1. What is persistence store?

The place where we will save our data is called as persistence store.

1. Persistence operations: insert, retrieve ,update, delete(CRUD)
2. Persistence technologies:

* Persistence technologies are used to develop persistence logic
* Persistence logic is used to perform persistence operations in persistence store.

1. We have several technologies to develop persistence logic

* Java JDBC
* Spring JDBC
* ORM framework
* Spring ORM
* Spring Data JPA etc.

Best Practices to develop persistence logic

* We should maintain persistence logic as separate layer(DAO or Repository layer)
* Generally we will call it as Data Access Layer(DAO)
* In persistence logic we should fallow Table per DAO concept
* For every table dedicated DAO we need to create(Single Responsibility principle)
* For every table dedicated DAO we need to create. That DAO layer responsible to perform operations with the table.(which improves readability)

E.g. ContactDao.java, PhoneDao.java

Or ContactDaoExtender.java

A class should not exceed more than 2000 lines

If we want to write join query, like we need to join two tables Contact and phone we can write query in either Contact DAO or in phone DAO.

If our method talking to one table, then we have to write query in that particular Dao only.

When we are creating table names, column names, sequence names the length limitations is 30 characters.

If required name is having more than 30 characters generally DB people will ignore vowels from the name.

e.g. USER\_MASTRE--------------------🡪USR\_MSTR

For every table atleast one primary key is highly recommended.

For every table, it is highly recommended to maintain below auditing columns

CREATED\_DATE, CREATED\_BY,UPDATED\_DATE,UPDATED\_BY

For every primary key create dedicated sequence to generate primary key column value

Always recommended to maintain cache for static table data to avoid DB interactions(for e.g if there is a static data we only read the data from that table we don’t perform any other CRUD operation s .Whenever we need that data ,instead of hitting the DB we can retrieve by cache only. (which will improves the performance of the application)

**Data Base Installation**

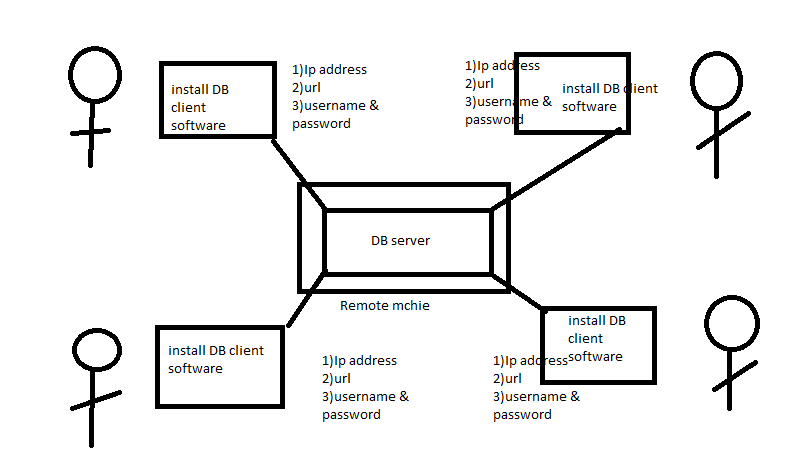
1. Data Base Server
2. Data Base client

In company environment DB Server will be installed by DB admins in Remote machines.

After installing DB server, DB team will share DB details to team members

(Username, Password, URL, Service Name)

Who will install DB server in company? Data base administrator will install



Consider one project several peoples are working there is One Database server and all are working on that DB server all peoples need to install DB client software to working on that DB server(they can connect to the DB server by using ip address, username ,password of that DB server)

Developers and testers will install Data Base software in their machines

For Oracle DB server, install. SQL Developer, Toad etc .

For Mysql Db server install Mysql workbench

Based on DB we have to install client software.

Why Spring Data JPA?

**Common persistence Operations:**

**Below operations are very generic operations perform in DB**

* Insert One record
* Insert multiple records at a time
* Update record
* Update multiple records at a time
* Retrieve one record
* Retrieve few records
* Retrieve all records
* Delete one record
* Delete few record at a time
* Delete all records
* Get count of records

To perform above operations we need to write methods in DAO interface and we should implement those methods in DaoImpl class.

If our project contains 1000 tables then we will create 1000 Dao interfaces.

In all Dao interfaces all above methods will be common(We need to implement all methods).

Note: All the implementation classes will contain same logic(Boiler plate code)

To avoid Boiler Plate code in Dao layer Spring Data came into picture.

If we use Spring Data then no need to write even single method also to perform above CRUD operations for table.

