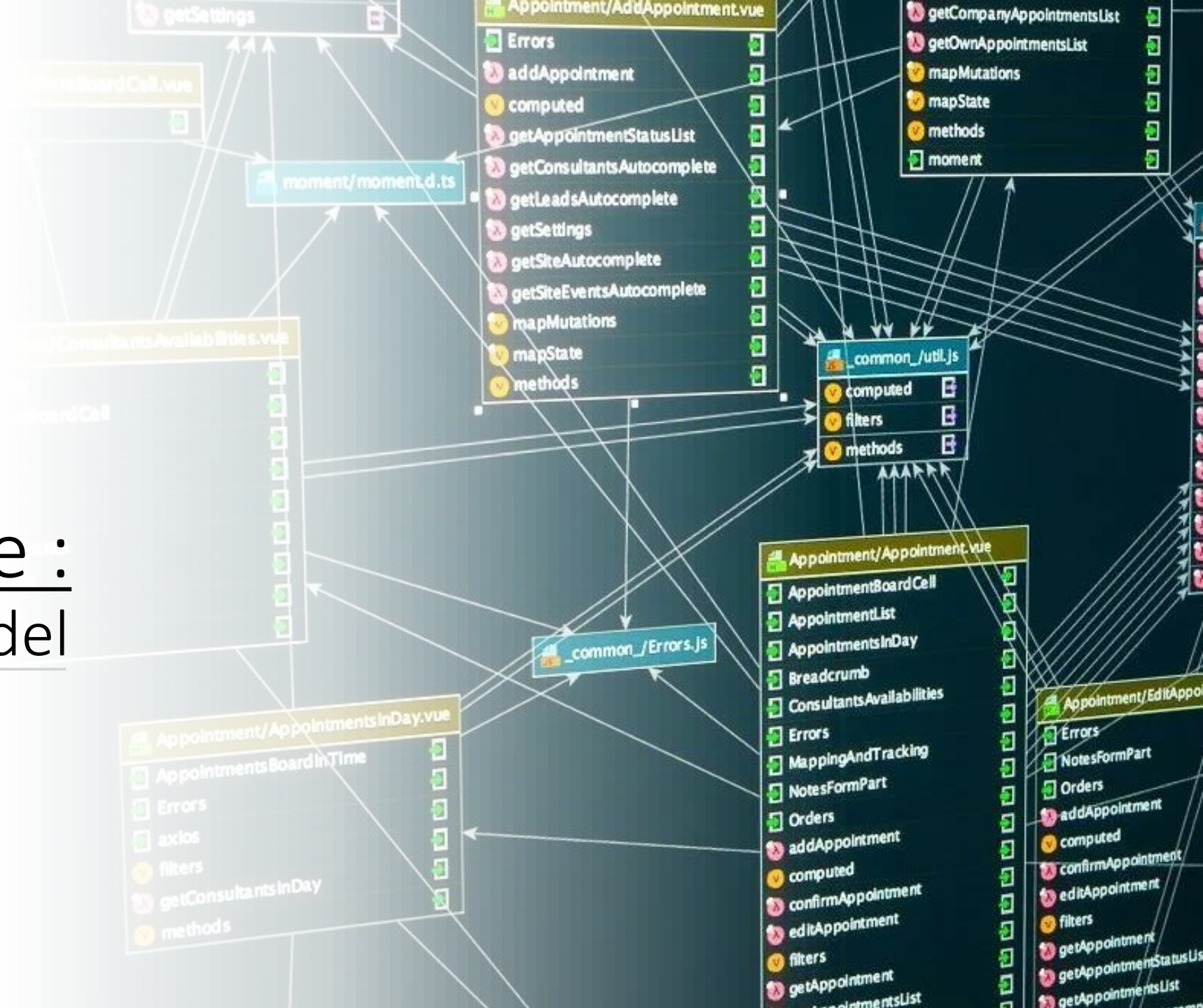
JPA & Hibernate :

