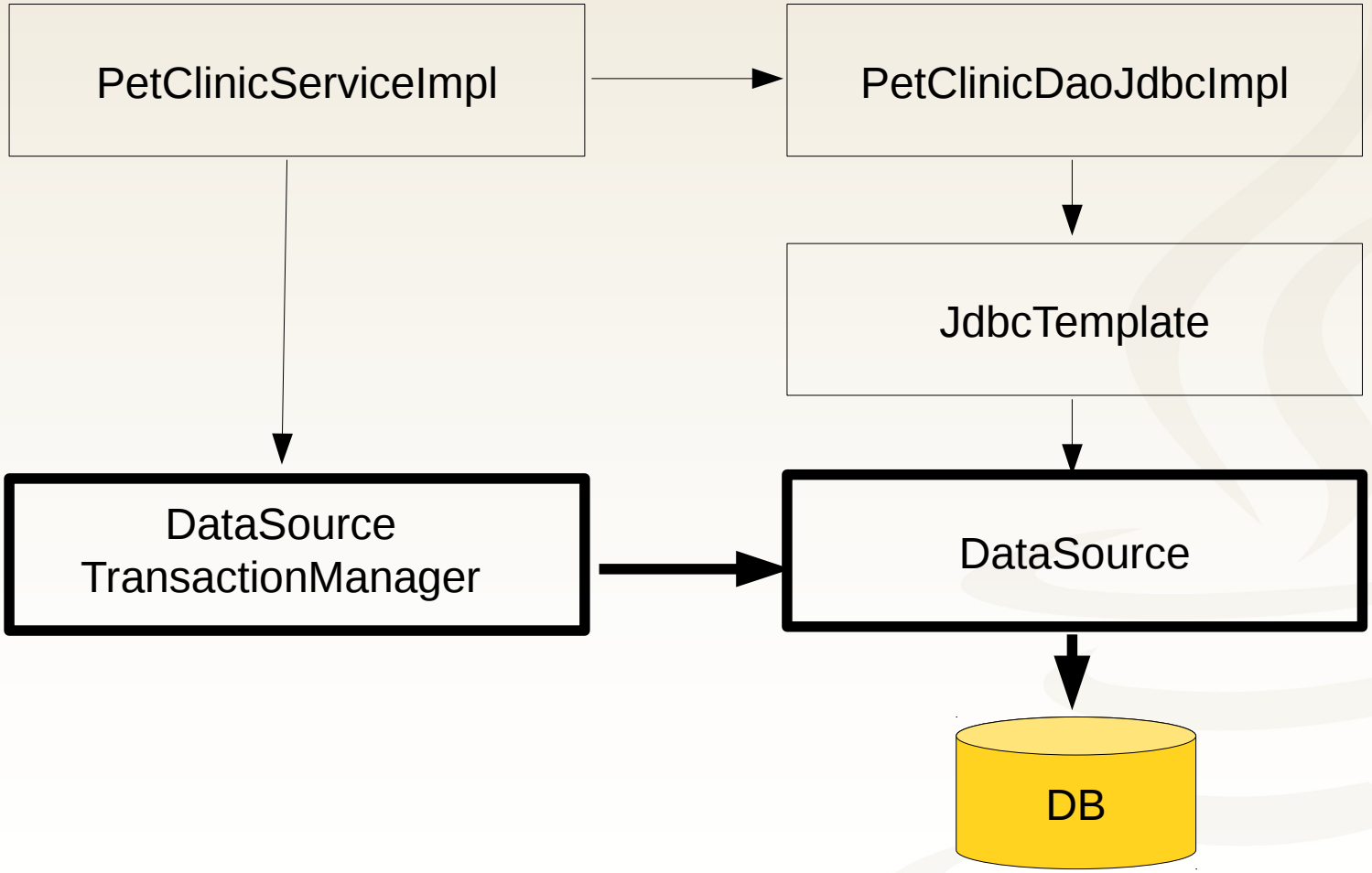
Programatik Transaction Yönetimi

Programatik Transaction Yönetimi

- Asenkron biçimde **arka planda** ve **uzun süre çalışacak** kod bloklarını transactional yapmak için kullanılabilir
- Ya da bir **metodun sadece belirli bölümleri**ni transactional yapmak için kullanılabilir
- Spring programatik TX için iki yol sunar
  - **TransactionTemplate** üzerinden
  - Doğrudan **PlatformTransactionManager** kullanarak

Programatik Transaction Yönetimi



TransactionTemplate Kullanımı

- Servis metotlarındaki transactional işlemler **TransactionTemplate isimli bir nesne üzerinden** yürütülür
- Spring'in genel **template mantığı**na sahiptir
- Transactional kod bloğunu TransactionTemplate'a iletmek için **Callback** yöntemi kullanılır

TransactionTemplate Kullanımı

TransactionTemplate'ı oluşturmak için **PlatformTransactionManager**'a ihtiyaç duyulur

```
TransactionTemplate transactionTemplate = new
TransactionTemplate(transactionManager);

transactionTemplate.execute(new TransactionCallbackWithoutResult() {

    protected void doInTransactionWithoutResult(TransactionStatus status){

    //transactional is mantigi...

    }
});
```

Metot başlangıcında Spring bir TX başlatır. Metot başarılı sonlandığında da TX'i commit eder. İş bloğu içerisinde Meydana gelecek herhangi bir **exception durumunda ise TX rollback** edilir

Eğer transactional iş bloğu sonucunda bir değer dönülmesi gerekiyorsa Bu durumda **TransactionCallback** arayüzü kullanılabilir

TransactionTemplate Kullanımı

- TransactionTemplate'in **default ayarları @Transactional ile aynı**dır
- Ancak **herhangi bir exception'da rollback** gerçekleşir
- Rollback kuralını değiştirmek mümkün değildir
- Diğer ayarlar değiştirilebilir

PlatformTransactionManager Kullanımı

- **PlatformTransactionManager** bean'ı servis bean'ine enjekte edilir
- Servis metodu içerisinde **TransactionDefinition** ile bir TX tanımı oluşturulur
- Bu tanım ile TX manager üzerinden bir **TransactionStatus** elde edilir
- İş mantığı yürütülür
- İşlem sonucunda TransactionStatus ile **commit** veya **rollback** gerçekleştirilir

PlatformTransactionManager Kullanımı

TransactionDefinition TX tanımını ifade eder
TX'in Isolation, Propagation, Timeout, read-only özellikleri tanımlanabilir

```
DefaultTransactionDefinition txDef = new
DefaultTransactionDefinition();
txDef.setPropagationBehavior(TransactionDefinition.PROPAGATION_REQUIRED);

TransactionStatus status =
transactionManager.getTransaction(txDef);
```

TransactionStatus transaction execution'ı kontrol eder

```
try {
  // is mantigi calistirilir...
  transactionManager.commit(status);
} catch (MyException ex) {
  transactionManager.rollback(status);
  throw ex;
}
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com