Working with Relationships in Spring Data REST

Working with Relationships in Spring Data REST

One-to-One Relationship

Let's define two entity classes ***A*** and ***B*** having a one-to-one relationship, using the *@OneToOne* annotation. The association is **owned by** the *B Entity* end of the association:


```
@Entity
public class A{
private Long id;
.....
}
```

```
@Entity
public class B{
private Long id;
.......
}
```

Working with Relationships in Spring Data REST

One-to-One Relationship


```
@Entity
public class A{
private Long id;
@OneToOne
@JoinColum(name="b_id")
private B b;
}
```

```
@Entity
public class B{
private Long id;
@OneToOne(mappedBy="b")
private A a;
}
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

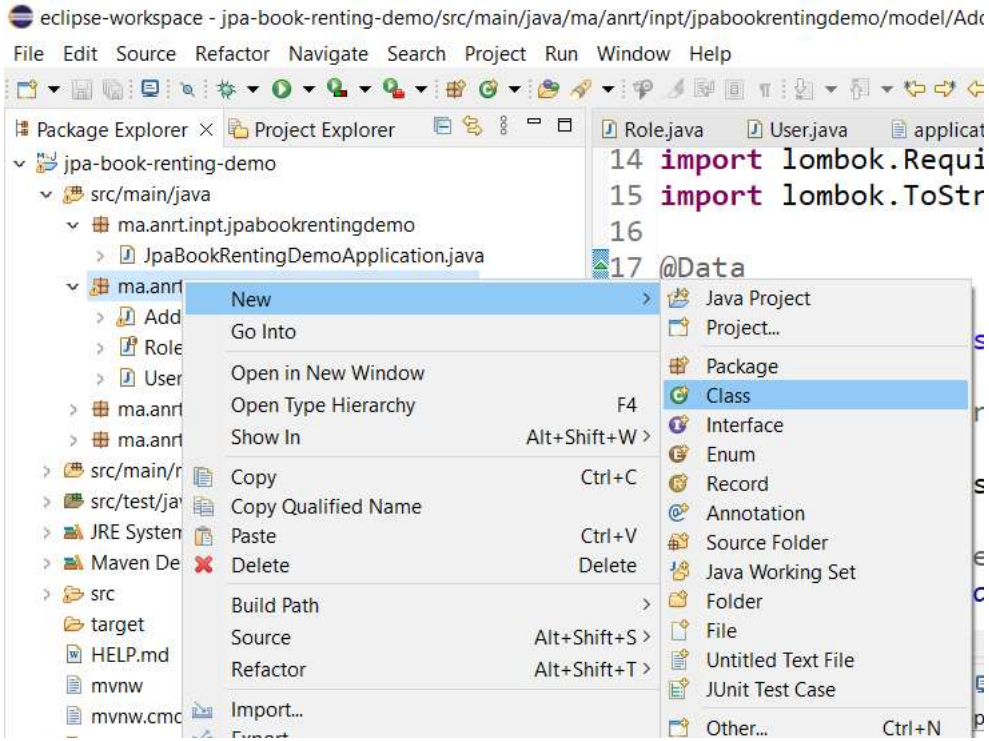

```
@Entity
public class User{
private Long id;
//other attributes
....
@OneToOne
@JoinColum(name="address_id")
private Address address;
}
```

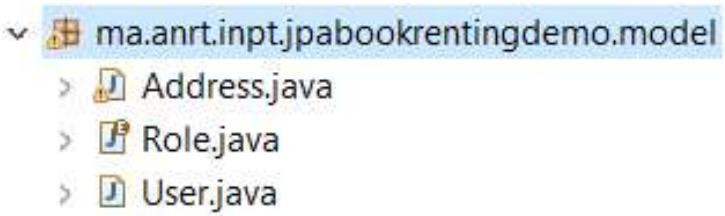
```
@Entity
public class B{
private Long id;
//other attributes
....
@OneToOne(mappedBy="address")
private User user;
}
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Create a new Class “Address” in the model package





Working with Relationships in Spring Data REST

One-to-One Relationship example

Use annotation of JPA and Lombok to define the Address Entity as follow

```
@Data
@Entity
@Table(name="addresses")
@NoArgsConstructor
@RequiredArgsConstructor
@ToString
public class Address {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@NonNull
private String app;
@NonNull
private String im;
@NonNull
private String blv;
@NonNull
private String ville;
}
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Add the Mapping annotations in both User and Address Entities

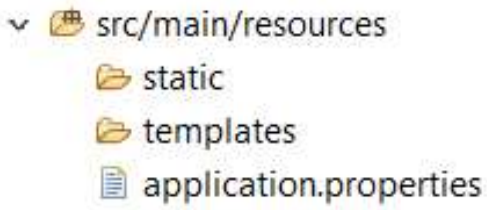
```
@Data
@Entity
@Table(name="users")
…..
public class User {
@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;
……..
//Mapping to address
@OneToOne
@JoinColumn(name="address_id")
private Address address;
}
```