Entity Lifecycle Model



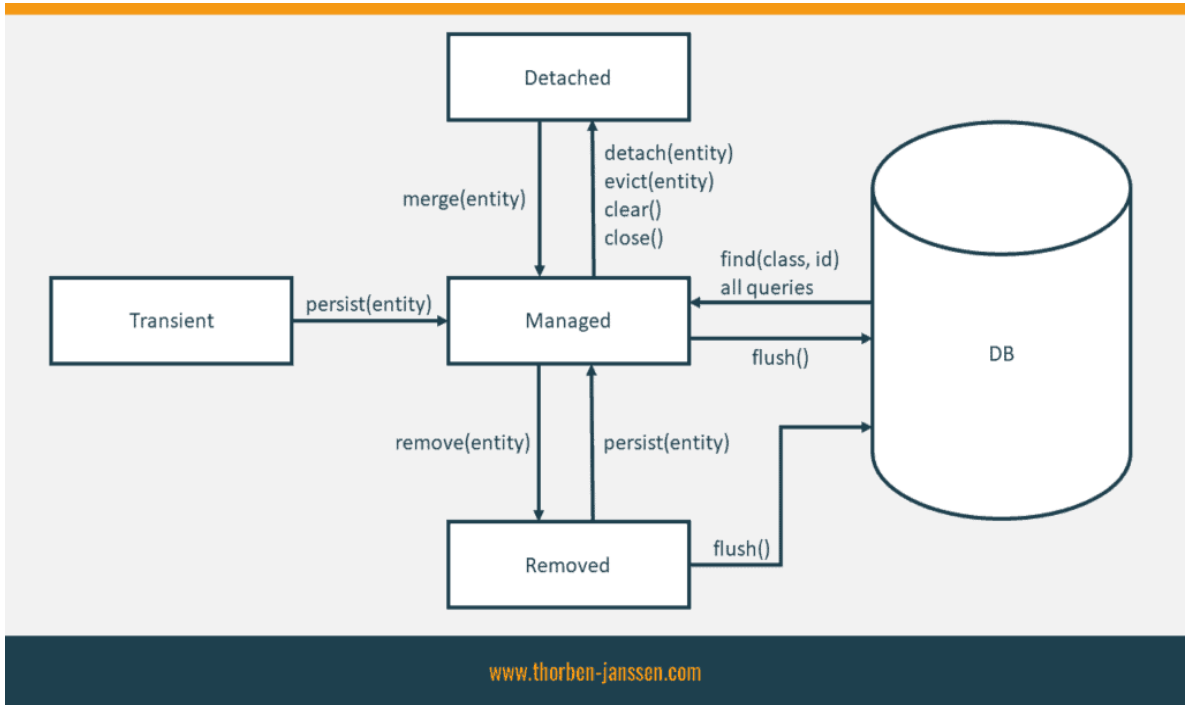
What is an entity?

- In the context of Java Persistence API (JPA), an entity is a lightweight Java object that represents **a persistent object in a database**. An entity typically corresponds to a row in a database table, and can have attributes that map to columns in the table.
- To create an entity in JPA, a Java class is annotated with the **@Entity** annotation, which tells the persistence provider that the class should be treated as an entity. The class may also have additional annotations that provide more information about how it maps to the database, such as @Table to specify the name of the database table, and @Column to specify the name of the column.

```java
@Entity
public class User {

    @Id
    @GeneratedValue
    private int id;

    private String userName;
    private String firstName;
    private String lastName;
    @Transient
    private String fullName;

    // Standard getters/setters
}
```

Java's transient keyword is used to denote that a field is not to be serialized, whereas JPA's @Transient annotation is used to indicate that a field is not to be persisted in the database.

4 Life Cycle States of an Entity



1. Transient

The lifecycle state of a newly instantiated entity object is called transient. The entity hasn't been persisted yet, so it doesn't represent any database record.

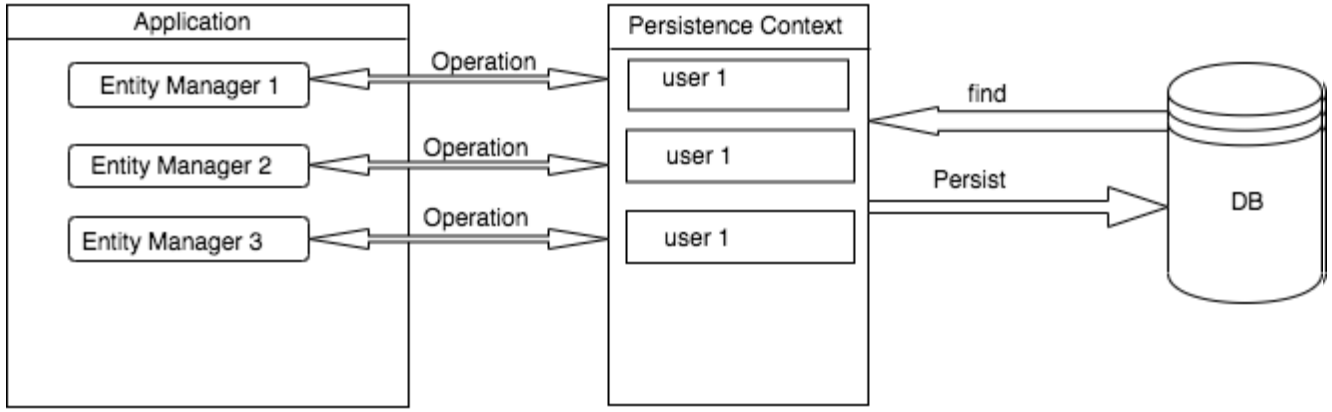
Your ***persistence context**** doesn't know about your newly instantiate object. Because of that, it doesn't automatically perform any saving method or tracks any changes. As long as your entity object is in the lifecycle state transient, you can think of it as a basic Java object without any connection to the database and any JPA-specific functionality.

That changes when you provide it to the ***EntityManager.find**** method. The entity object then changes its lifecycle state to managed and gets attached to the current persistence context.


```
1  Author author = new Author();
2  author.setFirstName("Thorben");
3  author.setLastName("Janssen");
```

JPA/Hibernate Persistence Context

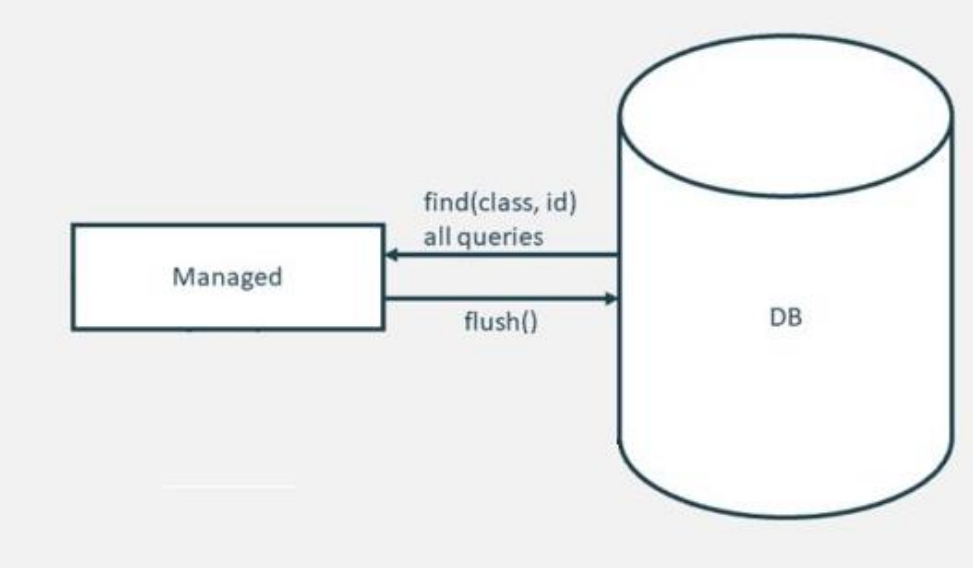
In Java Persistence API (JPA), the persistence context refers to a group of entity instances managed by the EntityManager interface. This interface is responsible for coordinating the persistence of entity objects between a Java application and a persistence provider. Whenever an entity is retrieved from the database, it is added to the persistence context and its state is tracked. The persistence context also handles relationships between entities, ensuring that modifications made to one entity are reflected in associated entities.



Extended Persistence Context

2. Managed

Once an object is persisted in the database, it enters the managed stage. In this stage, the object is associated with a persistent identity and any changes made to the object will be tracked and persisted to the database when a transaction is committed.

