Java Persistence API (JPA) – Full CRUD Example

This example demonstrates the four basic operations (Create, Read, Update, Delete) using JPA with MySQL as the database. Hibernate is used as the JPA provider.

1. Entity Class – Student.java

```
package com.example.jpa;

import javax.persistence.*;

@Entity
@Table(name = "students")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "student_name")
    private String name;

    @Column(name = "email")
    private String email;

    public Student() {}
    public Student(String name, String email) {
        this.name = name;
        this.email = email;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + ", email=" + email + "]";
    }
}
```

2. Main Application – MainApp.java

```
package com.example.jpa;

import javax.persistence.*;
import java.util.List;

public class MainApp {

    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("student_pu"
        EntityManager em = emf.createEntityManager();

        try {
            // CREATE
            em.getTransaction().begin();
            Student s1 = new Student("Ravi Sharma", "ravi@example.com");
            Student s2 = new Student("Anita Verma", "anita@example.com");
            em.persist(s1);
            em.persist(s2);
            em.getTransaction().commit();
            System.out.println("Students added successfully.\n");

            // READ (JPQL)
            System.out.println("--- Student Records ---");
            List<Student> students = em.createQuery("SELECT s FROM Student s", Student
            for (Student s : students) {
                System.out.println(s);
            }

            // UPDATE
            em.getTransaction().begin();
            Student studentToUpdate = em.find(Student.class, 1);
            if (studentToUpdate != null) {
                studentToUpdate.setEmail("ravi_updated@example.com");
                em.merge(studentToUpdate);
                System.out.println("\nUpdated: " + studentToUpdate);
            }
            em.getTransaction().commit();

            // DELETE
            em.getTransaction().begin();
            Student studentToDelete = em.find(Student.class, 2);
            if (studentToDelete != null) {
                em.remove(studentToDelete);
                System.out.println("\nDeleted: " + studentToDelete);
            }
            em.getTransaction().commit();

            // READ AGAIN
            System.out.println("\n--- Students after Update & Delete ---");
            students = em.createQuery("SELECT s FROM Student s", Student.class).getRes
            for (Student s : students) {
                System.out.println(s);
            }

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

3. Configuration File – persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.2">
```

```
    <persistence-unit name="student_pu">
        <class>com.example.jpa.Student</class>
        <properties>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Dr
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:
            <property name="javax.persistence.jdbc.user" value="root"/>
            <property name="javax.persistence.jdbc.password" value="password"/>

            <property name="javax.persistence.schema-generation.database.action" value
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL8Dial
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>
```

4. Explanation:

• CREATE: Adds new student records to the database using persist().
• READ: Fetches all students using a JPQL query (SELECT s FROM Student s).
• UPDATE: Modifies a record by finding the entity and using merge().
• DELETE: Removes a record by using remove() on a managed entity.

Expected Output:

```
Students added successfully.

--- Student Records ---
Student [id=1, name=Ravi Sharma, email=ravi@example.com]
Student [id=2, name=Anita Verma, email=anita@example.com]

Updated: Student [id=1, name=Ravi Sharma, email=ravi_updated@example.com]

Deleted: Student [id=2, name=Anita Verma, email=anita@example.com]

--- Students after Update & Delete ---
Student [id=1, name=Ravi Sharma, email=ravi_updated@example.com]
```