

Spring Boot

Spring Boot

Spring Boot is a framework built on top of Spring framework.

• This simplifies the development, configurations.

• As Spring Boot is developed by "Pivotal Teamwork".

• It is an open source framework.

• Spring Boot develops end-to-end applications.

Auto Configuration

Auto Configurations

→ Reduces the lines of code written by the developer.

→ Handles jars with version management.

• In Java projects structure won't ^{support} ~~allow~~ us to add dependencies & unable to handle jar files.

• By ^{support of} Maven ~~project~~ tool we have used the integration of dependencies also the configurations will be connected directly to the internet & will try to download required things.

• In same way spring starter project helps us to give predefined code for application's development.

Spring Boot

Spring Boot

- Spring Boot is a framework built on top of Spring framework.
- This simplifies the development, configurations.
- This Spring Boot is developed by "Pivotal Teamwork".
- It is an open source framework.
- Spring Boot develops end-to-end applications.

Auto Configuration

Key Configurations

- Reduces the lines of code written by the developer.
- Handles jars with version management.

→ In Java projects structure won't ^{support} ~~allow~~ us to add dependencies & unable to handle jar files.

→ By ^{support of} Maven ~~project~~ tool we have used the integrations of dependencies also the configurations will be connected directly to the internet & will try to download required things.

→ In same way spring starter project helps us to give predefined code for application development.

Spring starter projects

- I also call it as Abstract maven project
- It can do much more things than maven.
- It will automatically have some generated code in it that we need to use.
- A project with partial code with JAR files.

→ Here programmer will not write the configuration, but need to give the input data.

- i) properties file (application.properties)
- ii) YAMAL file (application.yml)

Difference b/w Spring and Spring Boot

→ Both are frameworks.

Spring Core	Spring Boot
<ul style="list-style-type: none">→ We need add dependencies and JAR files.→ We need to configure by ourselves.→ Configuration done by Maven.→ The Application development, which is going to provide modules for dependency injection→ Manually configure XML or Java based files.→ No server, no support for web application.	<ul style="list-style-type: none">→ We have Configuration inbuilt.→ It simplifies Configuration, deployments & also adds inbuilt servers.→ Configurations are completely automated.→ The writing of code will be reduced.→ We have embedded server called Tomcat, which is easy to deploy projects.

Spring Core

- Needs additional setup.
- Management of dependencies, user need mention versions manually.
- Application Development is slow, because we need configuration, perform dependency injections, additional dependencies.
- The control should be completely managed by the developer.

Spring Boot

- Completely simple & easy.
- Select what dependencies we need.
- Application Development is faster and built in.
- The complete control managed by Spring starters, we just need to focus on business logic.

What is start class in Spring Boot?

It is the entry point for Boot applications execution.

Start class is also called as main class ~~also~~ in Spring Boot.

Note:- When we create the Spring Boot project, start class will get created by default.

@SpringBootApplication

It is equivalent to @Configuration,
@EnableAutoConfiguration and @ComponentScan
annotations.

@SpringBootApplication.run()

It is the method which helps to start the
JOC

which class is helping to start the JOC

@SpringBootApplication

```
public class Application {
```

```
    public static void main(String[] args)
```

```
    {
```

```
        ConfigurableApplicationContext ref = SpringApplication.
```

```
            run(Application.class, args);
```

```
    }
```

```
}
```

app:- AnnotationConfigApplicationContext

Default Dependencies

Spring-boot-starter:-

→ core dependency.

→ pull the Spring Core, Spring Beans, Spring
Context.

→ Base for all the starters.

spring-boot-starter-test:-

- Includes testing libraries like JUnit, Mockito, AssertJ and Spring Test.
- Used for Unit and Integration Testing.

→ To auto restart when you change code, we use "Spring Dev Tools" dependency.

Spring Banner:-

we have 3 different modes:-

1. console (default)
2. log → print logo inside the log file.
3. off.

~~spring.main~~
default

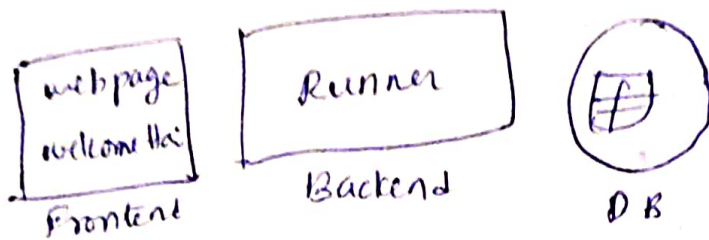
spring.main.banner-mode = console

To stop printing the banner in the console

spring.main.banner-mode = off

Runners In Spring Boot

- A runner is an auto executable component which is called by JOC, an application startup.
- It only executes only once (on Application startups)
- In simple this concept is used to execute any logic one time when application is getting started.



