

Le cycle de vie des entités JPA

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

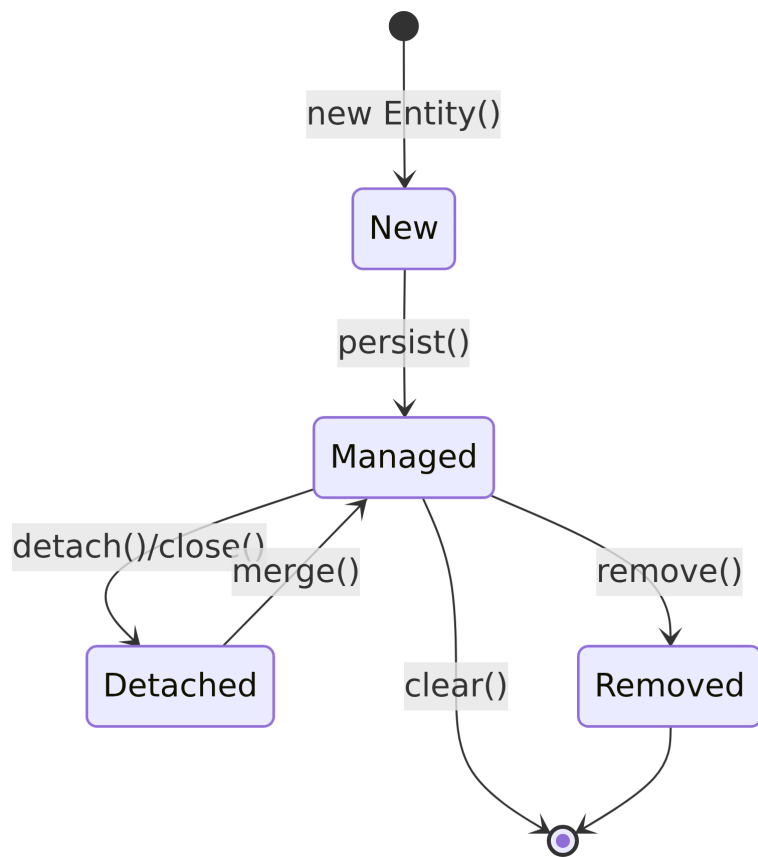
2026-10-14

Table des matières

1 Cycle de vie d'une entité	1
2 Etats d'une entité	1
3 Callbacks du cycle de vie	2
4 Callbacks sur l'entité	2
4.1 Exemple : Listener	2
4.2 Exemple : Entité avec Listener	2

1 Cycle de vie d'une entité

Le cycle de vie d'une entité JPA définit les différents états possibles d'une instance (**New**, **Managed**, **Detached**, **Removed**) et les transitions entre ces états via les opérations de l'EntityManager (**persist()**, **merge()**, **remove()**, **detach()**), permettant ainsi le suivi et la gestion de la synchronisation entre les objets Java en mémoire et leur représentation en base de données.



2 Etats d'une entité

- **New (Transient)**
 - Nouvelle instance Java
 - Inconnue de JPA
 - Pas d'identifiant assigné
- **Managed (Persistent)**
 - Gérée par EntityManager

- Synchronisée avec la BD
- Modifications automatiquement persistées
- Detached
 - Plus gérée par EntityManager
 - Modifications non suivies
 - Nécessite merge() pour synchroniser
- Removed
 - Marquée pour suppression
 - Sera supprimée en BD

3 Callbacks du cycle de vie

Comme le cycle de vie des entités géré par JPA, des annotations spécifiques sont proposées pour exécuter des méthodes de Callback d'une entité à des moments spécifiques :

- Persistence
 - @PrePersist: avant persist()
 - @PostPersist: après persist()
- Update
 - @PreUpdate: avant modification
 - @PostUpdate: après modification
- Remove
 - @PreRemove: avant suppression
 - @PostRemove: après suppression
- Load
 - @PostLoad: après chargement

4 Callbacks sur l'entité

- Annotations sur méthodes de l'entité
- Exécutées automatiquement lors des transitions

```
@Entity
public class Customer {
    @PrePersist
    void onCreate() {
        this.createdAt =
        LocalDateTime.now();
    }
}
```

4.1 Exemple : Listener

Définir un ou des **Listener** : des classes dédiées à la réaction aux événements du cycle de vie d'une entité.

```
@Slf4j
public class EntityMonitorListener {
    @PrePersist
    @PreUpdate
    @PreRemove

    private void
    beforeAnyUpdate(Object entity) {
        log.info("-->about to change
        in DB: " + entity);
    }

    @PostPersist
    @PostUpdate
    @PostRemove
    private void afterAnyUpdate(Object
    entity) {
        log.info("--> changed in DB"
        + entity);
    }

    @PostLoad
    private void afterLoad(Object
    entity) {
        log.info("--> loaded from DB:
        " + entity);
    }
}
```

4.2 Exemple : Entité avec Listener

Les associer à une ou plusieurs entités (@EntityListeners).

```
@Entity
@Table(name = "CUSTOMER_BIS")
@EntityListeners(EntityMonitorListener.class)
@Getter
@Setter
@ToString
```

```
@RequiredArgsConstructor(staticName = "of")
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@Slf4j
public class CustomerWithListener {
    @Id
    @GeneratedValue
    private long id;

    @NonNull
    private String name;
}
```