```
@Data
@Entity
@Table(name="addresses")
…..
public class Address {
@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;
……
//link to a user
@OneToOne(mappedBy = "address")
private User user;
}
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Change ddl-auto to create




```
spring.datasource.url=jdbc:postgresql://localhost:5432/db_book
spring.datasource.username=admin
spring.datasource.password=123456
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.show-sql=true
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Run the app and check database in PostgreSQL

Check these SQL statements:

- **A new field address_id has been added to the users table**

```
Hibernate:

    create table addresses (
       id  bigserial not null,
        app varchar(255),
        blv varchar(255),
        im varchar(255),
        ville varchar(255),
        primary key (id)
    )
Hibernate:

    create table users (
       id  bigserial not null,
        first_name varchar(255),
        last_name varchar(255),
        password varchar(255),
        role varchar(255),
        username varchar(100) not null,
        address_id int8,
        primary key (id)
    )
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

The new field “address_id” has been added to the users table, this came from `@JoinColumn(name="address_id")` in the User Entity

Data Output | Explain | Messages | Notifications

	id [PK] bigint	first_name character varying (255)	last_name character varying (255)	password character varying (255)	role character varying (255)	username character varying (100)	address_id bigint
1	1	Abdel	LAM	123	USER	abdel.lam	[null]

Working with Relationships in Spring Data REST

One-to-One Relationship example

```
14 @SpringBootApplication
15 public class JpaBookRentingDemoApplication {
16     @Autowired
17     UserService userService;
18     public static void main(String[] args) {
19         SpringApplication.run(JpaBookRentingDemoApplication.class, args);
20     }
22 CommandLineRunner runner() {
37 }
```

Create a test as following

```
@Bean
 CommandLineRunner runner() {
 return args ->{
 //create an Address
 Address address = new Address("A3","F112","Boulevard Med V","Rabat");
 // create a User
 User user = new User("Abdel","LAM", "abdel.lam","123",Role.USER);
 //set an address to the User, we can create a another contructor that includes
address as an argument
//save User into database
 user.setAddress(address);
 userService.saveUser(user);
// find by username
 User foundUser=userService.findUserByUsername("abdel.lam");
 System.out.println(foundUser);
 };
 }
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

You will get the following error once your run the app

```
java.lang.IllegalStateException: Failed to execute CommandLineRunner
        at org.springframework.boot.SpringApplication.callRunner(SpringApplication.java:770) ~[s
        at org.springframework.boot.SpringApplication.callRunners(SpringApplication.java:751) ~[
        at org.springframework.boot.SpringApplication.run(SpringApplication.java:309) ~[spring-b
        at org.springframework.boot.SpringApplication.run(SpringApplication.java:1301) ~[spring-
```

Caused by: org.springframework.dao.InvalidDataAccessApiUsageException: org.hibernate.TransientPropertyValueException: **object references an unsaved transient instance** - save the transient instance before flushing : ma.anrt.inpt.jpabookrentingdemo.model.User.address -> ma.anrt.inpt.jpabookrentingdemo.model.**Address;** nested exception is java.lang.IllegalStateException: org.hibernate.TransientPropertyValueException: object references an unsaved transient instance - save the transient instance before flushing : ma.anrt.inpt.jpabookrentingdemo.model.User.address -> ma.anrt.inpt.jpabookrentingdemo.model.Address
at org.springframework.orm.jpa.EntityManagerFactoryUtils.convertJpaAccessExceptionIfPossible(EntityManagerFactoryUtils.java:371) ~[spring-orm-5.3.14.jar:5.3.14]
at org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:235) ~[spring-orm-5.3.14.jar:5.3.14]

Working with Relationships in Spring Data REST

One-to-One Relationship example

2 Fixes:

- Save Address first before saving User through AddressRepository
- Tell Hibernate to save them for you

```
@Bean
 CommandLineRunner runner() {
 return args ->{
 //create an Address
 Address address = new Address("A3","F112","Boulevard Med V","Rabat");
 // create a User
 User user = new User("Abdel","LAM", "abdel.lam","123",Role.USER);
 //set an address to the User, we can create a another contructor that includes address as an argument
//save User into database
 user.setAddress(address);
 userService.saveUser(user);
// find by username
 User foundUser=userService.findUserByUsername("abdel.lam");
 System.out.println(foundUser);
 };
 }
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

2nd Fix:

