9 GENNAIO 2020

HIBERNATE

APPROFONDIMENTI SU HIBERNATE

ROBERTO AMATO

Sommario

HIBERNATE

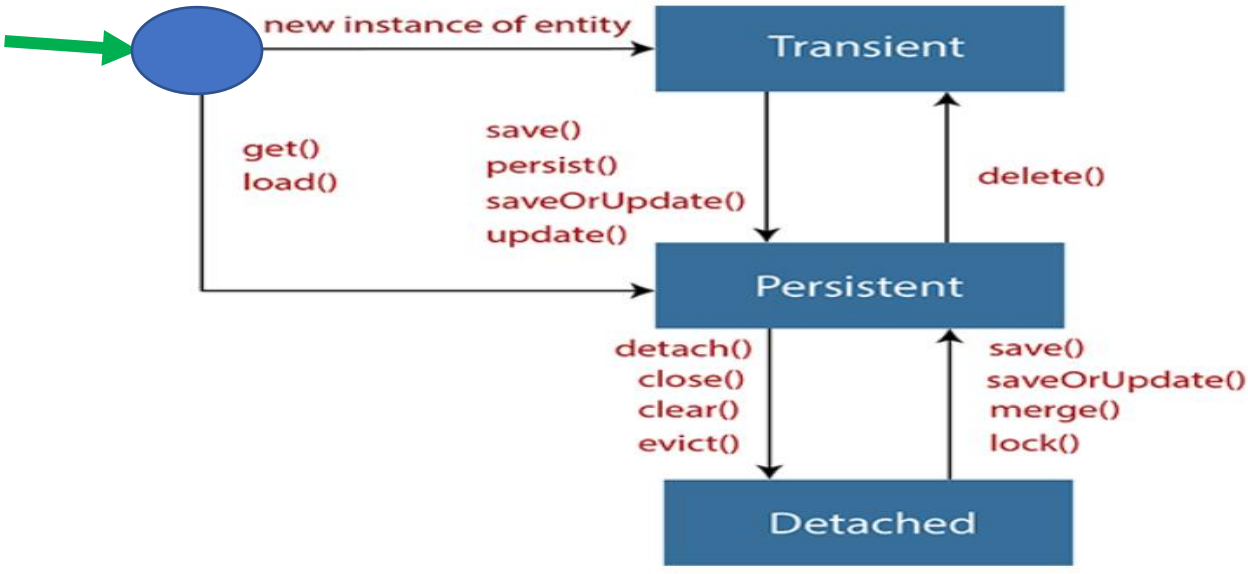
Ciclo di vita di una Entity

Ogni entity (classe Java secondo lo standard POJO che viene mappata nella corrispondente tabella del database) ha un proprio ciclo di vita all'interno dell'ecosistema Hibernate.

Una entity in Hibernate durante il suo ciclo di vita può assumere i seguenti 3 stati:

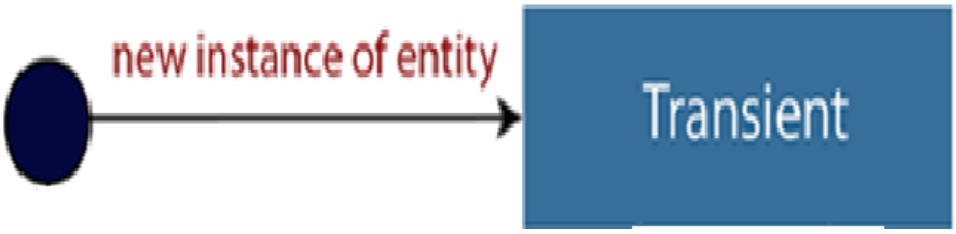
- **Transient** (transitorio)
- **Persistent**/Attached/ (Persistente o Attacato)
- **Detached** (Distaccato)

Ciclo di vita di una entity può essere facilmente modellato da un automa a stati finiti come il seguente:



- *STATO TRANSIENT (TRANSITORIO)*

Quando creiamo una istanza di una entity (ovvero della classe POJO) che mappa l'entità del database, questa entra nello stato Transient ovvero tansitorio, nel senso che l'oggetto della classe POJO appena istanziato, da ora potrebbe essere persistito (ovvero salvato) e quindi potrebbe transitare verso il database.

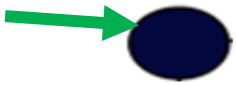


Esempio:

Supponiamo di avere la seguente classe entity di nome Utente:

```
/*
 *
 * Classe Utente secondo lo standard POJO.
 * Questa classe mappa la tabella del database che si chiama "TABELLA_UTENTE"
 *
 */
@Entity
@Table(name="TABELLA_UTENTE")
public class Utente{

    @Id
    @Column(name="ID")
    private Long id;

    @Column(name="USERNAME")
    private String username

    @Column(name="PASSWORD")
    private String password

    public Utente(){
        super();
    }

    // metodi get e set
    ....
    ....
    ....

}
```

Fintanto che non viene istanziato un oggetto delle classe Utente, l'oggetto non esiste e quindi non si trova in nessuno stato, che noi per semplicità lo abbiamo indicato con uno "stato" fittizio senza nome.

