Spring Boot

Documentation

DATE – 28-02-2024

HIMANSHU KAKKAR

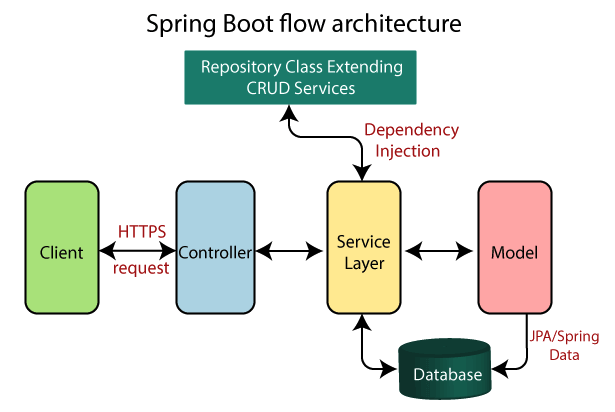
What is SpringBoot ?

**Spring Boot** is an open-source Java framework used to create a Micro Service. Spring boot is developed by Pivotal Team, and it provides a faster way to set up and an easier, configure, and run both simple and web-based applications. It is a combination of Spring Framework and Embedded Servers. The main goal of Spring Boot is to reduce development, unit test, and integration test time and in Spring Boot, there is no requirement for XML configuration.

The Spring Boot documentation provides extensive details on various aspects of developing applications with Spring Boot. Here's a breakdown of what you can find in the documentation:

1. **Getting Started**: The documentation starts with a quick overview of Spring Boot and provides instructions on how to set up your development environment and create your first Spring Boot application. It covers topics such as building with Maven or Gradle, running your application, and basic configuration.
2. **Features and Capabilities**: The documentation dives into the features and capabilities provided by Spring Boot, including auto-configuration, dependency management, and production-ready features like metrics, health checks, and externalized configuration.
3. **Spring Boot Fundamentals**: It covers fundamental concepts of Spring Boot such as starters, which are pre-configured dependencies for common use cases like web applications, data access, security, and messaging. It also explains how Spring Boot simplifies dependency injection, application context setup, and property binding.
4. **Configuration**: Spring Boot allows you to configure your application using properties, YAML files, environment variables, or Java configuration classes. The documentation explains how to configure various aspects of your application, including logging, database connections, caching, and profiles.
5. **Testing**: Spring Boot provides support for testing your applications, including unit tests, integration tests, and end-to-end tests. The documentation explains how to write tests using Spring Boot's testing utilities and how to configure and run tests with tools like JUnit and Mockito.
6. **Building APIs**: If you're building RESTful APIs with Spring Boot, the documentation covers topics such as creating controllers, handling requests and responses, validation, error handling, and documentation using tools like Swagger.
7. **Data Access**: Spring Boot provides support for accessing relational databases, NoSQL databases, and other data sources. The documentation explains how to configure data sources, create repositories with Spring Data, perform CRUD operations, and handle transactions.
8. **Security**: Spring Boot offers robust security features for protecting your applications against common security threats. The documentation covers topics such as authentication, authorization, securing endpoints, and protecting against common vulnerabilities like CSRF and XSS.
9. **Deployment**: Once you've built your application, you'll want to deploy it to production. The documentation provides guidance on packaging your application, configuring deployment environments, and monitoring and managing your application in production.
10. **Advanced Topics**: For more advanced use cases, the documentation covers topics such as reactive programming with Spring WebFlux, messaging with Spring Integration and Spring Cloud Stream, microservices with Spring Cloud, and more.
11. **Reference Documentation**: The documentation includes reference documentation for various Spring Boot features, including configuration properties, actuator endpoints, metrics, and more.

Overall, the Spring Boot documentation is a comprehensive resource for developers building applications with Spring Boot, providing detailed explanations, examples, and best practices for building robust, efficient, and maintainable applications.



Before getting to the Spring Boot Architecture, we must first understand what each of those layers and classes means. The four layers in Spring Boot are as follows:

* Presentation Layer
* Business Layer
* Persistence Layer
* Database Layer

The following information is provided about these layers:

* **Presentation Layer:** A JSON structure is made up of viewpoints. A front layer is what is seen from the outside and a back layer is where the structure is built from viewpoints. It interprets JSON and handles authentication and HTTP requests. After authentication, it enters the business layer for further processing.
* **Business Layer:**As part of business logic, it manages all of the business decisions and performs the business validation and consent. For example, only admins may modify the customer’s account.
* **Persistence Layer:**All of the storage logic in this program, including database questions, is accounted for in this portion of code. It also translates between database rows and company items.
* **Database Layer:**The structure is constructed in this way: The HTTP request or internet requests are handled by the Controllers from the demonstration layer, the providers control the company logic, as well as also the repositories preserve persistence (storage logic). Each provider, service, and repository may be controlled by a single Controller. There are many repositories, which may be managed by many repositories, and databases may be managed by many databases.

**KEY COMPONENTS OF SPRINGBOOT**

Spring Boot Framework has mainly four major Components.

* Spring Boot Starters
* Spring Boot AutoConfigurator
* Spring Boot CLI
* Spring Boot Actuator

SPRINGBOOT STARTER

Spring Boot Starters is one of the major key features or components of Spring Boot Framework. The main responsibility of Spring Boot Starter is to combine a group of common or related dependencies into single dependencies. We will explore this statement in detail with one example. For instance, we would like to develop a Spring WebApplication with Tomcat WebServer. Then we need to add the following minimal jar dependencies in your Maven’s pom.xml file or Gradle’s build.gradle file.

SPRINGBOOT AUTOCONFIGURATOR

Another important key component of Spring Boot Framework is Spring Boot AutoConfigurator. Most of the Spring IO Platform (Spring Framework) Critics opinion is that "To develop a Spring-based application requires lot of configuration (Either XML Configuration of Annotation Configuration). Then how to solve this problem. The solution to this problem is Spring Boot AutoConfigurator. The main responsibility of Spring Boot AutoConfigurator is to reduce the Spring Configuration. If we develop Spring applications in Spring Boot,then We dont need to define single XML configuration and almost no or minimal Annotation configuration. Spring Boot AutoConfigurator component will take care of providing those information.

SPRINGBOOT CLI

Spring Boot CLI(Command Line Interface) is a Spring Boot software to run and test Spring Boot applications from command prompt. When we run Spring Boot applications using CLI, then it internally uses Spring Boot Starter and Spring Boot AutoConfigurate components to resolve all dependencies and execute the application. We can run even Spring Web Applications with simple Spring Boot CLI Commands. Spring Boot CLI has introduced a new “spring” command to execute Groovy Scripts from command prompt. **spring command example:**

spring run HelloWorld.groovy

SPRINGBOOT ACTUATOR

Spring Boot Actuator components gives many features, but two major features are

* Providing Management EndPoints to Spring Boot Applications.
* Spring Boot Applications Metrics.

When we run our Spring Boot Web Application using CLI, Spring Boot Actuator automatically provides hostname as “localhost” and default port number as “8080”. We can access this application using “<https://localhost:8080/>” end point. We actually use HTTP Request methods like GET and POST to represent Management EndPoints using Spring Boot Actuator. We will discuss some more details about Spring Boot Actuator in coming posts.

THANK YOU