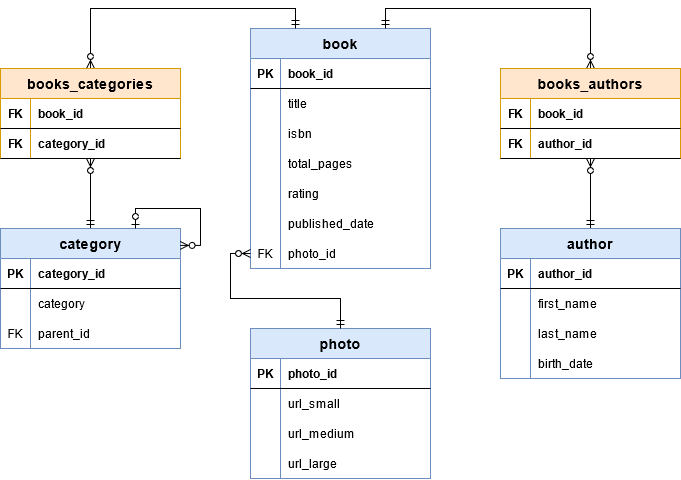
1. **Problem Statement**

**Database Relationships In Spring**

In this we are showing Database relationships (One-to-Many, Many-to-Many, One-to-One) with examples in Spring Data JPA.



1. There is a Many-to-Many relationship between books and categories. A book can have more than one category, and a category can have more than one book.
2. photo table attached to the books, the relationship here will be One-to-One.
3. First we will create a Spring project using Spring Initializr. Then we will establish the database connection.
4. Next, we will look at these relationships in more detail by applying the diagram through the code.

**Steps: -**

1. Installation
2. Implementation
3. For the construction and Getter & Setter methods, we will use Lombok library. Data annotation contains ToString, Getter, Setter, EqualsAndHashCode and RequiredArgsConstructor.
4. OneToOne Relation

Relationship between Book and Photo is an example to OneToOne relation.

For that relation, we need to create a foreign key for photo\_id parameter in Book table. OneToOne annotation will create this relation for us. We should add this annotation to both entity classes that’s because this is a bidirectional relationship.

1. we will use JoinColumn annotation in Book class. On the Photo class, simply put OneToOne annotation and fill mappedBy field by variable name on the owning side. In this example, this value will be ‘photo’.
2. OneToMany Relation

Relationship between parent category and children categories in Category table is an example to OneToMany relation.

In this case, both parent and children tables are same, we will add both OneToMany and ManyToOne annotations to same class. We should join a column for parent\_id.

1. ManyToMany Relation

Relationship between book and category is an example to ManyToMany relation.

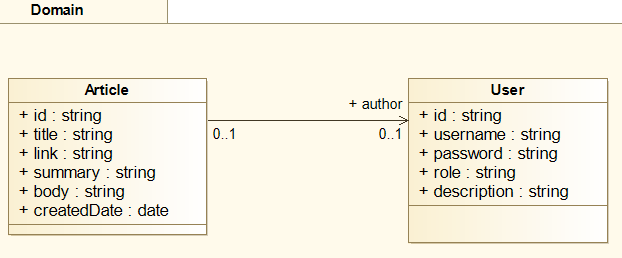
In this relation, we need to create a new table to handle ManyToMany relationship. This new table will hold foreign keys for both book\_id and category\_id fields. We will follow the relationship between books and categories by this table.

1. **Problem Statement**

**Class diagram for the domain model**

Since the domain model is the most important component of an application, this section will design it first.

The following is the simple class diagram for this web application:

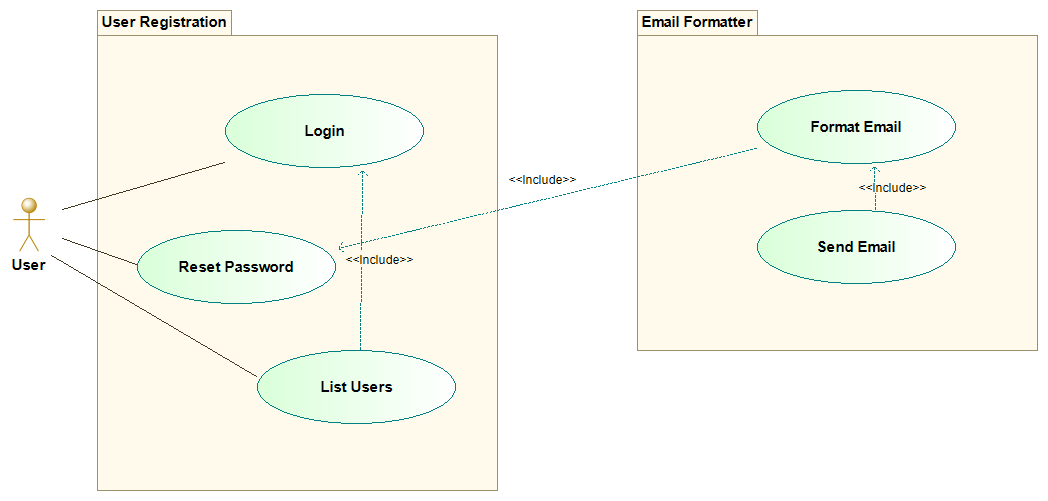


There are two main domain models, as shown in the preceding diagram. They are as follows:

1. **Article:** This is the main domain model, which will store the actual article title, body, link, summary, author, created date, and so on
2. **User:** This domain model will store the username, password, and role of a registered user
3. **Problem Statement**

**The use case diagram**

The following use case diagram shows the requirement for Email Formatter:



The actor is the **User**of the Tweety. It has the following use cases:

1. **Login**: This use case is required to authenticate users so that each user can be uniquely identified to allow only authenticated users to perform actions.
2. **List Users**: This use case is where a user can list all the usernames of available users. It requires the user to be authenticated.
3. **Reset Password**: This use case is where a user can request to reset his/her password.
4. **Format Email**: This use case is where, based on a **Reset Password** request, an email will be formatted and filled with ...