Types of Runners

- 1. Application Runner.
 - 2. CommandLine Runner.
- } Functional interfaces. → run method

Types of Interfaces

- 1. Regular :- Multiple abstract methods
- 2. Functional :- It has only one abstract method
- 3. Marker :- we won't have method (empty interface)

1. Application Runner

@Component

public class implements ApplicationRunner {

@Override

public void run(ApplicationArguments args) throws

~~P.S.V. M (SpringBootApplication) throws Exception {~~

S.O.P ("Loading all the required things");

}

Command Line Runner:-

@Component
public class

Startup implements CommandLineRunner {

@Override

~~public~~

public void run(String... args) throws Exception {
 S.O.P("Loading the Data");

}

}

When to use?

Application Runner:- when you need structured argument parsing. For huge amount of data to access, we use Application Runner.

Command Line Runner:- For simple post startup logic needing command-line arguments.

Application

Note to remember the usage of application's properties:

Allowed special symbols are :- dot(.), hyphen(-) & underscore(_).

Keys and values are allowed should be as String.

Spring supports type conversion

Eg: String \Rightarrow int

String 12345 \Rightarrow int 12345

Spring AOP (Aspect oriented programming)

AOP is a programming paradigm that allows you to separate cross-cutting concerns (like logging, security, transactions etc--) from the business logic of application.

In Spring AOP, we use Aspects (reusable code) that can be applied to different parts of the application without modifying the actual code.

Aspect:-

A class that contains cross cutting concerns (like logging)

Advice:- The action taken by an aspect.

Join Point:- A point in the application where an advice can be applied.

Pointcut:- Expression that defines where an advice should be applied.

Types of Advices

1. Before - @Before
2. After - @After
3. Around - @Around
4. After Returning - @AfterReturning
5. After Throwing - @AfterThrowing

Logging:-

It is an essential practice in software development that involves recording information about a program's execution.

Good logging helps developers monitor application, diagnose problems and understand the system behaviour over time.

Spring Data JPA ..

JPA (Java Persistence API)

CRUD 1: USER \Rightarrow 4 Queries (4 methods)

CRUD 2: PRODUCT \Rightarrow 4 methods

CRUD 3: PAYMENT \Rightarrow 4 methods

1000: $1000 \times 4 \Rightarrow 4000$ - 4 methods

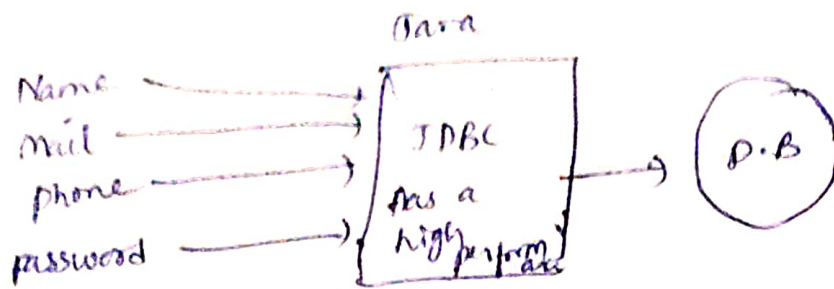
Spring Data JPA is one of the module in Spring Framework.

It is used to develop persistence layer (DB logic)

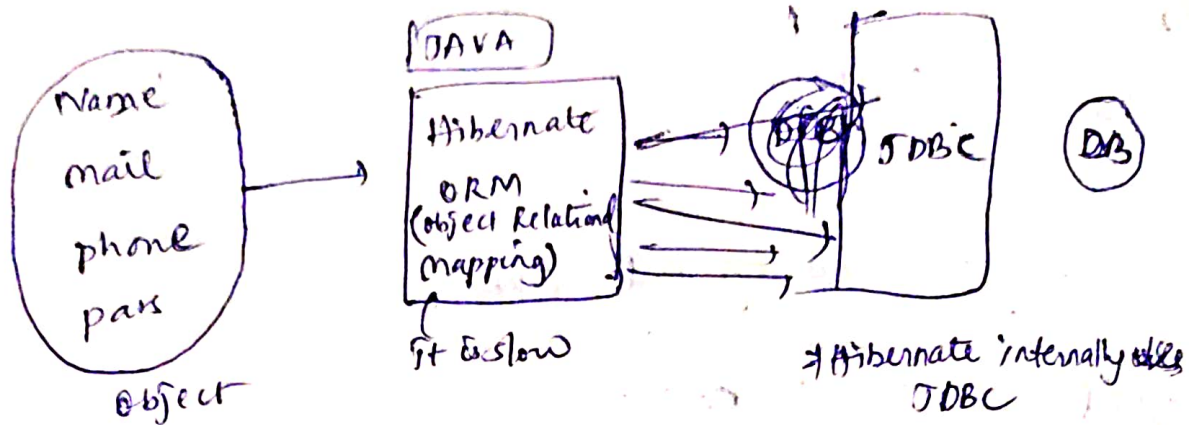
To avoid boiler plate code in DAO classes, we

