

Query

Configurazione del Progetto

1. Crea il progetto Maven in IntelliJ IDEA

Step:

- IntelliJ IDEA > New Project > Maven
- Nome progetto: jpa-query-demo

```
<dependencies>
  <!-- Hibernate Core -->
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.2.5.Final</version>
  </dependency>

  <!-- PostgreSQL JDBC Driver -->
  <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.7.5</version>
  </dependency>
</dependencies>
```

Struttura delle Classi

Entità principali:

- Utente
- Categoria
- Articolo

Relazioni:

- Utente ha molte Categorie
- Categoria ha molti Articoli

STRUTTURA DEL PROGETTO

```
jpa-query-demo/  
├── src/  
│   ├── main/  
│   │   ├── java/  
│   │   │   ├── com/example/demo/  
│   │   │   │   ├── dao/  
│   │   │   │   │   ├── ArticoloDAO.java  
│   │   │   │   │   ├── CategoriaDAO.java  
│   │   │   │   │   └── UtenteDAO.java  
│   │   │   │   ├── model/  
│   │   │   │   │   ├── Articolo.java  
│   │   │   │   │   ├── Categoria.java  
│   │   │   │   │   └── Utente.java  
│   │   │   │   └── Main.java  
│   │   └── resources/  
└──
```

```
|  
|  
| └─ META-INF/  
|     └─ persistence.xml
```

Step 1: persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.2"  
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"  
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence  
        http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">  
    <persistence-unit name="epicode">  
        <properties>  
            <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/yyyyyyyyyy"/>  
            <property name="javax.persistence.jdbc.user" value="postgres"/>  
            <property name="javax.persistence.jdbc.password" value="xxxxxxx"/>  
            <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>  
            <property name="hibernate.default_schema" value="public"/>  
            <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>  
            <property name="hibernate.hbm2ddl.auto" value="create"/>  
        </properties>  
    </persistence-unit>  
</persistence>
```

Assicurati di sostituire:

- yyyyyyyyyy con il **nome del tuo database**
- xxxxxxxx con la **tua password PostgreSQL**

Step 2: Classi JPA (Utente , Categoria , Articolo)

Utente.java

```
package com.example.demo.model;

import jakarta.persistence.*;
import java.util.*;

@Entity
public class Utente {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;

    private String email;

    // Relazione OneToMany verso Categoria
    @OneToMany(mappedBy = "utente", cascade = CascadeType.ALL)
    private List<Categoria> categorie = new ArrayList<>();

    public Utente() {}

    public Utente(String nome, String email) {
        this.nome = nome;
        this.email = email;
    }

    // Getter e Setter
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
```

```

public String getNome() { return nome; }
public void setNome(String nome) { this.nome = nome; }

public String getEmail() { return email; }
public void setEmail(String email) { this.email = email; }

public List<Categoria> getCategorie() { return categorie; }
public void setCategorie(List<Categoria> categorie) { this.categorie = categorie; }

@Override
public String toString() {
    return "Utente{id=" + id + ", nome='" + nome + "', email='" + email + "'}";
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Utente)) return false;
    Utente utente = (Utente) o;
    return Objects.equals(id, utente.id);
}

@Override
public int hashCode() {
    return Objects.hash(id);
}
}

```

Categoria.java

```

package com.example.demo.model;

import jakarta.persistence.*;

```

```
import java.util.*;

@Entity
@NamedQuery(name = "Categoria.findByNome", query = "SELECT c FROM Categoria c WHERE c.nome = :nome")
public class Categoria {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;

    private String titolo;

    // Relazione ManyToOne verso Utente
    @ManyToOne
    private Utente utente;

    // Relazione OneToMany verso Articolo
    @OneToMany(mappedBy = "categoria", cascade = CascadeType.ALL)
    private List<Articolo> articoli = new ArrayList<>();

    public Categoria() {}

    public Categoria(String nome, String titolo, Utente utente) {
        this.nome = nome;
        this.titolo = titolo;
        this.utente = utente;
    }

    // Getter, Setter, toString, equals, hashCode

    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getNome() { return nome; }
```

```

public void setNome(String nome) { this.nome = nome; }

public String getTitolo() { return titolo; }
public void setTitolo(String titolo) { this.titolo = titolo; }

public Utente getUtente() { return utente; }
public void setUtente(Utente utente) { this.utente = utente; }

public List<Articolo> getArticoli() { return articoli; }
public void setArticoli(List<Articolo> articoli) { this.articoli = articoli; }

@Override
public String toString() {
    return "Categoria{id=" + id + ", nome='" + nome + "', titolo='" + titolo + "'}";
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Categoria)) return false;
    Categoria that = (Categoria) o;
    return Objects.equals(id, that.id);
}

@Override
public int hashCode() {
    return Objects.hash(id);
}
}

```



Articolo.java

```
package com.example.demo.model;
```

```

import jakarta.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.Objects;

@Entity
@NamedQueries({
    @NamedQuery(name = "Articolo.findByCategoria",
        query = "SELECT a FROM Articolo a WHERE a.categoria.nome = :nomeCategoria"),
    @NamedQuery(name = "Articolo.findByPrezzoRange",
        query = "SELECT a FROM Articolo a WHERE a.prezzo BETWEEN :min AND :max")
})
public class Articolo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;

    private BigDecimal prezzo;

    private LocalDate data;

    @ManyToOne
    private Categoria categoria;

    public Articolo() {}

    public Articolo(String nome, BigDecimal prezzo, LocalDate data, Categoria categoria) {
        this.nome = nome;
        this.prezzo = prezzo;
        this.data = data;
        this.categoria = categoria;
    }
}

```



```
// Getter, Setter, toString, equals, hashCode
```

```
public Long getId() { return id; }
```

```
public void setId(Long id) { this.id = id; }
```

```
public String getNome() { return nome; }
```

```
public void setNome(String nome) { this.nome = nome; }
```

```
public BigDecimal getPrezzo() { return prezzo; }
```

```
public void setPrezzo(BigDecimal prezzo) { this.prezzo = prezzo; }
```

```
public LocalDate getData() { return data; }
```

```
public void setData(LocalDate data) { this.data = data; }
```

```
public Categoria getCategoria() { return categoria; }
```

```
public void setCategoria(Categoria categoria) { this.categoria = categoria; }
```

```
@Override
```

```
public String toString() {
```

```
    return "Articolo{id=" + id + ", nome='" + nome + "', prezzo=" + prezzo + ", data=" + data + "}";
```

```
}
```

```
@Override
```

```
public boolean equals(Object o) {
```

```
    if (this == o) return true;
```

```
    if (!(o instanceof Articolo)) return false;
```

```
    Articolo that = (Articolo) o;
```

```
    return Objects.equals(id, that.id);
```

```
}
```

```
@Override
```

```
public int hashCode() {
```

```
    return Objects.hash(id);
```

```
}
```

```
}
```



UtenteDAO.java

```
package com.example.demo.dao;

import com.example.demo.model.Utente;
import jakarta.persistence.EntityManager;

public class UtenteDAO {
    private final EntityManager em;

    public UtenteDAO(EntityManager em) {
        this.em = em;
    }

    public void save(Utente utente) {
        em.persist(utente);
    }

    public Utente findById(Long id) {
        return em.find(Utente.class, id);
    }
}
```



CategoriaDAO.java

```
package com.example.demo.dao;

import com.example.demo.model.Categoria;
import jakarta.persistence.EntityManager;

import java.util.List;
```

```

public class CategoriaDAO {
    private final EntityManager em;

    public CategoriaDAO(EntityManager em) {
        this.em = em;
    }

    public void save(Categoria categoria) {
        em.persist(categoria);
    }

    public List<Categoria> findByNome(String nome) {
        return em.createNamedQuery("Categoria.findByNome", Categoria.class)
            .setParameter("nome", nome)
            .getResultList();
    }

    public List<Object[]> countByUtente() {
        return em.createQuery("SELECT c.utente.nome, COUNT(c.id) FROM Categoria c GROUP BY c.utente.nome", Object[].class)
            .getResultList();
    }
}

```



ArticoloDAO.java

