Gestire le tabelle relazionali Sql con Spring Data e JPA

Di seguito esaminiamo tre possbili relazioni tra le tabelle SQL, mostrandone la gestione con Spring Data:

- One to One
- One to Many
- Many to Many

Consideriamo tre versioni dello stesso progetto, sperimentando l'uso delle relazioni uno a uno, uno a molti e molti a molti.

Vogliamo gestire una tabella di viaggi cui saranno collegati dei documenti.

Inizialmente si associa un viaggio ad un solo documento.

Successivamente si collega un viaggio a molti documenti.

Infine si collegano molti viaggi ad un documento e un documento potrà essere collegato a molti viaggi.

Il database (le tabelle documenti e viaggi) viene rigenerato ogni volta che si avvia il programma.

Configurazione del progetto

pom

Aggiungere allo starter project le seguenti dipendenze:

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
```

application properties

Il file application properties è lo stesso per tutti e tre i progetti.

```
#Connessione al db
spring.datasource.url=jdbc:mysql://localhost:3306/Its2020?
```

```
createDatabaseIfNotExist=true
spring.datasource.username=Its2020
spring.datasource.password=Its2020

# Drop e Create delle tabelle del db (create, create-drop, validate,
update)
spring.jpa.hibernate.ddl-auto=create

# Hibernate SQL dialect

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

Data access layer

Il data access layer è lo stesso per tutti e tre i progetti.

DocRepo

Repository dei documenti, lo gestiamo con l'interfaccia CrudRepository.

```
package relazioni.repos;

import org.springframework.data.repository.CrudRepository;

import relazioni.entities.Documento;

public interface DocRepo extends CrudRepository<Documento, Integer> {

}
```

ViaggioRepo

Repository dei viaggi, lo gestiamo con l'interfaccia CrudRepository.

```
package relazioni.repos;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import relazioni.entities.Viaggio;


public interface ViaggioRepo extends CrudRepository<Viaggio, Integer> {
```

```
    //derived query
    List<Viaggio> findByDestinazioneAndGenere(String destinazione, String
genere);
}
```

Esempio di Relazione OneToOne

Viaggio OneToOne

```
package relazioni.entities;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "viaggi")
public class Viaggio implements Serializable {

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

    private String destinazione;

    @OneToOne(mappedBy = "viaggio", fetch = FetchType.LAZY, cascade =
CascadeType.ALL)
    private Documento documento;

    public Documento getDocumento() {
        return documento;
    }


    public void setDocumento(Documento documento) {
        this.documento = documento;
    }


    public Viaggio() {}
```

```
    public Viaggio(String destinazione) {
        this.destinazione = destinazione;
    }


    public int getId() {
        return id;
    }

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

    public String getDestinazione() {
        return destinazione;
    }

    public void setDestinazione(String destinazione) {
        this.destinazione = destinazione;
    }

    @Override
    public String toString() {
        return "Viaggio [id=" + id + ", destinazione=" + destinazione +
"]";
    }
}
```

Documento OneToOne

```
package relazioni.entities;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "documenti")
public class Documento implements Serializable {

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

```
    private String path;

    public Documento() {}

    public Documento(String path) {
        this.path = path;
    }

    @OneToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name="viaggi_id", nullable = false)
    private Viaggio viaggio;

    public Viaggio getViaggio() {
        return viaggio;
    }
    public void setViaggio(Viaggio viaggio) {
        this.viaggio = viaggio;
    }


    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public int getId() {
        return id;
    }

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

Classe runner SpringBootApplication

```
package relazioni;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import relazioni.entities.Documento;
import relazioni.entities.Viaggio;
import relazioni.repos.DocRepo;
```

```
import relazioni.repos.ViaggioRepo;

@SpringBootApplication
public class Prj44SpringRelazioniDbApplication {

    public static void main(String[] args) {
        SpringApplication.run(Prj44SpringRelazioniDbApplication.class,
args);
    }

    @Bean
    public CommandLineRunner esempio1(ViaggioRepo viaggioRepo, DocRepo
docRepo) {


        return a -> {

            Viaggio v = new Viaggio("Roma");
            Documento d = new Documento("locandina_roma.pdf");

            v.setDocumento(d);
            d.setViaggio(v);

            viaggioRepo.save(v);



        };

    }
}
```

Esempio Relazione OneToMany

Viaggio OneToMany

```
package relazioni.entities;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
```

```
import javax.persistence.Table;

@Entity
@Table(name = "viaggi")
public class Viaggio implements Serializable {

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

    private String destinazione;

    @OneToMany(mappedBy = "viaggio", fetch = FetchType.LAZY, cascade =
CascadeType.ALL)
    private Set<Documento> documento;


    public Set<Documento> getDocumento() {
        return documento;
    }

    public void setDocumento(Set<Documento> documento) {
        this.documento = documento;
    }

    public Viaggio() {}

    public Viaggio(String destinazione) {
        this.destinazione = destinazione;
    }


    public int getId() {
        return id;
    }

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

    public String getDestinazione() {
        return destinazione;
    }

    public void setDestinazione(String destinazione) {
        this.destinazione = destinazione;
    }

    @Override
    public String toString() {
        return "Viaggio [id=" + id + ", destinazione=" + destinazione +
"]";
    }
```

```
}
```

