Introduction to Spring Boot

Spring is widely used for creating scalable applications. For web applications Spring provides Spring MVC which is a widely used module of spring which is used to create scalable web applications.
But main **disadvantage of spring projects is that configuration is really time-consuming and can be a bit overwhelming for the new developers.** Making the application production-ready takes some time if you are new to the spring.

Solution to this is Spring Boot. Spring Boot is built on the top of the spring and contains all the features of spring. And is becoming favourite of developer's these days because of it's a rapid production-ready environment which enables the developers to directly focus on the logic instead of struggling with the configuration and set up.

Spring Boot is a microservice-based framework and making a production-ready application in it takes very less time.
Prerequisite for Spring Boot is the basic knowledge Spring framework.
For revising the concepts of spring framework

Features of Spring Boot

Spring Boot is built on the top of the conventional spring framework. So, it provides all the features of spring and is yet easier to use than spring.

- **It allows to avoid heavy configuration of XML which is present in spring:**
  Unlike the Spring MVC Project, in spring boot everything is auto-configured. We just need to use proper configuration for utilizing a particular functionality.
  For example: If we want to use hibernate(ORM) then we can just add **@Table annotation** above model/entity class(discussed later) and add **@Column annotation** to map it to table and columns in the database
- **It provides easy maintenance and creation of REST end points:**
  Creating a REST API is very easy in Spring Boot. Just the annotation **@RestController** and **@RequestMapping(/endPoint)** over the controller class does the work.
- **It includes embedded Tomcat-server:**
  where we have to manually add and install the tomcat server, Spring Boot comes with an embedded Tomcat server, so that the applications can be hosted on it.
- **Deployment is very easy, war and jar file can be easily deployed in the tomcat server:**

- **Deployment is very easy, war and jar file can be easily deployed in the tomcat server:** **war** or **jar** files can be directly deployed on the Tomcat Server and Spring Boot provides the facility to convert our project into war or jar files. Also, the instance of Tomcat can be run on the cloud as well.
- **Microservice Based Architecture:**
  Microservice, as the name suggests is the name given to a module/service which focuses on a single type of feature, exposing an API(application peripheral interface).
  Let us consider an example of a hospital management system.
  - In case of monolithic systems, there will be a single code containing all the features which are very tough to maintain on a huge scale.
  - But in the microservice-based system, each feature can be divided into smaller subsystems like service to handle patient registration, service to handle database management, service to handle billing etc.

  Microservice based system can be easily migrated as only some services need to be altered which also makes debugging and deployment easy. Also, each service can be integrated and can be made in different technologies suited to them.

Evolution of Spring Boot

1. Spring Boot came into existence when in October 2012, a client, **Mike Youngstrom** made a Jira request asking for bootstrapping the spring framework so that it can be quickly started. And hence in early 2013, Spring Boot was made.
2. In April 2014, Spring Boot **1.0** was created followed by various versions.
3. Spring Boot **1.1** on June 2014,
4. **1.2** in March 2015,
5. **1.3** in December 2016,
6. **1.4** in January 2017 and
7. Spring Boot **1.5** on February 2017.

Spring Boot Architecture

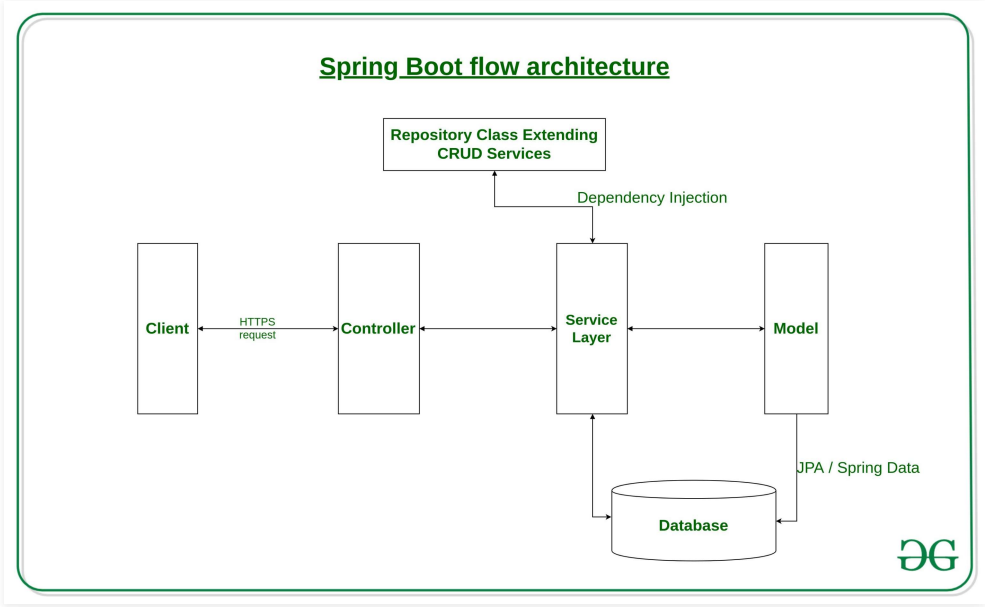
To understand the architecture of Spring Boot, let us first see different layers and classes present in it.

- **Layers in Spring Boot:** There are four main layers in Spring Boot:
  - **Presentation Layer:** As the name suggests, it consists of views(i.e. frontend part)
  - **Data Access Layer:** CRUD (create, retrieve, update, delete) operations on the database comes under this category.

- **Service Layer:** This consist of service classes and uses services provided by data access layers.
- **Integration Layer:** It consists of web different web services(any service available over the internet and uses XML messaging system).

- Then we have utility classes, validator classes and view classes.
- All the services provided by the classes are implemented in their corresponding classes and are retrieved by implementing the dependency on those interfaces.

Spring Boot flow architecture:



- Since Spring boot uses all the features/modules of spring-like Spring data, Spring MVC etc. so the architecture is almost the same as spring MVC, except for the fact that there is no need of **DAO** and **DAOImpl classes** in Spring boot.
- Creating a data access layer needs just a repository class instead which is implementing CRUD operation containing class.
- A client makes the https request(PUT/GET)
- Then it goes to controller and the controller mapped with that route as that of request handles it, and calls the service logic if required.
- Business logic is performed in the service layer which might be performing the logic on the data from the database which is mapped through JPA with model/entity class
- Finally, a JSP page is returned in the response if no error occurred.

Setup Spring Boot:

1. Setup Java JDK from Oracle's official site.
2. Download and Setup STS(Spring Tools Suite).
3. Start a new spring starter project

- Click on File -> New -> Spring starter project
- Fill the appropriate details and add dependency and finish.
- Edit the application properties.
- Run the main file as a Java application.