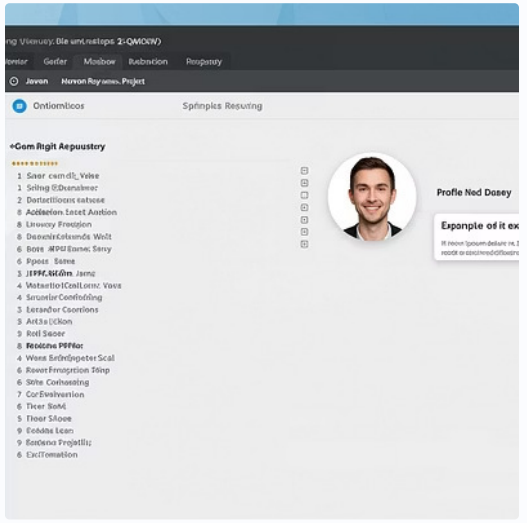
Cloud-Native

Expanded features for modern deployments

Getting Started & Resources







Visit the official docs at docs.spring.io/spring-boot and explore 200+ courses on major platforms.

Find answers in 1M+ Stack Overflow questions and 2.5K example applications on GitHub.