Documento ManyToOne

Il viaggio è uno a molti ed è proprietario della relazione, conseguentemente il documento deve essere annotato molti a uno.

```
package relazioni.entities;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "documenti")
public class Documento implements Serializable {

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

    private String path;

    public Documento() {}

    public Documento(String path) {
        this.path = path;
    }

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name="viaggi_id", nullable = false)
    private Viaggio viaggio;

    public Viaggio getViaggio() {
        return viaggio;
    }
    public void setViaggio(Viaggio viaggio) {
        this.viaggio = viaggio;
    }
```

```
    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public int getId() {
        return id;
    }

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



}
```

Classe runner SpringBootApplication

```
package relazioni;

import java.util.HashSet;
import java.util.Set;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import relazioni.entities.Documento;
import relazioni.entities.Viaggio;
import relazioni.repos.DocRepo;
import relazioni.repos.ViaggioRepo;

@SpringBootApplication
public class Prj44SpringRelazioniDbApplication {

    public static void main(String[] args) {
        SpringApplication.run(Prj44SpringRelazioniDbApplication.class,
args);
    }

    @Bean
    public CommandLineRunner esempio2(ViaggioRepo viaggioRepo, DocRepo
docRepo) {
```

```
        return a -> {

            Viaggio v = new Viaggio("Roma");
            Documento d = new Documento("locandina_roma.pdf");
            Documento d2 = new Documento("foto_colosseo.jpg");
            d.setViaggio(v);
            d2.setViaggio(v);

            Set<Documento> docs = new HashSet<>();

            docs.add(d);
            docs.add(d2);

            v.setDocumento(docs);
            viaggioRepo.save(v);



        };

    }


}
```

Esempio Relazione ManyToMany

Viaggio ManyToMany

Qui viene definita la tabella relazionale e collegate le chiavi primarie delle tabelle entità.

```
package relazioni.entities;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinColumns;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
```

```
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "viaggi")
public class Viaggio implements Serializable {

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

    private String destinazione;
    private String genere;

    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable(name= "rel_viaggi_docs", joinColumns = {
            @JoinColumn(name="viaggi_id",referencedColumnName =
"id",nullable = false, updatable = false)
    }, inverseJoinColumns = { @JoinColumn(
name="doc_id",referencedColumnName = "id",nullable = false, updatable =
false   )    })
    private Set<Documento> documenti = new HashSet<>();



    public Set<Documento> getDocumenti() {
        return documenti;
    }

    public void setDocumenti(Set<Documento> documenti) {
        this.documenti = documenti;
    }

    public Viaggio() {}

    public Viaggio(String destinazione) {
        this.destinazione = destinazione;
    }


    public int getId() {
        return id;
    }

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

    public String getDestinazione() {
        return destinazione;
    }
```

```
    public void setDestinazione(String destinazione) {
        this.destinazione = destinazione;
    }

    @Override
    public String toString() {
        return "Viaggio [id=" + id + ", destinazione=" + destinazione +
"]";
    }



}
```

Documento ManyToMany

```
package relazioni.entities;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "documenti")
public class Documento implements Serializable {

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

    private String path;

    public Documento() {}

    public Documento(String path) {
        this.path = path;
    }

    @ManyToMany(mappedBy = "documenti" , fetch =  FetchType.LAZY)
```

```
    private Set<Viaggio> viaggi = new HashSet<>();

    public Set<Viaggio> getViaggi() {
        return viaggi;
    }

    public void setViaggi(Set<Viaggio> viaggi) {
        this.viaggi = viaggi;
    }

    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public int getId() {
        return id;
    }

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



}
```

Classe runner SpringBootApplication

```
package relazioni;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import relazioni.entities.Documento;
import relazioni.entities.Viaggio;
import relazioni.repos.DocRepo;
import relazioni.repos.ViaggioRepo;

@SpringBootApplication
public class Prj44SpringRelazioniDbApplication {
```

```
    public static void main(String[] args) {
        SpringApplication.run(Prj44SpringRelazioniDbApplication.class,
args);
    }

    @Bean
    public CommandLineRunner esempio2(ViaggioRepo viaggioRepo, DocRepo
docRepo) {


        return a -> {

            Viaggio v = new Viaggio("Roma");
            Viaggio v2 = new Viaggio("Napoli");

            viaggioRepo.save(v);
            viaggioRepo.save(v2);

            Documento d = new Documento("locandina_roma.pdf");

            Documento d2 = new Documento("foto_vesuvio.jpg");

            Documento d3 = new Documento("vademecum_viaggi.pdf");

            Documento d4 = new Documento("elenco_alberghi.xls");


            docRepo.saveAll(Arrays.asList(d, d2, d3, d4));

            v.getDocumenti().addAll(Arrays.asList(d, d3, d4));
            v2.getDocumenti().addAll(Arrays.asList(d2, d3, d4));


            viaggioRepo.save(v);
            viaggioRepo.save(v2);



        };

    }



}
```