* Spring Boot :

Spring is widely used for creating scalable applications. for web spring provides spring mvc which is useful for scalable web applications.

But, main disadrantage of spring projects is that configuration is really time-consuming and can be a bit overhelming for the new developers. Making the application production-ready takes some time if you are new to the spring.

- Solution is Spring Boot. Spring Boot is built on the top of the spring and contains all the features of spring.

- Spring Boot is a rapid production-ready environment which enables the developers to directly focus on the logic instead stuggling with the config. and set up.

- Spring Boot is a microservice-based framework and making a production-ready application in it takes very less time.

* 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 yet easier to use than spring.

- It allows to avoid heavy configuration of XML which is present in spring:

  Unlike the spring mvc, in spring Boot everything is auto-configured. We just need to use proper configuration for utilizing a particular functionality. For example: if you want to use hibernate then just add @Table annotation above entity.

- It provides easy maintenance and creation of REST endpoints:

  Creating REST API is very easy in Spring Boot. Just annotation @RestContoller and @Request-Mapping (/endpoint) over the contoller class does the work.

- It includes embedded Tomcat-Server:

- Deployment is very easy, war and jar file can be easily deployed in the tomcat server:

  war or jar file 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 ~~and~~ which focuses on a single type of feature, exposing an API.

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 diffrent technologies suited to them.

* Spring Boot Architecture :

- Layers in Spring Boot :

① Presentation Layer :

As the name suggests, it consists of views (i.e. frontend part)

② Data Access Layer :

CRUD operations on the database comes under this category.

④ ~~Service Layer~~: Integration Layer:

It consists of web diffrent web services (any service available over the internet and uses XML messaging System.

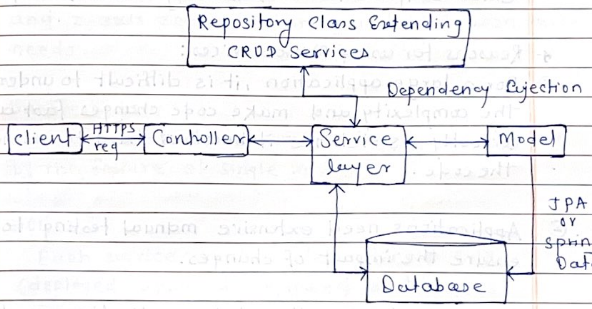
③ Service Layer :

This consists of service classes and uses services provided by data access layers.

- Then we have utility classes, validator classes and view classes.

- All the services provided by the classes are implemented in their corrensponding classes and are retrieved by implementing the dependency on these interfaces.

* Spring Boot flow Architecture :



* Microservices :

A microservice is a small, loosely coupled distributed service. Microservice Architectures evolved as a solution to the scalability and innovation challenges with Monolith architechures.

It allows you to take a large application and decompose or break into easily manageble small components with narowly defined responsibilities.

* Reasons for using Microservices:

① For a large application, it is difficult to understand the complexity and make code changes fast and correctly, sometimes it becomes hard to manage the code.

② Applications need extensive manual testing to ensure the impact of changes.

③ For small change, the whole application needs to be built and deployed.

④ The heavy applications slows down start-up time.

* Benefits of Microservices:

① Small Modules:

Application is broken into smaller modules which are easy for developers to code and maintain.

② Easier Process Adaption:

By using microservices, new Technology and process Adaption becomes easier. You can try new technologies with the newer microservices that we use.

③ Independant Scaling:

Each microservice can scale independently via X-axis scaling (cloning with more cpu or memory) and Z-axis scaling (sharding), based upon their needs.

④ Unaffected:

Large applications remain largely unaffected by the failure of single module.

⑤ DURS:

Each service can be independently DURS (deployed, updated, replaced and scaled)

* Restrictions of Microservices:

① Configuration Management

② Debugging

③ Automation

④ Testing

* Microservices frameworks for Java:

1. Spring Boot
2. Dropwizard
3. Restlet
4. Spark

Ninja Web framework

Play framework

Rest Express

Restx framework