use Spring data JPA.

Spring data JPA will use Hibernate Framework internally.



⇒ sends each value to JDBC

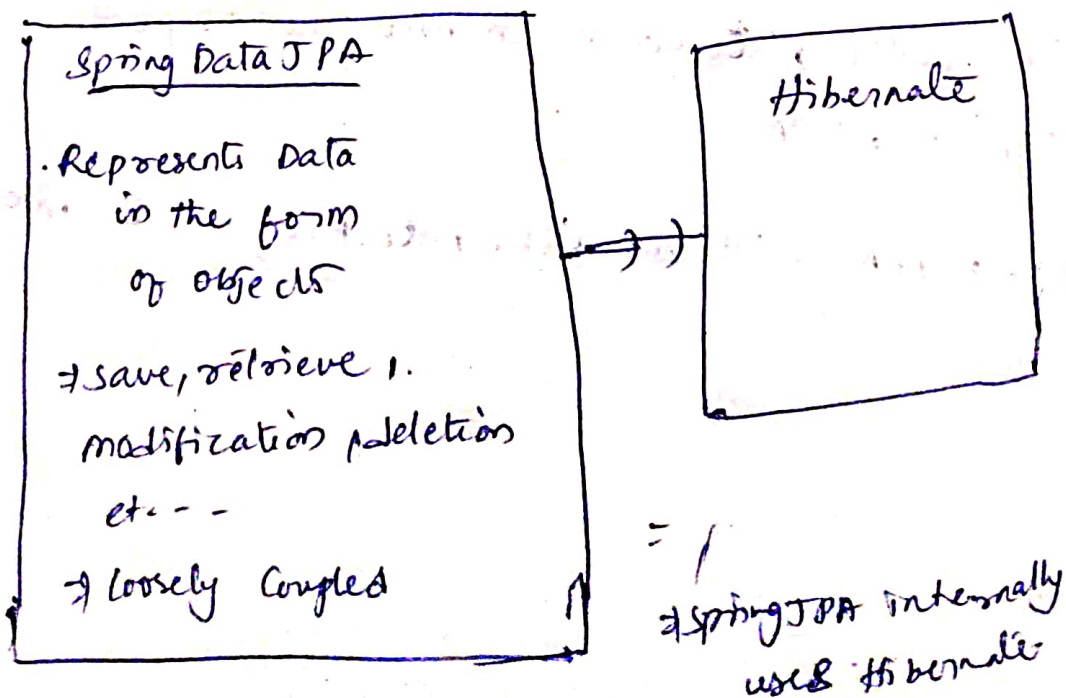


1. HQL queries (Hibernate Query Language)

2. JPQL (Java Persistence Query Language) → Database Independent queries.

3. JPQL is converted into SQL queries according to DB.

⇒ The performance is low.

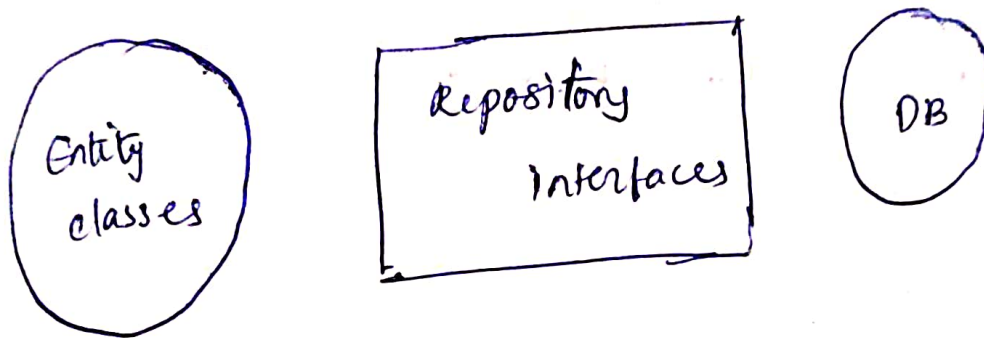


Spring Data JPA Repositories :-

→ simply persistence layer in development data JPA provided repository interfaces.