Spring data providing predefined methods for us to perform CRUD operations.

Hibernate also provide methods but we need o write configurations, session factory, session.

In spring data no need to do anything everything will taken care by Spring Data only.

For this reason spring data popular in the market.

What is the disadvantage of Spring Data?

It is difficult to write Join Query in the Spring Data. Specially they designed the Spring data for one table communication.

Now a days spring ORM is used in the Micro Service based applications. One micro service will do one small operation. For that Spring data is highly recommended.

For monolithic complex application Spring data is not recommended.

What is spring Data?

Spring Data is part of spring framework which is used to develop persistence layer in our application.

Spring Data provided 2 predefined repository interfaces, they are

1. CrudRepository
2. JpaRepository

These two interfaces provided several methods to perform Crud operations

To use spring data repository interfaces provided methods we should create our repository interface by extending Spring Data Repsoitory interfaces.

e.g. public interface MyOwnRepository extends CrudRepository<E,ID>{

}

or

public interface MyOwnInterface extends JpaRepository<E,ID>{

}

**What is the difference between CrudRepository and JpaRepository?**

* CrudRepository provided methods to perform crud operations.
* JpaRepository provided to perform crud operations + paginations + sorting

Whatever the functionalities that we have in the CrudRepository that functionalities we have in the JpaRepository also and additional functionalities also.

Pagination: The process of displaying records in multiple pages is called pagination.

Organizing the data in multiple pages. Because if we display data in single page it does not look good.

e.g. Gmail ->In Gmail open inbox in the right side it will show 50 of 100 < >

it we want to see next few mails then we have to click > arrow.

e.g. search anything in Google first it will display few data then we need to click page numbers 1 2 3 4 5 6 7 8 9 10 like this.

Note: we can say JpaRpository is superior to Crud repository.

**Steps to develop spring boot application using Spring Data JPA:**

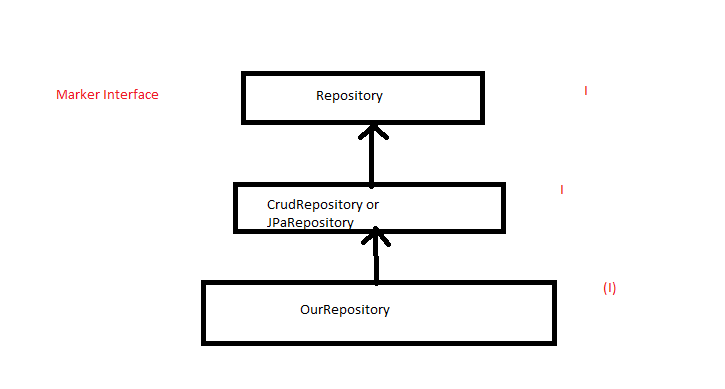
**Note : make sure DB server is installed in your machine**

1. Create Spring Boot stand alone application using following dependencies
2. Spring-boot- starter
3. Spring-boot-starter-data-jpa
4. Project Lombok
5. My- sql- driver
6. Configure Data Source properties in application. properties file
7. Create Entity class (Java class------------------------db table mapping)
8. Create repository interface for our entity by extending spring data repository
9. Test application behavior by using spring data methods

What is Entity class?

A java class which is representing structure of the DB table is called as Entity.

Our object should mapped with DB table data.



Our interface indirectly extending properties from Repository interface ( this is marker interface)

As our interface extending properties from repository interface spring data will provide implementation class for our interface in the run time by using proxy design pattern.

@Repository 🡪it is used to define spring bean like service, component, configuration etc

Writing this annotation is optional. Even if we don’t write this annotation also spring data will consider this as Spring bean ,because interface will extends CrudRepository or JpaRepository.

Whenever our application starts component scanning will happen.

When we are using Spring Data Jpa first they will scan for data repositories available in the class path. Spring data will provide a implementation class for repositories in the run time.

What is the use of extending Marker interface?

Special instructions we are providing to get the extra functionality.

During runtime implementation class of our repository will be created.

To check wethere it is created or not,got main class, you will find one method SpringApplication.run.

SpringApplication.*run*(SpringdatajpaappApplication.**class**, args); --🡪this will returns ConfigurableApplicationAontext()

ConfigurableApplicationContext context=SpringApplication.run(Application.class);

ContactMasterRepository bean=context.getBean(ContactMasterRepository.**class**);

System.***out***.println(bean.getClass().getName()); //output jdk.proxy4.$proxy98

(this is implementation class name) for this we can conclude that spring data jpa will create implementation class for our repository during run time.

