

@Repository
@Transactional
@PersistenceContext

@Component

Notació genèrica per identificar components que haviem de ser detectats (i gestionats) per Spring



@Repository

Indica que el component particular s'encarregarà de realitzar operacions relacionades amb la persistència dels objectes (accés a bbdd)

Si esborrem @Transactional del nostre codi?

*javax.persistence.TransactionRequiredException:
No EntityManager with actual transaction available for current thread*

Els mètodes:

- persist
- merge
- remove

necessiten executar-se amb una transacció activa

@Transactional

```
@Transactional  
public void metode (String nom) {  
    // crea objecte persona  
    em.persist(persona);  
}
```

Crea una transacció al invocar-se al mètode i la finalitza al acabar el mateix mètode

Totes les operacions dins del mètode s'executen dins la mateixa transacció

disclaimer: algunes coses dites aquí no són 100% exactes. Més envant es precisaran


```
@Repository  
@Transactional  
public class Utilitat {  
    ...  
}
```

*Anotant tota la classe, s'aplica a tots el
mètodes*

@PersistenceContext

```
@Repository
@Transactional
public class Utilitat {

    @PersistenceContext
    private EntityManager em;

    ... reste mètodes
}
```

Persistence Context és un concepte molt important que veurem les següents setmanes.

Guarda similituds amb el concepte de transacció: com una "àrea de treball"

EntityManager hem vist que és l'objecte que ens ofereix els principals mètodes de persistència (p.e. find, persist, merge i remove)

@PersistenceContext fa que Spring injecti a la variable anotada un EntityManager associat al persistence context actiu