



@Transactional



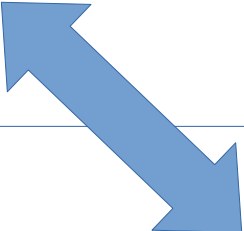
@Transactional

- Spring fournit une annotation @Transactional
- Automagiquement cela encapsule votre code hibernate dans une transaction.
- Inutile de démarrer et de fermer à chaque fois la transaction manuellement .
- Spring va gérer ce point pour vous. C'est un fonctionnement typique en arrière plan de spring, qu'un développeur doit maîtriser.

Exemple

```
public List<User> getUsers() {  
    // récupérer une session hibernate  
    Session session = sessionFactory.getCurrentSession();  
  
    // ouvrir transaction  
    session.beginTransaction();  
  
    // créer une requête  
    Query<User> query= session.createQuery("from T_user", User.class);  
  
    //executer la requête , récupérer son résultat  
    List<User> users = query.getResultList();  
  
    // commit – fermer transaction  
    session.getTransaction().commit();  
  
    return users;  
}
```

On se focalise sur
le code essentiel



```
@Transactional  
public List<User> getUsers() {  
    // récupérer une session hibernate  
    Session session = sessionFactory.getCurrentSession();  
  
    // créer une requête  
    Query<User> query= session.createQuery("from T_user", User.class);  
  
    //executer la requête , récupérer son résultat  
    List<User> users = query.getResultList();  
  
    return users;  
}
```