+++++++++++++
SpringDataJPA
+++++++++++++
  Pre-requisite : CoreJava, JDBC,ORM,SpringCore,SpringJDBC

Working with SpringDataJPA
 a. starters required are
      1. springdatajpa
      2. mysqldriver

+++++++
pom.xml
+++++++

```xml
<dependencies>
      <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
      </dependency>
      <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
      </dependency>
</dependencies>
```

application.properties
++++++++++++++++++++++

```
## MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/pwskillsbatch
spring.datasource.username=root
spring.datasource.password=root123
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

++++++
Entity
++++++

```java
package in.pwskills.nitin.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="CORONA_VACCINE_TAB")
public class CoronaVaccine {

      @Id
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      public Long regNo;

      @Column(length = 20)
      public String name;
```

```
@Column(length = 20)
public String company;

@Column(length = 20)
public String country;
public Double price;
public Integer requiredDoseCount;

public CoronaVaccine() {
      System.out.println("OBJECT CREATED BY FRAMEWORK...");
}

public Long getRegNo() {
      return regNo;
}

public void setRegNo(Long regNo) {
      this.regNo = regNo;
}

public String getName() {
      return name;
}

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

public String getCompany() {
      return company;
}

public void setCompany(String company) {
      this.company = company;
}

public String getCountry() {
      return country;
}

public void setCountry(String country) {
      this.country = country;
}

public Double getPrice() {
      return price;
}

public void setPrice(Double price) {
      this.price = price;
}

public Integer getRequiredDoseCount() {
      return requiredDoseCount;
}

public void setRequiredDoseCount(Integer requiredDoseCount) {
      this.requiredDoseCount = requiredDoseCount;
}
```

```
      @Override
      public String toString() {
            return "CoronaVaccine [regNo=" + regNo + ", name=" + name + ",
company=" + company + ", country=" + country
                        + ", price=" + price + ", requiredDoseCount=" +
requiredDoseCount + "]";
      }

}
```

Create a Repository layer using any of the following interfaces
a. CrudRepository
b. PagingAndSortingRepository
c. JpaRepository

++++++++++++++++
Repository layer
++++++++++++++++

```
public interface ICoronaVaccineRepo extends CrudRepository<CoronaVaccine, Integer>
{

}
```

Note: For this interface implementation class will be given by the runtime environment as "Proxy Class".

Code to perform insert operation
++++++++++++++++++++++++++++++++

```
@Override
public String registerVaccine(CoronaVaccine vaccine) {
      System.out.println("Implementation class is :: " +
repo.getClass().getName());

      CoronaVaccine savedVaccine = null;
      if (vaccine != null)
            savedVaccine = repo.save(vaccine);// method to perform insert is ::
save(Entity)

      return savedVaccine != null ? "vaccine registerd succesfully :: " +
savedVaccine.getRegNo()
                        : "Vaccine registartion failed";
}
```

Output

```
Hibernate:
    insert
    into
        corona_vaccine_tab
        (company, country, name, price, required_dose_count)
    values
        (?, ?, ?, ?, ?)
vaccine registerd succesfully :: 1
```

Code to perform select operation
++++++++++++++++++++++++++++++++

```
@Override
public Iterable<CoronaVaccine> fetchAllDetails() {
      return repo.findAll();
}
```

```
Output
Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_,
        coronavacc0_.company as company2_0_,
        coronavacc0_.country as country3_0_,
        coronavacc0_.name as name4_0_,
        coronavacc0_.price as price5_0_,
        coronavacc0_.required_dose_count as required6_0_
    from
        corona_vaccine_tab coronavacc0_
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
CoronaVaccine [regNo=1, name=covaxin, company=biotech, country=IND, price=2400.0,
requiredDoseCount=2]
CoronaVaccine [regNo=2, name=biper, company=unicorn, country=USA, price=2500.0,
requiredDoseCount=1]
CoronaVaccine [regNo=3, name=serum, company=chn-biotech, country=China,
price=3000.0, requiredDoseCount=2]
CoronaVaccine [regNo=4, name=covaxin, company=sampleDel, country=RSA, price=2500.0,
requiredDoseCount=2]

Code to perform select operation based on id
++++++++++++++++++++++++++++++++++++++++++++
@Override
public Optional<CoronaVaccine> fetchVaccineById(Long id) {
      return repo.findById(id);
}

Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_0_,
        coronavacc0_.company as company2_0_0_,
        coronavacc0_.country as country3_0_0_,
        coronavacc0_.name as name4_0_0_,
        coronavacc0_.price as price5_0_0_,
        coronavacc0_.required_dose_count as required6_0_0_
    from
        corona_vaccine_tab coronavacc0_
    where
        coronavacc0_.reg_no=?
OBJECT CREATED BY FRAMEWORK...
CoronaVaccine [regNo=2, name=biper, company=unicorn, country=USA, price=2500.0,
requiredDoseCount=1]

+++++++++++++++++++++++++++++++++++++++++++++
If record not available for the given id:: 20
++++++++++++++++++++++++++++++++++++++++++++++
Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_0_,
        coronavacc0_.company as company2_0_0_,
        coronavacc0_.country as country3_0_0_,
        coronavacc0_.name as name4_0_0_,
        coronavacc0_.price as price5_0_0_,
        coronavacc0_.required_dose_count as required6_0_0_
    from
        corona_vaccine_tab coronavacc0_
    where
        coronavacc0_.reg_no=?
Record not available for the given id :: 20
```

3. Delete the record on the basis of id

```
@Override
public String removeVaccineById(Long id) {
      Optional<CoronaVaccine> optional = repo.findById(id);
      if (optional.isPresent()) {
            repo.deleteById(id);
            return "record deleted having id ::"+id;
      } else {
            return "record not available for deletion with the id:: "+id;    }
      }

}
```

output

```
Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_0_,
        coronavacc0_.company as company2_0_0_,
        coronavacc0_.country as country3_0_0_,
        coronavacc0_.name as name4_0_0_,
        coronavacc0_.price as price5_0_0_,
        coronavacc0_.required_dose_count as required6_0_0_
    from
        corona_vaccine_tab coronavacc0_
    where
        coronavacc0_.reg_no=?
OBJECT CREATED BY FRAMEWORK...
Hibernate:
    delete
    from
        corona_vaccine_tab
    where
        reg_no=?
record deleted having id ::3
```

Deleting record w.r.t object
++++++++++++++++++++++++++++

```
@Override
public String removeVaccineByObject(CoronaVaccine vaccine) {
      Optional<CoronaVaccine> optional = repo.findById(vaccine.getRegNo());
      if (optional.isEmpty()) {
            return "record not found for deletion";
      } else {
            repo.delete(vaccine);
            return "record deleted with the id:: "+vaccine.getRegNo();
      }
}
```

Output

```
OBJECT CREATED BY FRAMEWORK...
Hibernate:
    select
```

```
        coronavacc0_.reg_no as reg_no1_0_0_,
        coronavacc0_.company as company2_0_0_,
        coronavacc0_.country as country3_0_0_,
        coronavacc0_.name as name4_0_0_,
        coronavacc0_.price as price5_0_0_,
        coronavacc0_.required_dose_count as required6_0_0_
    from
        corona_vaccine_tab coronavacc0_
    where
        coronavacc0_.reg_no=?
record not found for deletion
```

```
++++++++++++++++++++++++++++++++++++++++++
Code to delete all the records in database
++++++++++++++++++++++++++++++++++++++++++
@Override
public String removeAllVaccines() {
      long count = repo.count();
      if (count!=0) {
            repo.deleteAll();
            return "No of records deleted are :: "+count;
      } else {
            return "Table is empty no records to delete...";
      }
}
```

```
Output
Hibernate:
    select
        count(*) as col_0_0_
    from
        corona_vaccine_tab coronavacc0_
Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_,
        coronavacc0_.company as company2_0_,
        coronavacc0_.country as country3_0_,
        coronavacc0_.name as name4_0_,
        coronavacc0_.price as price5_0_,
        coronavacc0_.required_dose_count as required6_0_
    from
        corona_vaccine_tab coronavacc0_
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
Hibernate:
    delete
    from
        corona_vaccine_tab
    where
        reg_no=?
Hibernate:
    delete
    from
        corona_vaccine_tab
    where
        reg_no=?
No of records deleted are :: 2
```

```
Case2 : when no records are available to delete
Output
Hibernate:
    select
        count(*) as col_0_0_
    from
        corona_vaccine_tab coronavacc0_
Table is empty no records to delete...