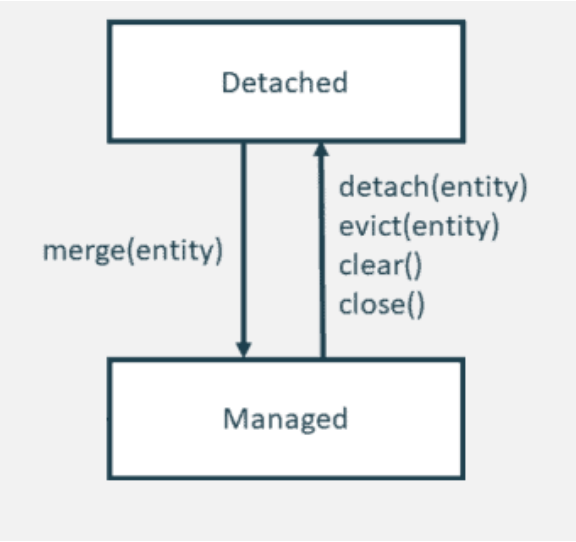



```
1  Author author = new Author();
2  author.setFirstName("Thorben");
3  author.setLastName("Janssen");
4  em.persist(author);
```

3. Detached

When a managed object is no longer attached to the current persistence context, it enters the detached stage. In this stage, any changes made to the object will not be tracked by the persistence context and will not be persisted to the database.

The detached stage provides a way to temporarily disconnect an entity from the persistence context, without losing its persistent identity, and to later re-attach the entity and apply any changes made while it was detached.

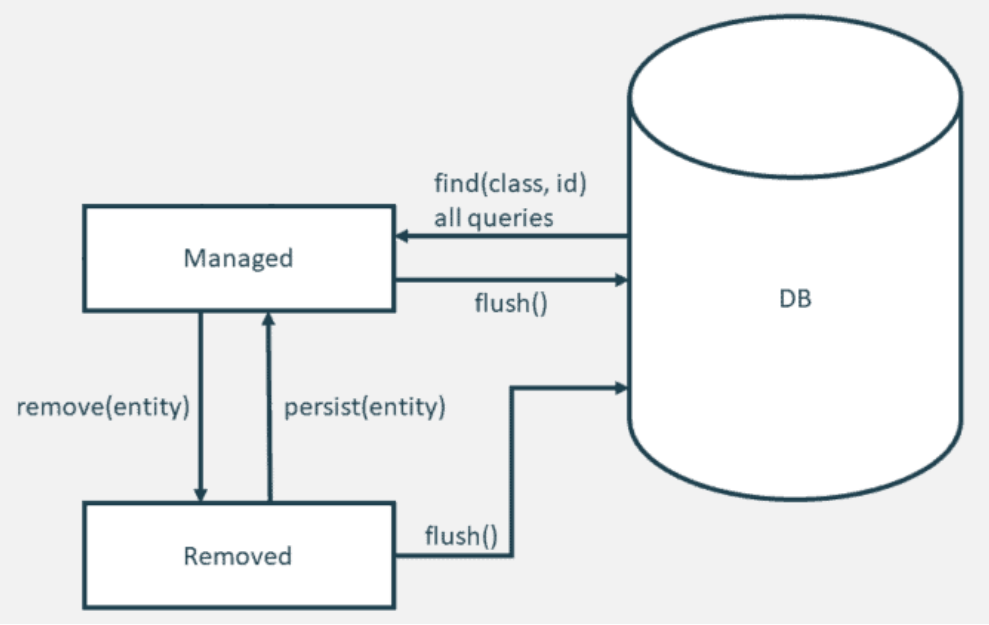



```
1 | em.detach(author);
```

4. Removed

This is the final stage in the entity lifecycle model where the object is marked for deletion. In this stage, the object is associated with a persistent identity but will be deleted from the database when a transaction is committed.

(When you call the remove method on your EntityManager, the mapped database record doesn't get removed immediately. The entity object only changes its lifecycle state to removed. During the next flush operation, Hibernate will generate a delete statement to remove the record from the database table)




```
1 | em.remove(author);
```

Methods

Operation	Spring Data JPA Method	JPA Method
Create	JpaRepository.save(entity)	EntityManager.persist(entity)
Read/Retrieve	JpaRepository.findById(id)	EntityManager.find(entityClass, primaryKey)
Update	JpaRepository.save(entity)	EntityManager.merge(entity)
Delete	JpaRepository.delete(entity)	EntityManager.remove(entity)

Thanks for listening