=> In Last session we have discussion related to Data JPA

=> Spring Data JPA is used to develop persistence layer in our spring based applications

=> Spring Data JPA provided predefined repositories to simplify CRUD operations implementation in our application

-----------------------------------------------------------------------------------

Building First Application Using Spring Data JPA with Oracle Database

----------------------------------------------------------------------------------

1) Setup Database Server Software

2) Create a table in database

3) Create Spring Starter Application with below dependencies

a) spring-boot-starter-data-jpa

b) oracle-driver

4) Create Entity class and map with database table

5) Create Repository interface

6) Configure data source properties in application.properties file

7) Test the application functionality by calling repository methods

----------------------------------------------------------------------------------

create table users\_dtls

(

user\_id number(10),

user\_name varchar2(20),

user\_phno number(10),

user\_age number(5),

user\_country varchar2(10),

primary key (user\_id)

);

---------------------------------------------------------------------------------

@Entity

@Table(name = "USERS\_DTLS")

public class User {

@Id

@Column(name = "USER\_ID")

private Integer userId;

@Column(name = "USER\_NAME")

private String username;

@Column(name = "USER\_PHNO")

private Long usesrPhno;

@Column(name = "USER\_AGE")

private Integer userAge;

@Column(name = "USER\_COUNTRY")

private String userCountry;

//setter, getters & toString()

}

----------------------------------------------------------------------------------

@Entity -> To represent java class as entity class

@Table -> To map java class with database table name

@Id -> To represent variable which is mapped with primary key column in table

@Column -> To map entity class variable name with table column name

Note: @Table and @Column annotations are optional. If we don't use @Table it will consider class name as table name. If we don't use @Column it will consider variable name as column name.

---------------------------------------------------------------------------------

public interface UserRepository extends CrudRepository<User, Integer>{

}

----------------------------------------------------------------------------------

spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver

spring.datasource.url=jdbc:oracle:thin:@localhost:1521/XE

spring.datasource.username=IES\_DEV

spring.datasource.password=IES\_DEV

----------------------------------------------------------------------------------

@SpringBootApplication

public class Application {

public static void main(String[] args) {

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

UserRepository userRepo = context.getBean(UserRepository.class);

User entity = new User();

entity.setUserId(101);

entity.setUsername("Ram");

entity.setUserAge(45);

entity.setUsesrPhno(1234567890l);

entity.setUserCountry("India");

userRepo.save(entity);

System.out.println("\*\*\*\* Record Inserted \*\*\*\*");

}

}

--------------------------------------------------------------------------------------------------------------

Note: UserRepository is an interface which is extending from CrudRepository. Data JPA will create implementation for our UserRepository using Proxy.

=> When we call context.getBean(UserRepository.class) it will give proxy class object using that proxy class object we can call CrudRepository interface methods to perform Crud operaions

--------------------------------------------------------------------------------------------------------------