++++++++++++++++++++++++++++++++
Code to perform update operation
++++++++++++++++++++++++++++++++
CoronaVaccine vaccine = new CoronaVaccine();
vaccine.setRegNo(5L);
vaccine.setCompany("BIOCORN-IND");
vaccine.setPrice(3500.0);

String status = service.registerVaccine(vaccine);
System.out.println(status);

@Override
public String registerVaccine(CoronaVaccine vaccine) {
      System.out.println("Implementation class is :: " +
repo.getClass().getName());

      CoronaVaccine savedVaccine = null;
      if (vaccine != null)
            savedVaccine = repo.save(vaccine);// method to perform insert/update is
:: save(Entity)

      return savedVaccine != null ? "vaccine registerd succesfully :: " +
savedVaccine.getRegNo()
                        : "Vaccine registartion failed";
}

Output
Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_0_,
        coronavacc0_.company as company2_0_0_,
        coronavacc0_.country as country3_0_0_,
        coronavacc0_.name as name4_0_0_,
        coronavacc0_.price as price5_0_0_,
        coronavacc0_.required_dose_count as required6_0_0_
    from
        corona_vaccine_tab coronavacc0_
    where
        coronavacc0_.reg_no=?
OBJECT CREATED BY FRAMEWORK...
Hibernate:
    update
        corona_vaccine_tab
    set
        company=?,
        country=?,
        name=?,
        price=?,
        required_dose_count=?
    where
        reg_no=?
vaccine registerd/updated succesfully :: 5



+++++++++++++++++++++++++++++++++++++++
Working with PagingAndSortingRepository
+++++++++++++++++++++++++++++++++++++++
public interface PagingAndSortingRepository<T, ID> extends CrudRepository<T, ID> {
      Iterable<T> findAll(Sort sort);
      Page<T> findAll(Pageable pageable);
}

ICoronaVaccineRepo.java
++++++++++++++++++++++
public interface ICoronaVaccineRepo extends
PagingAndSortingRepository<CoronaVaccine, Long> {

}

Sorting the records on the basis of fields like name and price
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
@Override
public Iterable<CoronaVaccine> fetchDetails(boolean asc, String... properties) {
      System.out.println(repo.getClass().getName());

      Sort sort = Sort.by(asc ? Direction.ASC : Direction.DESC, properties);
      return repo.findAll(sort);
}
Output
Hibernate:
    select
        coronavacc0_.reg_no as reg_no1_0_,
        coronavacc0_.company as company2_0_,
        coronavacc0_.country as country3_0_,
        coronavacc0_.name as name4_0_,
        coronavacc0_.price as price5_0_,
        coronavacc0_.required_dose_count as required6_0_
    from
        corona_vaccine_tab coronavacc0_
    order by
        coronavacc0_.price asc,
        coronavacc0_.name asc
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
OBJECT CREATED BY FRAMEWORK...
CoronaVaccine [regNo=6, name=covidshield, company=unicorn, country=USA,
price=2000.0, requiredDoseCount=2]
CoronaVaccine [regNo=9, name=nuvaxovid, company=Novavax, country=CHINA,
price=2000.0, requiredDoseCount=3]
CoronaVaccine [regNo=8, name=covaxin, company=Bharat Biotech, country=IND,
price=2500.0, requiredDoseCount=2]
CoronaVaccine [regNo=7, name=serum, company=RSACorn, country=RSA, price=3000.0,
requiredDoseCount=1]
CoronaVaccine [regNo=5, name=covidShield, company=BIOCORN-IND, country=IND,
price=3500.0, requiredDoseCount=1]

+++++++++++++++++++
Pagination concept
++++++++++++++++++
Iterable<CoronaVaccine> records = service.fetchDetailsByPageNo(1,5, false,
"price");
records.forEach(System.out::println);

CoronaMgmtServiceImpl.java
+++++++++++++++++++++++++
@Override
public Iterable<CoronaVaccine> fetchDetailsByPageNo(int pageNo, int pageSize,
boolean asc, String... properties) {
      PageRequest pageable = PageRequest.of(pageNo, pageSize, asc ? Direction.ASC :
Direction.DESC, properties);
      Page<CoronaVaccine> page = repo.findAll(pageable);
      return page.getContent();
}

CoronaVaccine [regNo=5, name=covidShield, company=BIOCORN-IND, country=IND,
price=3500.0, requiredDoseCount=1]
CoronaVaccine [regNo=10, name=Moderna, company=Spikevax, country=RSA, price=3500.0,
requiredDoseCount=2]
CoronaVaccine [regNo=7, name=serum, company=RSACorn, country=RSA, price=3000.0,
requiredDoseCount=1]
CoronaVaccine [regNo=8, name=covaxin, company=Bharat Biotech, country=IND,
price=2500.0, requiredDoseCount=2]
CoronaVaccine [regNo=11, name=Gamalea, company=Sputnik V, country=IND,
price=2500.0, requiredDoseCount=2]
CoronaVaccine [regNo=6, name=covidshield, company=unicorn, country=USA,
price=2000.0, requiredDoseCount=2]
CoronaVaccine [regNo=9, name=nuvaxovid, company=Novavax, country=CHINA,
price=2000.0, requiredDoseCount=3]
```