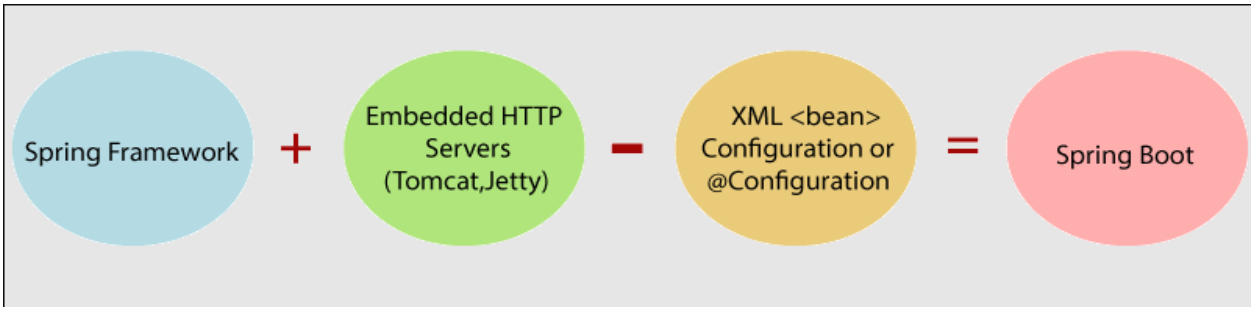
Spring Boot

Introduction

- It is an **open-source java framework** used to build applications.
- It provides an easier and faster way to set up, configure and run both simple and **web-based** applications.
- **Spring Boot** is a spring module that provides **rapid application development** feature to a spring framework.
- It is used to create a spring-based application that you can just run as it needs a minimal spring configuration.
- Spring Representation image:



- **Spring Boot** is a combination of **spring framework** and **embedded servers (http servers).**

Why Spring Boot

- The DI used in Spring Boot is the same as Spring.
- It contains powerful database transaction management capabilities.
- It simplifies integrating with other java frameworks like hibernate, JPA, structs, Filters, EJB, etc.
- It reduces the cost and development time of building a web application.

Features of Spring Boot

- **Auto-configuration:** we need not define any configurations.

- **stand-alone applications:** no need to follow any traditional approach like MVC.
- Embedded web server support.
- Spring Boot starter: starter is a collection of all dependencies.
- Spring Boot CLI.
- Spring Boot Actuator which is used to manage web applications.
- Microservices support.
- Embedded database support.
- Spring boot initializer.
- More secure compared to spring.
- Testing happens automatically with higher efficiency.

Advantages of Spring Boot

- It offers a CLI tool for developing and testing a spring boot application.
- It offers several plugins.
- There is no requirement of configuration file or xml file.
- It increases productivity and reduces the development time.

Spring Boot Versions

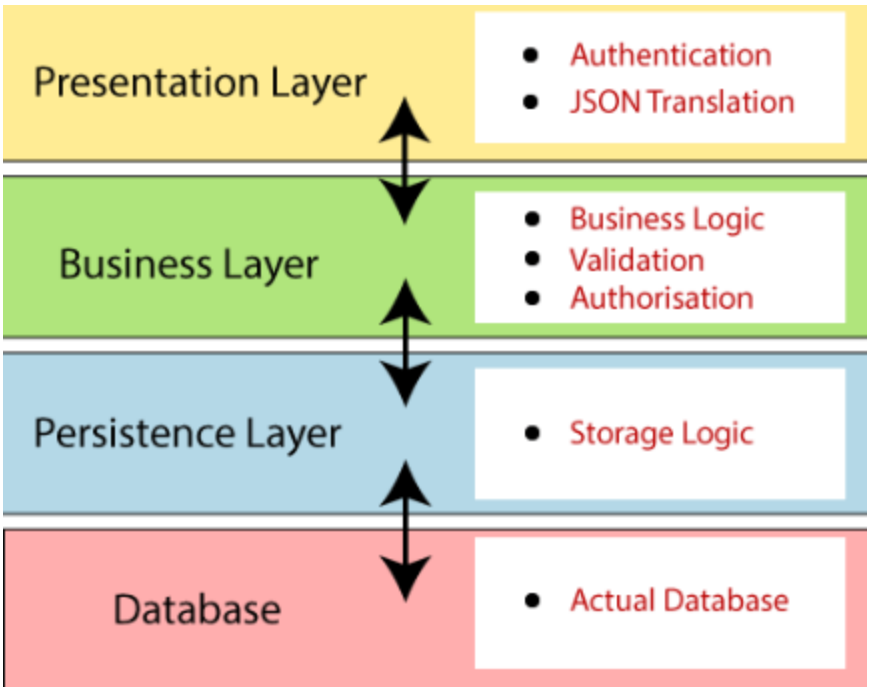
- Latest version of Spring Boot is 3.3 as of July 2024
- You can use the following link to get complete version history: Versions

Spring Boot Architecture

It follows a **layered architecture** in which each layer communicates with either the above layer or below layer.

The Spring Boot architecture consists of the following layers:

- Presentation Layer
- Business Layer
- Persistence Layer
- Database Layer



Presentation Layer

It handles the http requests from the users and transfers the same to Business Layer.

Business Layer

It handles all the Business logic of your application.

Persistence Layer

- It handles all the storage logic of your application.

Database Layer

- In this layer CRUD operations can be performed.

Requirements of Spring Boot

- Maven: Open-source project management tool
- Java with 17 or above
- Spring framework 5.o
- Spring Tool Suite (STS): IDE / Plugin

Spring Boot Project

We can start working with spring boot in three different ways:

- Using Spring Initializr
- Using STS
- Using CLI

Spring Boot Project using Spring Intializr

- Open the Spring Initializr using the following link: Spring Initializr
- Provide group and artifact name.
- Add Spring - web dependency and click on generate.
- Extract the downloaded zip file and extract it.
- Import the project folder into your IDE (Eclipse).
- Run the Application (Application.java) -> Run as java application

Spring Boot Project using Spring Tool Suite (STS)

- Install STS ide and follow the process given below.
- Create a maven project by providing a group and artifact name.
- Add dependencies in pom.xml
- Run the application (Application.java) -> Run as java application

Spring Boot Project using CLI

- It is a tool which you can download from the official site of Spring Framework.
- Spring Boot project creations through CLI are outdated and no longer used.
- Use the following Link to work using Spring Boot CLI: SB CLI Guide

Example: To print hello in console

- Open the Spring Intializr using the following link: Spring Initializr
- Provide group and artifact name.
- Add Spring - web dependency and click on generate.
- Extract the downloaded zip file and extract it.
- Import the project folder into your IDE (Eclipse).
- Run the Application (Application.java) -> Run as java application
- Add the following changes in Application.java file
- src/main/java -> com.example.demo ->DemoApplication.java

DemoApplication.java

```
@SpringBootApplication
public class DemoApplication{
    public static void main(String args[]){
        System.out.println("Hello Spring Boot");
      }
    }
```

Spring Boot Application/Project Components

application.properties

- In Spring Boot, whenever you create a new Spring Boot Application in spring starter, or inside an IDE (Eclipse or STS) a file is located inside the *src/main/resources* folder named as **application.properties**
- This file contains the different configurations which is required to run the application in a different environment, and each environment will have a different property defined by it.

- Inside the application properties file, we define every type of property like changing the port, database connectivity, connection to the eureka server, and many more.

Example: To change the port name

- Sometimes when you run your spring application you may encounter an error stating as **port already in use.**
- So, in this case you may change your port number and rerun your application and we can change the port number using by adding below lines in **application.properties.**
- Add the below lines in application.properties
  - spring.application.name=demo
  - server.port = 2024

Annotations

Spring Boot Annotations are a form of metadata that provides data about a spring application.

Core Spring Framework Annotations

@Required

- It is method level annotation.
- This annotation will be used for a particular method which is mandatory to execute.

@ComponentScan

- It is a class Level annotation.
- It is used when we want to scan a package from beans.

@Bean

- This is a method level annotation.
- It is an alternative of XML <bean> tag.
- It tells the method to produce a bean to be managed by the spring container.

@Component

- It is a class level annotation.
- It is used to mark a java class as a Bean.

@Service

- It is a class level annotation.
- It is used in a class where business logic of the application is defined.

@Repository

- It is a class level annotation.
- This annotation is used to a class where Data Access Object (DAO) implements.

Spring Boot Annotations

Spring annotations are present in **org.springframework.boot.autoconfigure** and **org.springframework.boot.autoconfigure.condition** packages.

@SpringBootApplication

- It is a class-level annotation that is used to mark the main class of a Spring Boot application

- It is a combination of following three annotations:
    - @EnableAutoConfiguration
    - @ComponentScan
    - @Configuration

@Controller Annotation

- This annotation provides **Spring MVC** features.
- It is used to create Controller classes and simultaneously it handles the HTTP requests.
- Generally, we use **@Controller** annotation with **@RequestMapping** annotation to map HTTP requests with methods inside a controller class.

