Aynı Bean İçerisinde Başka Bir Transactional Metot Çağırmak

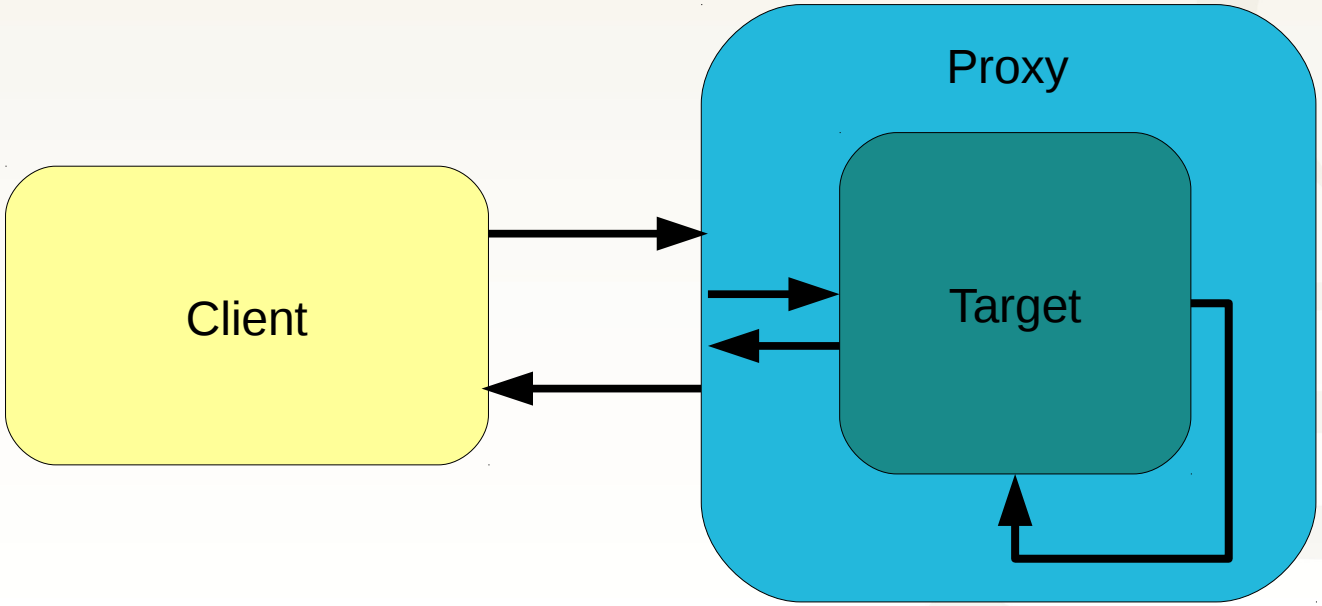
Aynı Bean İçerisinde Başka Bir Transactional Metot Çağırmak

JAVA Eğitimleri

- Spring transaction yönetimi **proxy örüntüsü** üzerine kuruludur
- Transaction davranışı **sadece public metot invokasyonları**nda devreye girebilir
- Bu metot invokasyonları ayrıca **proxy dışından gerçekleşmeli**dir
- Bir metot içerisinde **aynı bean'in başka bir metodu**na erişildiğinde bu proxy üzerinden gerçekleşmez

Aynı Bean İçerisinde Başka Bir Transactional Metot Çağırmak

- Dolayısı ile bu çağrı proxy üzerinden gerçekleşmediği için **transaction davranışı da devreye girmeyecek**tir



Aynı Bean İçerisinde Başka Bir Transactional Metot Çağırmak

```
@Service
public class FooService {

   @Transactional(propagation=Propagation.REQUIRED)
   public void foo() {
      this.bar();
   }

   @Transactional(propagation=Propagation.REQUIRES_NEW)
   public void bar() {

   }
}
```

Aynı Bean İçerisinde Başka Bir Transactional Metot Çağırmak

- Bu probleme **üç farklı çözüm yolu** vardır
  - İkinci transactional metodu **ayrı bir bean**'e taşımak
  - Metot içerisinde kendine **ApplicationContext üzerinden lookup** yapmak
  - Metot içerisinde o anki **proxy nesneye erişim** sağlamak

İkinci Metodu Ayrı Bir Bean'e Taşımak

```
@Service
public class FooService {

    @Autowired
    private BarService barService;

    @Transactional(propagation = Propagation.REQUIRED)
    public void foo() {
        barService.bar();
    }
}


@Service
public class BarService {
    @Transactional(propagation = Propagation.REQUIRES_NEW)
    public void bar() {

    }
}
```

Metot İçerisinde Kendine Lookup Yapmak

JAVA Eğitimleri

```
@Service
public class FooService
                implements ApplicationContextAware, BeanNameAware {

    private ApplicationContext applicationContext;
    private String beanName;

    @Override
    public void setApplicationContext(
                ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    @Override
    public void setBeanName(String beanName) {
        this.beanName = beanName;
    }

...
}
```

Metot İçerisinde Kendine Lookup Yapmak

JAVA Eğitimleri

```
@Service
public class FooService
                implements ApplicationContextAware, BeanNameAware {

...

    @Transactional(propagation = Propagation.REQUIRED)
    public void foo() {
        FooService fooService = applicationContext
                        .getBean(beanName,FooService.class);
        fooService.bar();
    }

    @Transactional(propagation = Propagation.REQUIRES_NEW)
    public void bar() {
    }
}
```

Metot İçerisinde Proxy Nesneye Erişim Sağlamak

```
@Service
public class FooService {

    @Transactional(propagation=Propagation.REQUIRED)
    public void foo() {
        FooService proxy =
            (FooService)AopContext.currentProxy();
        proxy.bar();
    }

    @Transactional(propagation=Propagation.REQUIRES_NEW)
    public void bar() {

    }
}
```

Metot İçerisinde Proxy Nesneye Erişim Sağlamak

- AopContext'in o anki proxy nesneyi dönebilmesi için AOP altyapısında **expose proxy özelliğ**inin aktive edilmiş olması gerekir

```
<beans...>
    <aop:config expose-proxy="true">
    </aop:config>

    <tx:annotation-driven/>
</beans>
```

İletişim

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