

Modelli di Persistenza in J2EE

1° incontro Java User Group Genova

1 Marzo 2007

Giampiero Granatella

1 Marzo 2007

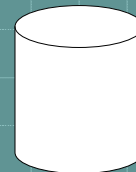
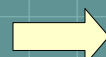
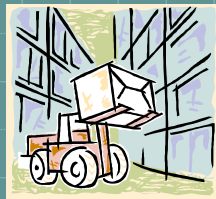
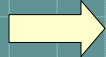
Persistenza e J2EE

1

Il problema della persistenza



Oggetti



DB relazionali

1 Marzo 2007

Persistenza e J2EE

2

L'evoluzione della persistenza in Java

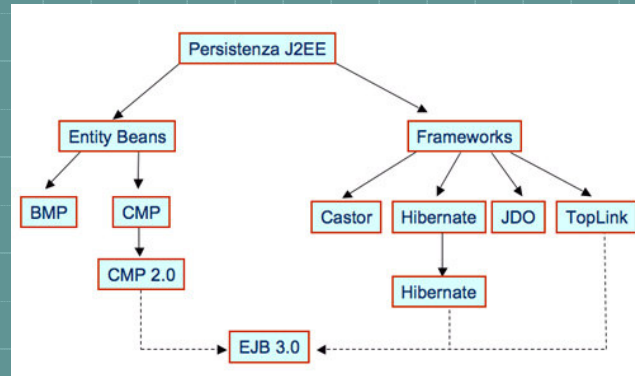


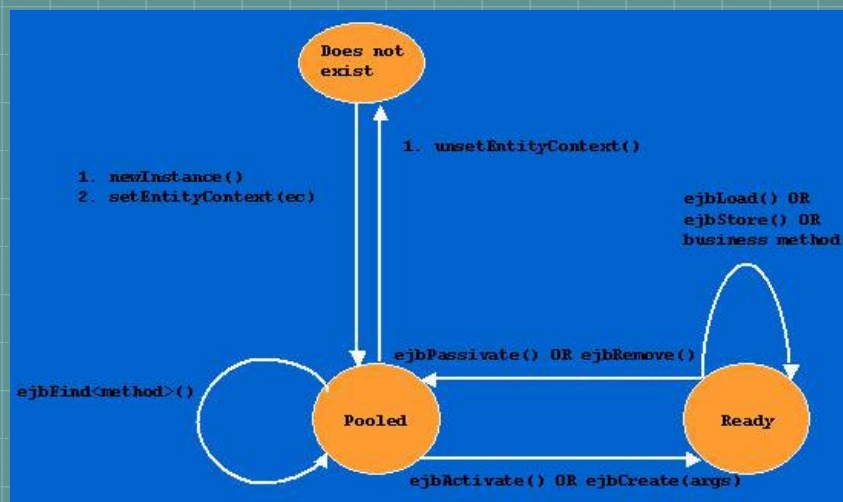
Immagine tratta Enterprise Java Beans 3 – Giovanni Puliti - Mokabyte

1 Marzo 2007

Persistenza e J2EE

3

Il precedente ciclo di vita degli EJB CMP (ver 2)



1 Marzo 2007

Persistenza e J2EE

4

Come realizzare la persistenza in J2EE ver 5

- ◆ La classe deve avere una Annotation con `javax.persistence.Entity`
- ◆ La classe deve fornire un costruttore public o protected con nessun argomento
- ◆ La classe, i suoi metodi e i suoi attributi non devono essere dichiarati final.
- ◆ Gli attributi dichiarati persistenti devono essere private o protected o devono avere la notazione `javax.persistence.Transient`

1 Marzo 2007

Persistenza e J2EE

5

Come realizzare la persistenza in J2EE ver 5 (2)

- ◆ Ogni entità deve avere una primary key che la identifichi. Gli attributi che sono primari key usano l'annotazione `javax.persistence.Id`. Se si hanno chiavi composte bisogna creare un'apposita classe.
- ◆ La molteplicità delle relazioni è gestita attraverso le notazioni `javax.persistence.OneToOne`, `javax.persistence.OneToMany`, ... sugli opportuni attributi dell'entità
- ◆ E' possibile fornire anche la direzionalità di una relazione

1 Marzo 2007

Persistenza e J2EE

6

Il ciclo di vita dei nuovi entity bean

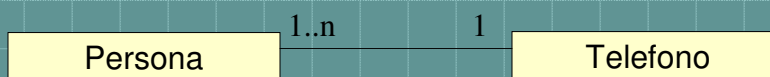
- ◆ Il ciclo di vita non è gestito dal container, ma esistono delle Annotation che lo rendono fruibile
 - ◆ @PrePersist: il metodo è invocato appena prima che il bean sia reso persistente nel DB;
 - ◆ @PostPersist: il metodo è invocato appena dopo che il bean sia reso persistente nel DB;
 - ◆ @PreRemove: il metodo è invocato appena prima che il bean sia rimosso dal DB;
 - ◆ @PostRemove: il metodo è invocato appena dopo che il bean sia rimosso dal DB;
 - ◆ @PreUpdate: il metodo è invocato appena prima che il bean sia aggiornato nel DB;
 - ◆ @PostUpdate: il metodo è invocato appena dopo che il bean sia aggiornato nel DB;
 - ◆ @PostLoad: il metodo è invocato appena dopo che i dati sono letti dal DB e immessi nel bean.

1 Marzo 2007

Persistenza e J2EE

7

Il modello d'esempio



1 Marzo 2007

Persistenza e J2EE

8

Netbeans e la persistenza J2EE

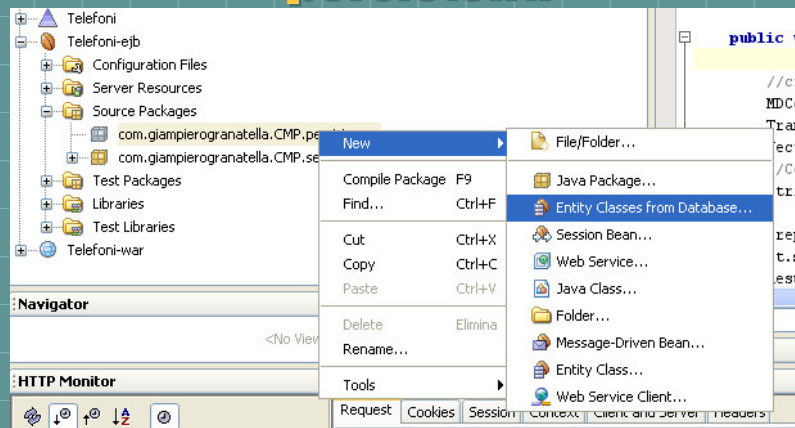
- ◆ Usiamo Netbeans per creare il nostro esempio completo per gestire la CMP (Container Based Persistence). La persistenza è infatti gestita dal un Container, nel nostro caso Glassfish (o Sun Java Server).
- ◆ Creiamo un nuovo progetto Enterprise->Enterprise Application che chiamo per semplicità Telefoni. A sua volta vengono creati altri due progetti Telefoni-ejb e Telefoni-war.
- ◆ Dal Tab 'Runtime' controlliamo di avere il Driver per postgres, e creiamo sotto 'Databases' -> 'New Connections' (col tasto destro). L'url della connessione è
jdbc:postgresql://127.0.0.1:5432/persistenza

1 Marzo 2007

Persistenza e J2EE

9

La creazione delle classi persistenti

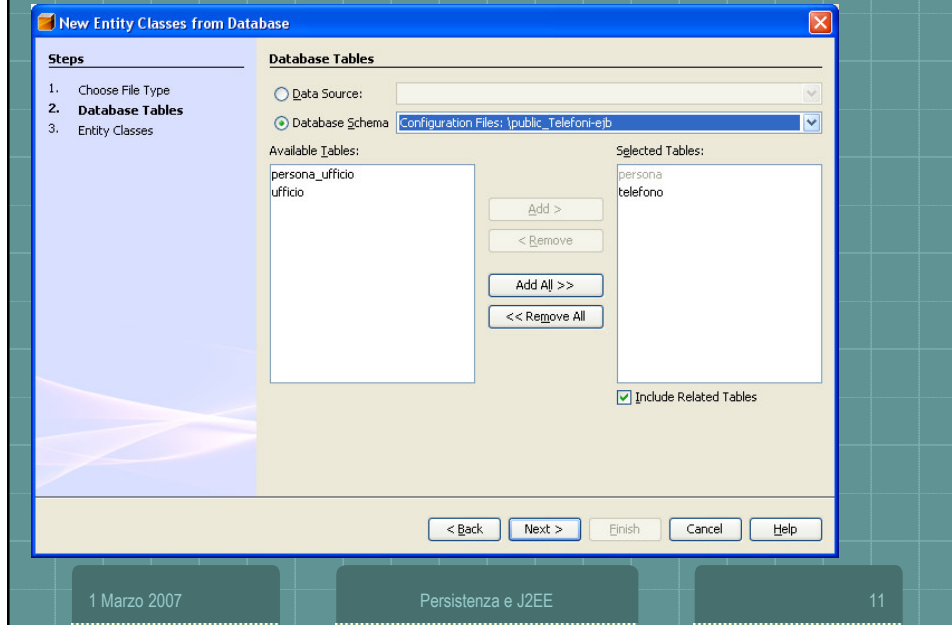


1 Marzo 2007

Persistenza e J2EE

10

La creazione delle classi persistenti (2)



La classe persona

```
@Entity
@Table(name = "persona")
@NamedQueries( {
    @NamedQuery(name = "Persona.findById", query = "SELECT p FROM
Persona p WHERE p.id = :id"),
    @NamedQuery(name = "Persona.findByCognome", query = "SELECT p
FROM Persona p WHERE p.cognome = :cognome"),
    @NamedQuery(name = "Persona.findByNome", query = "SELECT p
FROM Persona p WHERE p.nome = :nome")
})
public class Persona implements Serializable {
```

Gli attributi della classe persona

@Id

@Column(name = "id", nullable = false)

private Integer id;

@Column(name = "Cognome")

private String cognome;

@Column(name = "Nome")

private String nome;

@OneToMany(cascade = CascadeType.ALL, mappedBy = "personald")

private Collection<Telefono> telefonoCollection;

1 Marzo 2007

Persistenza e J2EE

13

I metodi della classe persona

- ◆ Un costruttore senza parametri
- ◆ Metodi get e set per gli attributi
- ◆ ...

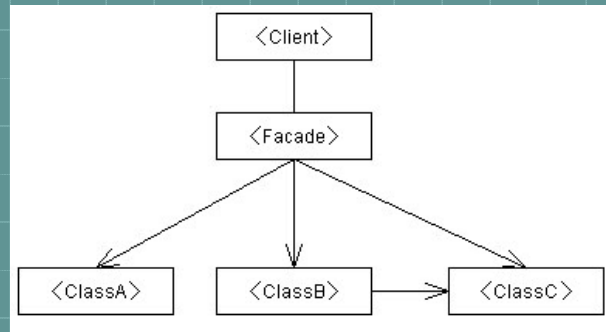
1 Marzo 2007

Persistenza e J2EE

14

Accedere alle classi persistenti

- ◆ Per accedere alle classi persistenti si utilizza un classe facade (in un ambiente distribuito sarà un session bean).



1 Marzo 2007

Persistenza e J2EE

15

Come accedere alle classi persistenti

```
@Stateless
public class Telefoni_sessionBean implements Telefoni_sessionRemote,
Telefoni_sessionLocal {
    @PersistenceContext
    private EntityManager em;
    public Telefoni_sessionBean() {
    }

    public Persona addPersonaTelefono(Persona pers, Telefono tel){

        Persona mergedpers = em.merge(pers);
        mergedpers.getTelefonoCollection().add(tel);
        return mergedpers;
    }
    ...
}
```

1 Marzo 2007

Persistenza e J2EE

16

Ricerca degli oggetti

```
public Persona searchForPersona(String id){
    Persona pers = (Persona)em.find(Persona.class, id);
    return pers;
}

public Telefono searchForTelefono(String id){
    Telefono tel = (Telefono)em.find(Telefono.class, id);
    return tel;
}

public List findPersonaByCognome(String cognome){
    List persone =
em.createNamedQuery("findPersonaByCognome").setParameter("cognome",
cognome).getResultList();
    return persone;
}
```

1 Marzo 2007

Persistenza e J2EE

17

Aggiungere un telefono ad una persona

```
public Persona addPersonaTelefono(Persona pers, Telefono tel){

    Persona mergedpers = em.merge(pers);
    mergedpers.getTelefonoCollection().add(tel);
    return mergedpers;

}
```

1 Marzo 2007

Persistenza e J2EE

18

Salvare/cancellare oggetti

//I seguenti due metodi sono usati per rimuovere un'entità o renderla persistente sul DB

```
@TransactionAttribute(TransactionAttributeType.REQUIRED)
public void remove(Object obj){
    Object mergedObj = em.merge(obj);
    em.remove(mergedObj);
}

public void persist(Object obj){
    em.persist(obj);
}
```

1 Marzo 2007

Persistenza e J2EE

19

Osservazioni finali

- ◆ L'uso delle Annotations rende molto semplice gestire classi persistenti.
- ◆ Il nuovo modello nasce solo dalla semplificazione? Credo di no. Con il nuovo modello le entità sono gestite in locale. In un ambiente distribuito i loro servizi sono esposti tramite un pattern Facade attraverso un session Bean. A questo punto gli oggetti persistenti non hanno più un'interfaccia remota, si riduce la loro capacità di essere distribuiti. Questo implica un predeterminato modello per le applicazioni distribuite. Questo modello entità (Entity Bean o classi persistenti) + facade (Session Bean) può essere usato con profitto anche per J2EE precedente alla versione 5.

1 Marzo 2007

Persistenza e J2EE

20

Osservazioni finali (2)

- ◆ Il nuovo modello di persistenza porta a pensare a componenti anche al di fuori di EJB e applicazioni distribuite. Il concetto di persistenza può essere utilizzato con facilità anche per semplici applicazioni web o stand-alone. Questo porta ad una maggiore pulizia dei progetti anche di piccole dimensioni.