- Tell Hibernate to save them for you

```
@Entity
public class User {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@NonNull
.................
//Mapping to address
@OneToOne(cascade = CascadeType.PERSIST) // or cascade = CascadeType.ALL
@JoinColumn(name="address_id")
private Address address;
}
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Run & Test: you will get a null error

```
java.lang.StackOverflowError: null
at java.base/java.lang.AbstractStringBuilder.<init>(AbstractStringBuilder.java:112) ~[na:na]
at java.base/java.lang.StringBuilder.<init>(StringBuilder.java:127) ~[na:na]
at ma.anrt.inpt.jpabookrentingdemo.model.Address.toString(Address.java:22) ~[classes/:na]
at java.base/java.lang.String.valueOf(String.java:4215) ~[na:na]
at java.base/java.lang.StringBuilder.append(StringBuilder.java:169) ~[na:na]
at ma.anrt.inpt.jpabookrentingdemo.model.User.toString(User.java:26) ~[classes/:na]
at java.base/java.lang.String.valueOf(String.java:4215) ~[na:na]
at java.base/java.lang.StringBuilder.append(StringBuilder.java:169) ~[na:na]
at ma.anrt.inpt.jpabookrentingdemo.model.Address.toString(Address.java:22) ~[classes/:na]
at java.base/java.lang.String.valueOf(String.java:4215) ~[na:na]
at java.base/java.lang.StringBuilder.append(StringBuilder.java:169) ~[na:na]
at ma.anrt.inpt.jpabookrentingdemo.model.User.toString(User.java:26) ~[classes/:na]
at java.base/java.lang.String.valueOf(String.java:4215) ~[na:na]
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Check whether the address instance was persisted in Database: **working good**

Data Output | Explain | Messages | Notifications

	id [PK] bigint	app character varying (255)	blv character varying (255)	im character varying (255)	ville character varying (255)
1	1	A3	Boulevard Med V	F112	Rabat

Data Output | Explain | Messages | Notifications

	id [PK] bigint	first_name character varying (255)	last_name character varying (255)	password character varying (255)	role character varying (255)	username character varying (100)	address_id bigint
1	1	Abdel	LAM	123	USER	abdel.lam	1

Working with Relationships in Spring Data REST

One-to-One Relationship example

Here is the problem

```
@Bean
 CommandLineRunner runner() {
 return args ->{
 //create an Address
 Address address = new Address("A3","F112","Boulevard Med V","Rabat");
 // create a User
 User user = new User("Abdel","LAM", "abdel.lam","123",Role.USER);
 //set an address to the User, we can create a another contructor that includes address as an argument
//save User into database
 user.setAddress(address);
 userService.saveUser(user);
// find by username
 User foundUser=userService.findUserByUsername("abdel.lam");
 System.out.println(foundUser);
 };
 }
```

foundUser instance refers to an address instance which is null, caused by Lombok

Working with Relationships in Spring Data REST

One-to-One Relationship example

Similar problem

I assume the `@ToString` annotation tells some tool you're using (Lombok?) to generate a toString method that prints the values of all the fields. Each of the classes refer to the other: Product has a Categorie and Categorie has a list of Product instances. So when the toString implementation prints a Categorie, it calls toString on each Product, which then calls toString on its Categorie, etc. Since Product presumably refers to a Categorie which includes that Product in its products list, the toString calls bounce back and forth until the stack overflows. The solution is to avoid printing either Categorie,products or Product.categorie from the toString method. If you're using Lombok, try annotating Categorie.products with `@ToString.Exclude`.

Working with Relationships in Spring Data REST

One-to-One Relationship example

Fixing the null the problem

```
@Entity
……
public class User {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
…….
//Mapping to address
@OneToOne(cascade = CascadeType.PERSIST) // , fetch = FetchType.EAGER
@JoinColumn(name="address_id")
@ToString.Exclude
private Address address;
}
```

```
@Bean
 CommandLineRunner runner() {
 return args ->{
 //create an Address
……. foundUser=userService.findUserByUsername("abdel.lam");
 System.out.println(foundUser);
 System.out.println(foundUser.getAddress());
 };
 }
```

Working with Relationships in Spring Data REST

One-to-One Relationship example

Everything is working fine. You will get the following in the console Notice that the Address displays also the mapped user, this was the reason for the previous issue

```
User(id=1, firstName=Abdel, lastName=LAM, username=abdel.lam, password=123, role=USER)
Address(id=1, app=A3, im=F112, blv=Boulevard Med V, ville=Rabat, user=User(id=1,
firstName=Abdel, lastName=LAM, username=abdel.lam, password=123, role=USER))
```

Create a REST API controller

Create a REST API controller

Add the following service methods:

- addUser
- updateUser
- seachUserByUsername
- deleteUserById

Create a REST API controller