CrudRepository:

* It is an interface provided by Spring Data Jpa.
* In this interface we have several methods which are used to perform CRUD operations.
* To create user defined repository interface we will extend properties from this CrudRepository interface.

If we create user defined repository interface, then implementation class will be created in runtime for our user defined repository interface(proxy class)

**Methods available in CrudRepository interface:**

1. **Object Save(T entity):** It is used to insert record into table . This is polymorphic method, the same method is used for both insert and update.

When it is going to insert, when it is going to update? Which algorithm it is going to use?

It depends on primary key column value present in the entity. If we are setting a primary key column value, with that primary key column value no row is available then record will be inserted. With that primary key if row is already present then record will be updated.

1. saveAll(Iterable<T> entities): To insert more than one record at time we can go for this method. This method is taking Iterable as a parameter. Iterable is super interface collection interface. If we want to update multiple records also we can use this method.
2. findById(serializable id): To retrieve record from table using primary key we will use this method.
3. findAllById(Iterable<Serializable> ids): To retrieve multiple records based on primary key column values we can use this.
4. findAll(): To retrieve all records from table we can use this.
5. Count(): This method used to get total number of rows available in the table
6. existById(Serializable id): This method is used to verify record presence in table using PK value. If record present it will give true else it will give false.
7. deleteById(serializable id): This method is used to delete record based on PK value.
8. deleteAllById(serializable ids): To delete more than one record based on PK value.
9. deleteAll(): delete all records from the table. .

findById(): Optional<E> findById(): Optional is a predefined public final class in java.util package. This class is used to grab the object. For the given primary key value record may available ,record may not be available that’s why method will get the data in the form of optional. If data is available it will return object else return empty.

We can work with the Spring Data JPA in below 3 ways:

1. By using predefined methods
2. We can generate queries using findBy methods-🡪Spring Dat JPA has the capabability to generate queries during run time based on the method name
3. We can work with custom queries also in Spring Data JPA

How to generate queries by using findBy() methods

In sql query we can retrieve data like this->

Select \* from contacts; (In spring DataJPA we can use (findAll() method)

select \* from contacts where contact\_id=1; (In spring DataJPA we can use (findById() method)

select \* from contacts where contact\_name=’Aswani’;

How to retrieve the data based on non primary key column using SpringDataJPA?

Consider a table which is having number of non primary key columns. How can we fetch the data based on non primary key column value? That’s why Spring DataJpa came with the concept called query generation;

Just we have to tell our requirement in the form of method by following some syntactical rule then based on our method name the Data Jpa will generate the query for us in the runtime.

CrudRepository and JPARepository will support for these 3 techniques

**GENERATORS**

Generators are used to generate value for Primary Key column

There are several predefined generators available to generate value for primary key column.

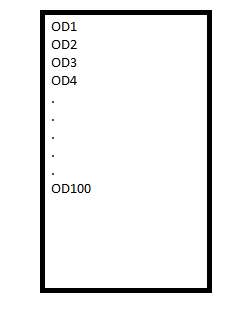
According to client requirement sometime we need to go for Custom Generator

e.g. Emp ids TCS1,TCS2,TCS3……..

Online Order Ids-🡪OD1,OD2,OD3......

To create custome generator we should implement IdentifierGenerator interface.

In that interface we have generate() method. We should write logic in that method to generate PK column value according to our requirement.



If we want to store value in this format our PK column type should be VARCHAR(because it is alphanumeric)

The Order id can be divided into 2 parts.

1->prefix 🡪OD (fixed value)-----------🡪we can keep it as yml or constant

2->suffix 🡪 Number(starts with 1 increment by 1)

* as prefix is fixed we can create a constant for that
* For suffix value we can create a Sequence

Steps to develop Develop JPA application with custom generator:

Create spring starter project with below dependencies

a). Spring-boot-starter-data-jpa

b). Mysql driver

c). Project Lombok

2) Create Entity class using Annotations

3) Create sequence in DB for suffix value generation

4) Create Custom generator class by implementing IdentifierGeneratorInterface

5) Configure custom generator in Entity

6) Create Repository interface by extending JpaRepository

When we create spring boot project automatically application.properties file will be generated. If we want to change into .yml file(right click on application.properties->select convert application.properties into yaml)

Application.yml

Spring:

Datasource:

Driver-class-name: com.mysql.cj.jdbc.Driver

Password:root

Username:root

url: jdbc:mysql://localhost:3306/contactms

jpa:

hibernate:

ddl-auto: update

properties:

hibernate: