03 - Creating Table and Inserting Data

Create a Spring Boot Maven Project. Add Spring Data JPA and PostgreSQL Driver dependency.

👉 **Repository Creation:**

Create a repository interface that extends JpaRepository. This interface automatically provides methods like save(), findAll(), findById(), deleteById(), etc., without needing to write their implementations.

The JpaRepository<Type, ID> takes two parameters:

1. Type: The class (entity) that represents a database table.
2. ID: The data type of the primary key of the table.

```
@Repository
public interface StudentRepo extends JpaRepository<Student, Integer> {

}
```

- Here, the StudentRepo interface is a repository for the Student entity, where the primary key type is Integer.

👉 **Entity Definition:**

Entities represent the data model or table structure in the database. Spring Data JPA uses these entities to map Java objects to database tables.

```
@Component
@Scope("prototype")
@Entity
public class Student {

	@Id
	private int rollNo;
	private String name;
	private int marks;

	public int getRollNo() {
		return rollNo;
	}

	public void setRollNo(int rollNo) {
		this.rollNo = rollNo;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getMarks() {
		return marks;
	}

	public void setMarks(int marks) {

		this.marks = marks;
	}
}
```

- In this, **@Entity** tells Spring Data JPA that this class is linked to a database table, and **@Id** marks rollNo as the primary key of the Student table.

👉 **Spring Boot Application Class:**

It initializes the Spring Application Context and loads all the beans.

- Create a Student object using the Spring context.
- Set values for the Student object.
- Use the StudentRepo to save the Student object into the database.

```
@SpringBootApplication
public class SpringDataJpaExApplication {
	public static void main(String[] args) {
	ApplicationContext context=
	SpringApplication.run(SpringDataJpaExApplication.class, args);
	  Student s1= context.getBean(Student.class);
	  Student s2=context.getBean(Student.class);
	  Student s3=context.getBean(Student.class);
	  StudentRepo repo=context.getBean(StudentRepo.class);

	  s1.setRollNo(101);
	  s1.setName("Navin");
	  s1.setMarks(75);

	  s2.setRollNo(102);
	  s2.setName("Kiran");
	  s2.setMarks(80);

	  s3.setRollNo(103);
	  s3.setName("Harsh");
	  s3.setMarks(70);

	  repo.save(s1);
	  repo.save(s2);
	  repo.save(s3);
	}
}
```

- Here, we retrieve the Student bean and StudentRepo bean from the container, then use repo.save() to persist the data.

👉 **Database Configuration:**

We need to configure in the application.properties file the connection to the database so that Spring Data JPA can automatically create and manage tables, and perform operations.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/telusko

spring.datasource.username=postgres

spring.datasource.password=root

spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true
```

Code Link :
https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/9%20Spring%20DataJPA/9.3%20Creating%20Table%20And%20Inserting%20Data