@RestController

- It is class level annotation which is used to handle **REST APIs** such as GET, PUT, POST, DELETE and used to create **RESTful web services** using Spring MVC.
- It is a combination of following two annotations:
    - @Controller
    - @ResponseBody

@RequestMapping

- It is a **method-level annotation** which is used to map the **HTTP requests** with the handler methods inside the controller class.
- Syntax: @RequestMapping (method = RequestMethod.GET)

@GetMapping

- It is a **method level annotation** which is used to handle specific Http requests.
- It maps the **HTTP GET** requests on the specific handler method. It is used to create a web service endpoint that **fetches a resource.**

@PostMapping

- It is a **method level annotation** which is used to handle specific Http requests.
- It maps the **HTTP POST** requests on the specific handler method. It is used to create a web service endpoint that **creates a resource.**

@PutMapping

- It is a **method level annotation** which is used to handle specific Http requests.
- It maps the **HTTP PUT** requests on the specific handler method. It is used to create a web service endpoint that **creates** or **updates a resource.**

@DeleteMapping

- It is a **method level annotation** which is used to handle specific Http requests.
- It maps the **HTTP DELETE** requests on the specific handler method. It is used to create a web service endpoint that **deletes** a resource.

@PatchMapping

- It maps the **HTTP PATCH** requests on the specific handler method.

@RequestBody

- It is used to **bind HTTP request** with an object in a method parameter.
- Internally it uses **HTTP Message Converters** to convert the body of the request.
- When we annotate a method parameter with **@RequestBody**, the Spring framework binds the incoming HTTP request body to that parameter.

@ResponseBody

- It binds the method return value to the response body.
- It tells the Spring Boot Framework to serialize a **return an object into JSON and XML format.**

@PathVariable

- It is used to **extract the values** from the URI.
- It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple @PathVariable in a method.

@RequestParam

- It is used to **extract the query parameters** form the URL.
- It is most suitable for web applications. It can specify default values if the query parameter is not present in the URL.

@RequestHeader

- It is used to get the details about the HTTP request headers.
- We use this annotation as a method parameter. The optional elements of the annotation are name, required, value, defaultValue.

@RequestAttribute

- It binds a method parameter to request attribute. It provides convenient access to the request attributes from a controller method.
- With the help of **@RequestAttribute annotation**, we can access objects that are populated on the server-side.

Dependencies Management

- Spring Boot manages dependencies and configuration automatically. Each release of Spring Boot provides a list of dependencies that it supports.
- It provides the centralization of dependency information by specifying the Spring Boot version in one place. It helps when we switch from one version to another.
- It avoids mismatch of different versions of Spring Boot libraries.
- We only need to write a library name with specifying the version. It is helpful in multi-module projects.

Spring Starters

Spring Starter is a collection of dependencies which comes as a single unit.

The different dependencies of spring starters are:

- SB starter web
- SB starter test
- SB starter actuator
- SB starter parent
- SB starter JPA
- SB thymeleaf
- SB dev tools
- SB security

SB Starter Web

- This is a dependency which is used to build a spring boot web application.
- Add dependency -> spring-starter-web

Spring Starter Test

- This is a default dependency which you have for every spring boot application.
- Default Dependency -> Spring-starter-test

Spring Starter JPA

- **JPA** stands for Java Persistence API.
- The purpose of this is **Database Connectivity** using JPA / Hibernate.
- JPA / Hibernate uses JDBC concepts to communicate with database.
- Automatic data source configuration is an advantage of SB framework (No need of any configuration files).
- Entity Manager is the main component / Class of Spring Boot framework to create queries.

REST - CRUD

- **REST** stands for **Representation State Transfer.**
- It refers to an architectural style for designing different applications (web applications).
- REST is not an API, it's architecture.
- When you develop RESTFUL web services in SB application, you are implementing a REST architecture based on REST principles.
- This involves different HTTP methods such as GET, PUT, POST, DELETE.
- In this concept, we will understand different http service methods to perform crud operations on a database.

Spring Boot MVC

MVC stands for **Model, View, Controller.**

MVC is a standard architecture or design pattern which is used to design web applications.

MVC:

- Model represents the business logic of our application
- View represents the presentation logic of our applications.
- It acts as an interface between model and view or FE and BE

Spring Starter Thymeleaf

- It is a server-side java template engine which is used to create and render html, css, js, jsp, xml files.
- It is integrated with the Spring Boot framework to represent the view of an application.