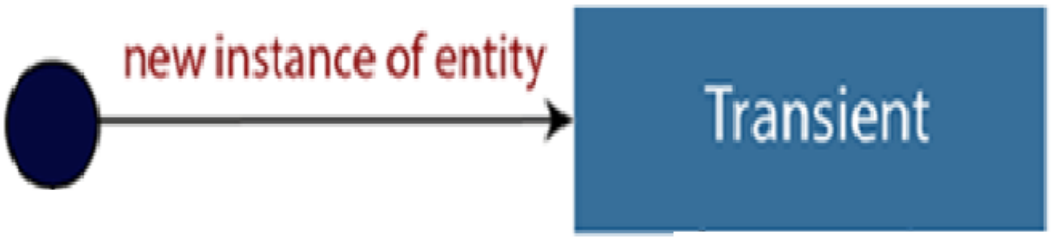


Non appena viene istanziato un oggetto della classe Utente (ovvero della entity Utente), l'oggetto si troverà nello stato di Transient, ovvero transitorio. Nel senso che da ora è possibile persistere tale oggetto sul database.

Esempio :

```
Utente user =  new Utente();
```

L'oggetto Utente dopo la precedente istruzione di istanziazione dell'oggetto si troverà nello stato Transient.



- *STATO PERSISTENT (PERSISTENTE)*

Un oggetto dallo stato Transient (anche detto attached, ovvero attaccato ad una sessione)può passare nello stato Persistent (ovvero persistente nel database) quando viene salvato.

Quindi quando vengono invocati i metodi della classe Session:

- save()
- persist()
- update()
- saveOrUpdate()

Da qui in poi l'oggetto rappresenta un record della tabella nel database in quando è stato persistito (salvato)

Esempio:

```
session.save(user);

oppure

session.persit(user);

oppure

session.update(user);

oppure

session.saveOrUpadat(user)
```

A fronte dell'esecuzione di uno dei due precedenti metodi, l'oggetto passa dallo stato Transient, allo stato Persistent.

La figura seguente mostra in modo chiaro il passaggio di stato a causa dell'invocazione di uno dei 2 metodi.

Notiamo che questo non è l'unico modo che ha un ogetto per arrivare nello stato Persistent.

In effetti, un altro modo consiste senza passare per lo stato Transitent a seguito dell'invocazione di uno dei seguenti metodi della classe Session:

- get()
- load()

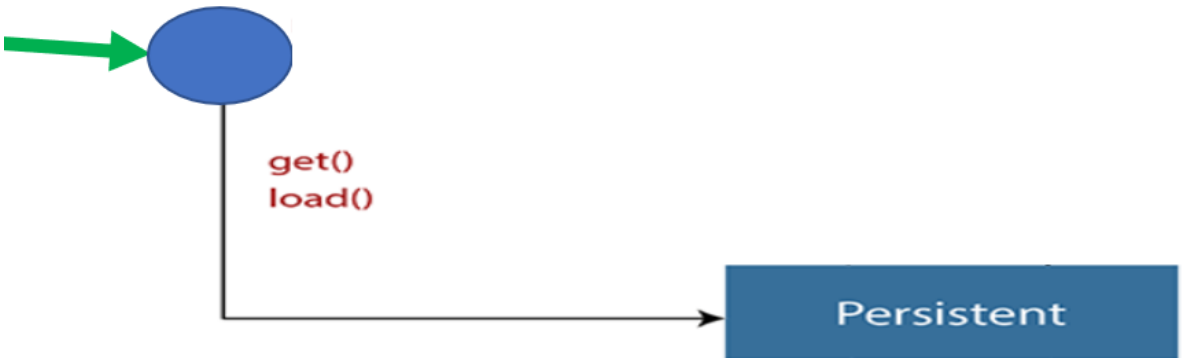
Esempio:

```
session.load(user);

session.get(user);
```

A fronte dell'esecuzione di uno dei due precedenti metodi, l'oggetto passa direttamente allo stato Persistent senza passare per lo stato Transient.

Questo significa che senza istanziare un oggetto (ricordiamo che un oggetto passa nello stato di Transient a seguito della sua instanziazione), possiamo porlo direttamente allo stato di Persistent grazie ai metodi load e

La figura seguente mostra in modo chiaro il passaggio di stato a causa dell'invocazione di uno dei 2 metodi.



- *STATO DETACHED (DISTACCATO)*

Innanzitutto perché lo stato Detached, un oggetto è detached da cosa?

La risposta è : stato detached perché l'oggetto può essere distaccato (detached) da una sessione con il database.

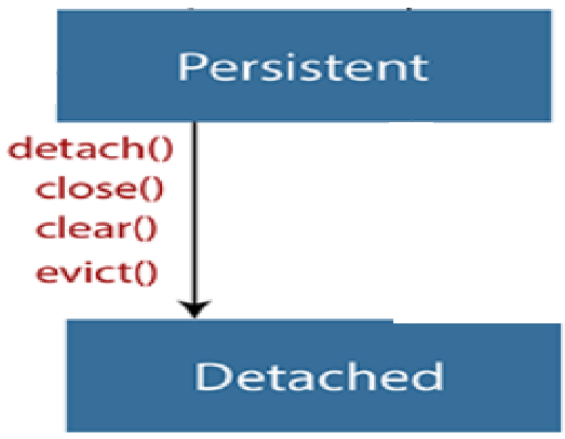
Un oggetto può passare dallo stato Persitent a Detached a fronte del metodo delle classe Session:

- detach()
- clear()
- evict()
- close()

Esempio:

```
session.detach(user);

oppure

session.clear();

oppure

session.close();

oppure

session.evict(user);
```

La figura seguente mostra in modo chiaro il passaggio di stato da Persitent a Detached a causa dell'invocazione del metodo delete.



Bibliografia

1 https://www.javatpoint.com/hibernate-lifecycle