```

package com.example.demo.dao;

import com.example.demo.model.Articolo;
import jakarta.persistence.EntityManager;

import java.math.BigDecimal;
import java.util.List;

```

```
public class ArticoloDAO {
    private final EntityManager em;

    public ArticoloDAO(EntityManager em) {
        this.em = em;
    }

    public void save(Articolo articolo) {
        em.persist(articolo);
    }

    public Articolo findById(Long id) {
        return em.find(Articolo.class, id);
    }

    public List<Articolo> findByCategoria(String nomeCategoria) {
        return em.createNamedQuery("Articolo.findByCategoria", Articolo.class)
            .setParameter("nomeCategoria", nomeCategoria)
            .getResultList();
    }

    public List<Articolo> findByPrezzoRange(BigDecimal min, BigDecimal max) {
        return em.createNamedQuery("Articolo.findByPrezzoRange", Articolo.class)
            .setParameter("min", min)
            .setParameter("max", max)
            .getResultList();
    }

    public List<Articolo> findAllOrderedByPrezzoDesc() {
        return em.createQuery("SELECT a FROM Articolo a ORDER BY a.prezzo DESC", Articolo.class)
            .getResultList();
    }

    public List<Articolo> findWithPagination(int page, int size) {
        return em.createQuery("SELECT a FROM Articolo a", Articolo.class)
            .setFirstResult((page - 1) * size)
```

```
        .setMaxResults(size)
        .getResultList();
    }
}
```



Main.java – Versione Senza Scanner

```
package com.example.demo;

import com.example.demo.dao.*;
import com.example.demo.model.*;
import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.Persistence;

import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.List;

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

        UtenteDAO utenteDAO = new UtenteDAO(em);
        CategoriaDAO categoriaDAO = new CategoriaDAO(em);
        ArticoloDAO articoloDAO = new ArticoloDAO(em);

        try {
            em.getTransaction().begin();

            // CREAZIONE DATI
```

```

Utente u1 = new Utente("Mario", "mario@email.com");
Utente u2 = new Utente("Luca", "luca@email.com");

Categoria c1 = new Categoria("libri", "Narrativa", u1);
Categoria c2 = new Categoria("film", "Cinema", u1);
Categoria c3 = new Categoria("tech", "Informatica", u2);

Articolo a1 = new Articolo("Java Basics", new BigDecimal("19.99"), LocalDate.now().minusDays(10), c1);
Articolo a2 = new Articolo("Spring Boot", new BigDecimal("29.99"), LocalDate.now().minusDays(5), c1);
Articolo a3 = new Articolo("Matrix", new BigDecimal("9.99"), LocalDate.now(), c2);
Articolo a4 = new Articolo("Laptop", new BigDecimal("799.99"), LocalDate.now().minusMonths(1), c3);

utenteDAO.save(u1);
utenteDAO.save(u2);
categoriaDAO.save(c1);
categoriaDAO.save(c2);
categoriaDAO.save(c3);
articoloDAO.save(a1);
articoloDAO.save(a2);
articoloDAO.save(a3);
articoloDAO.save(a4);

em.getTransaction().commit();

// ESECUZIONE QUERY DI TEST
System.out.println("🔪 Articoli nella categoria 'libri:");
articoloDAO.findByCategoria("libri").forEach(System.out::println);

System.out.println("\n🔪 Articoli con prezzo tra 10 e 100:");
articoloDAO.findByPrezzoRange(new BigDecimal("10"), new BigDecimal("100"))
    .forEach(System.out::println);

System.out.println("\n🔪 Articoli ordinati per prezzo decrescente:");
articoloDAO.findAllOrderedByPrezzoDesc().forEach(System.out::println);

System.out.println("\n🔪 Numero categorie per utente:");

```

```
        categoriaDAO.countByUtente().forEach(row ->
            System.out.println("Utente: " + row[0] + " - Categorie: " + row[1]));

        System.out.println("\n📄 Paginazione articoli (pagina 1, size 2):");
        articoloDAO.findWithPagination(1, 2).forEach(System.out::println);

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