Spring Starter Parent

- Spring starter parent is a special dependency that provides the default configuration of our application, and it provides a complete dependency tree to build a SB application.
- This dependency inherits dependency manager from "spring-boot-dependencies".

Spring Starter Actuator

- It is a dependency used to manage and monitor our complete SB application.
- This dependency is having an actuator model which consists of several features like health checkup, auditing jvm metrics, log information, cache statistics, etc.

Spring Starter DevTools

- It is a module that provides several features to improve the development of our web application.
- Some of the important features provided by this dependency are:
    - automatic restart
    - live reload
    - remote update
    - remote debugging

Spring Starter Security

- It is a framework which provides security to web applications which is built using SB.
- This framework targets on two major areas of a web application w.r.t security:
    - authentication
    - authorization

Authentication

- It is the process of knowing and identifying the user who wants to access the web application.

Authorization

- It is a process which allows the user to perform different actions on a web application.
- With respect to SB the security can be done in 2 ways:
    - Default Security
    - Custom Security

Spring Boot App Deployment on Render

Key Points about Render

- Render is a modern serverless platform designed for easy deployment and scaling of web applications, including Spring Boot applications.
- Render simplifies the deployment process by allowing applications to be deployed directly from a Git repository, eliminating the need for manual infrastructure management.
- Automatic scaling ensures optimal performance and resource utilization based on application demand, with high availability and reliability.
- Built-in monitoring and logging capabilities help developers track application performance and troubleshoot errors effectively.
- Render supports custom domain integration and provides automatic SSL certificate management for secure HTTPS encryption.
- Collaborative features include access control, environment variables management, and Git integration for streamlined team collaboration.
- Render offers competitive pricing models tailored to different use cases, with comprehensive documentation, tutorials, and community forums for support.

Deployment Steps

- Create a simple spring project using Spring Initializr.
- Create a controller class and required HTML pages.
- Run the application in localhost to verify your output.
- After verification, package your application into a JAR:
    - In Eclipse: Right-click on APP_NAME -> Run As -> Maven Build.
    - In the Goals field, enter `clean package` and run.
- Wait until a JAR snapshot is created successfully and refresh the entire project once.
- Verify whether you have a file like this: `APP_NAME.0.0.01-SNAPSHOT.jar` under your target folder.
- Open terminal or CMD, navigate to your app's target folder using cd:

    - cd C:\Users\DELL\Downloads\APP_NAME\target
- Enter the below command:
    - java -jar APP_NAME.0.0.01-SNAPSHOT.jar
- Wait for some time until you get something like: "Started your application."
- Create a file named "Dockerfile" and add the following lines:
    - FROM eclipse-temurin:17-jdk-alpine
    - VOLUME /tmp
    - COPY target/*.jar app.jar
    - ENTRYPOINT ["java", "-jar", "/app.jar"]
    - EXPOSE 8080
- Navigate to your project root directory:
    - cd C:\Users\DELL\Downloads\APP_NAME
- Initialize an empty Git repository using below command:
    - git init
- If a README file is generated, then remove the following lines from it:
    - target/
    - !.mvn/wrapper/maven-wrapper.jar
    - !**/src/main/**/target/
    - !**/src/test/**/target/
- If you don't remove them, your target folder will not be added to GitHub.
- Enter the following commands one after another to push your app to Git:
    - git add .
    - git commit -m "SOME COMMIT NAME"
    - git remote add origin https://github.com/username/repository
    - git push -u origin master
- For a clearer understanding of Git, refer to this document:
    - https://online.fliphtml5.com/rhriu/czyg/
- After successfully uploading the file to Git, open Render, login, and create a new web service. Select upload from Git.
- Wait until it gets deployed. It may take some time.
- Verify your deployment.

Disclaimer

- The following notes are general summaries and overviews of the topics discussed.
- These notes are not exhaustive and do not cover all aspects of the subject matter.
- The information provided herein is intended for educational purposes only and should not be used as a substitute for professional advice, detailed study, or official course materials.

References

For more detailed information, please refer to the following resources:

Reference 1: Complete Spring Boot Tutorial

Reference 2: Complete Spring Boot Tutorial

Reference 3: Spring Boot Cheat Sheet

Reference 4: Spring Boot Most Asked Questions

Reference 5: Top Spring Boot Interview Questions

Prepared By: ***Reddy Venkat Kalyan***
