JPA JPQL Bulk Data Operations

In the previous section, we fetched single column only. Now, we will learn how to handle bulk data and perform corresponding operations.

JPQL Bulk Data Example

In this example, we will take a basic entity class (in this case StudentEntity.java) and perform different operations on it.

- Create an entity class named as StudentEntity.java under com.javatpoint.jpa package.

StudentEntity.java

```
package com.javatpoint.jpa;
import javax.persistence.*;

@Entity
@Table(name="student")
public class StudentEntity {

   @Id
   private int s_id;
   private String s_name;
   private int s_age;

   public StudentEntity(int s_id, String s_name, int s_age) {
      super();
      this.s_id = s_id;
      this.s_name = s_name;
      this.s_age = s_age;
   }

   public StudentEntity() {
      super();
   }
```

```
    public int getS_id() {
        return s_id;
    }

    public void setS_id(int s_id) {
        this.s_id = s_id;
    }

    public String getS_name() {
        return s_name;
    }

    public void setS_name(String s_name) {
        this.s_name = s_name;
    }

    public int getS_age() {
        return s_age;
    }

    public void setS_age(int s_age) {
        this.s_age = s_age;
    }

}
```

- Now, map the entity class and other databases confiuguration in Persistence.xml file.

Persistence.xml

```
<persistence>
<persistence-unit name="Student_details">

    <class>com.javatpoint.jpa.StudentEntity</class>

    <properties>
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/studentdata"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
```

```
        <property name="javax.persistence.jdbc.password" value=""/>
        <property name="eclipselink.logging.level" value="SEVERE"/>
        <property name="eclipselink.ddl-generation" value="create-or-extend-tables"/>
      </properties>

    </persistence-unit>
</persistence>
```

- Now, we can perform any of the following operations on StudentEntity.java class.

JPQL Fetch

Here, we will fetch all the records from the database.

FetchData.java

```
package com.javatpoint.jpa.jpql;
import com.javatpoint.jpa.StudentEntity;
import javax.persistence.*;
import java.util.*;
public class FetchData {

    public static void main( String args[]) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( "Student_details" );
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin( );



        Query query = em.createQuery( "Select s from StudentEntity s ");

        @SuppressWarnings("unchecked")
       List<StudentEntity> list=(List<StudentEntity>)query.getResultList( );

        System.out.print("s_id");
        System.out.print("\t s_name");
       System.out.println("\t s_age");



        for( StudentEntity s:list ){
```

```
            System.out.print( s.getS_id( ));
            System.out.print("\t" +  s.getS_name( ));
           System.out.print("\t" + s.getS_age( ));
           System.out.println();
         }
em.getTransaction().commit();
        em.close();
        emf.close();
    }
}
```

Output:

```
s_id     s_name  s_age
101      Gaurav  24
102      Rahul   22
103      Chris   20
104      Ronit   26
105      Roy     21
```

JPQL Update

Here, we will update the records in database.

UpdateData.java

```
package com.javatpoint.jpa.jpql;
import javax.persistence.*;
public class UpdataData {

    public static void main( String args[]) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( "Student_details" );
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin( );

         Query query = em.createQuery( "update StudentEntity SET s_age=25 where s_id>103");
          query.executeUpdate();
```

```
        em.getTransaction().commit();
        em.close();
        emf.close();
    }
}
```

Output:

After the execution of the program, the following student table generates under MySQL workbench. To fetch data, run **select * from student** in MySQL.

S_ID	S_NAME	S_AGE
101	Gaurav	24
102	Rahul	22
103	Chris	20
104	Ronit	25
105	Roy	25

JPQL Delete

Here, we will delete the particular records from database.

DeleteData.java

```
package com.javatpoint.jpa.jpql;
import javax.persistence.*;
public class DeleteData {

    public static void main( String args[]) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( "Student_details" );
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin( );



        Query query = em.createQuery( "delete from StudentEntity where s_id=102");

        query.executeUpdate();

        em.getTransaction().commit();
```

```
        em.close();
        emf.close();
    }
}
```

Output:

After the execution of the program, the following student table generates under MySQL workbench. To fetch data, run **select * from student** in MySQL.

S_ID	S_NAME	S_AGE
101	Gaurav	24
102	Rahul	22
103	Chris	20

← prev next →