1. CRUD Repository :- Performs CRUD operations methods
2. JPA Repository :- (CRUD operations methods + sorting methods + pagination + QE (Query By Example))

Dependency :- Spring Data Jpa



Developing First JPA Application (Project)

1. Create Spring Boot Project with dependencies:
 - * Data-jpa → starter
 - * Mysql - Driver
 - * project - lombok
2. Configure datasource in application.properties file
3. Create an Entity class.
4. Creating Repository Interface by extending the CrudRepository interface

Crud Repository

save()

findAllById()

deleteAll()

saveAll()

count()

findById()

deleteById()

existsById()

delete()

findAll()

deleteAllById()

Requirement:-

Fetch books whose price is > 300

⇒ Find By Methods

⇒ Custom ~~queries~~ queries

Find By methods

Find By Methods are used to perform only select queries (operations).

⇒ Using Non primary keys columns also we can select records.

⇒ In find by methods name is very important.

⇒ Because, based on method name Jpa will convert the query for execution.

⇒ Find By Methods should represent Entity class variables.

Custom Queries

are custom queries:

- SQL queries are DB independent queries.
- HQL queries are DB dependent queries.
- In HQL we will use the Entity classname and variables.
- In SQL we will use the table name and column names.
- HQL queries can't execute in DB directly (conversion has to take place)
- SQL queries can be executed directly in DB.
- SQL is better in performance
- HQL is better in maintenance.
- Every HQL queries converted into SQL queries before execution
- That conversion will be done by dialect classes.
- Every Database will have Own dialect class.

Spa Repository

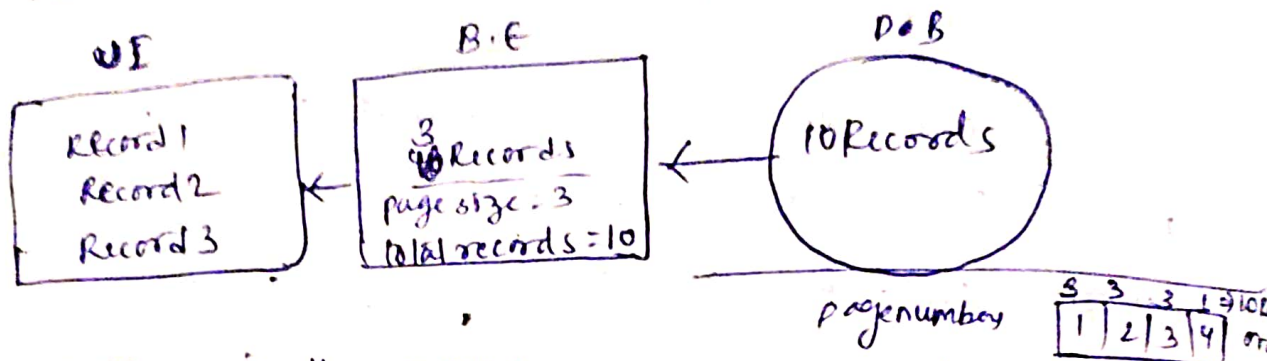
⇒ spa repository is an interface.

⇒ Query sorting + pagination + OBE (Query By Example)

Sorting

ascending, descending

Pagination



⇒ Pagination is the process of dividing a large set of data into pages.

⇒ This helps in reducing the load on the server.

⇒ Improves the performance.

⇒ Gives better user experience by avoiding the long scrolling.

Query By Example (QBE):-

QBE is a way to query the database by creating a dummy object (example) with the data you want to search for.

⇒ Spring Data spa compares the fields in the object and fetches the matched records automatically.

Spring Data JPA Relations

→ one to one relation ⇒ @OneToOne

→ one to many ⇒ @OneToMany

→ many to one ⇒ @ManyToOne

→ many to many ⇒ @ManyToMany

<u>Person</u>			<u>Passport</u>	
id	name	pass id	pass id	passport
1	Ping	2	1	100011
2	Sundara	1	2	300011
3	Pings	3	3	400011

PK (primary key) is indicated for the 'pass id' column in the Person table. A 'Primary key' label is also present above the 'pass id' column in the Passport table.