Spring-tx 事物源码解析

示例代码

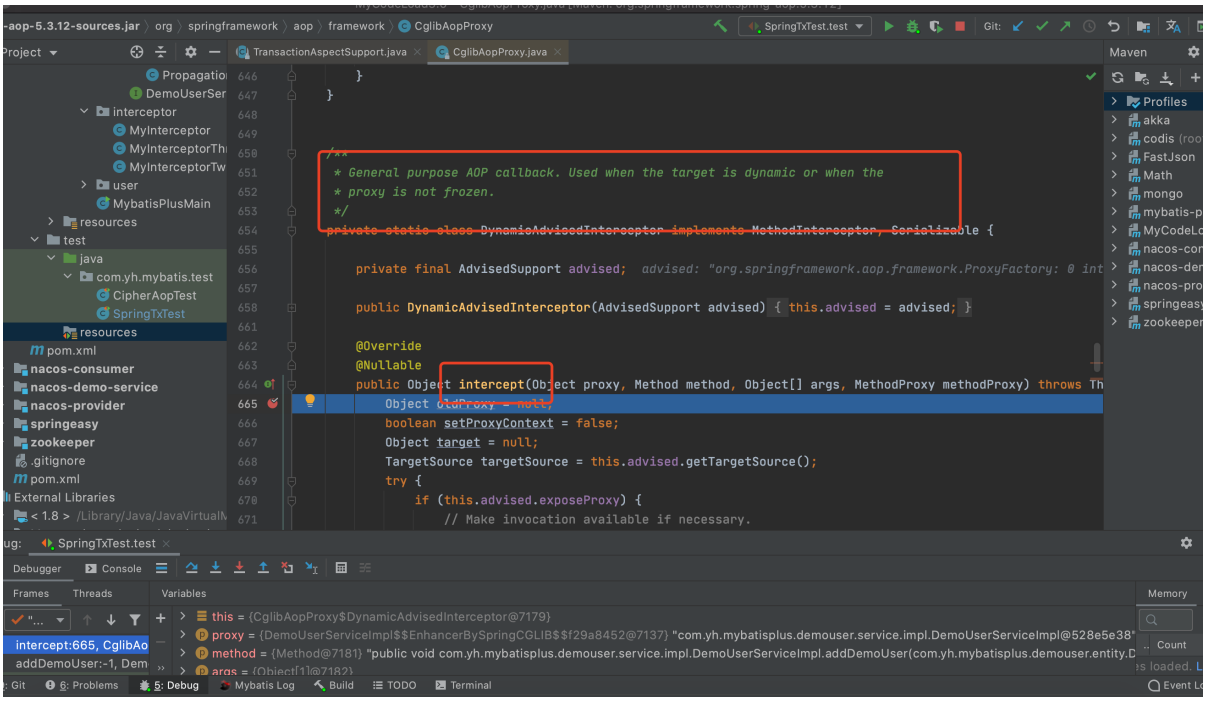
```
@Transactional
@Service
public class DemoUserServiceImpl implements DemoUserService {

    @Autowired
    private DemoUserMapper demoUserMapper;

    @Override
    public void addDemoUser(DemoUser demoUser) {
            demoUserMapper.insert(demoUser);
            int i = 1 / 0;
    }
}
```

```
@Test
public void test() {
demoUserService.addDemoUser(DemoUser.builder()
.name("事物测试11111").age(100)
.build());
}
```

Spring-tx 是如何 创建事物 提交事物 事物回滚?

```
 1 CglibAopProxy AOP Intercept()
```

```
/**
 * General purpose AOP callback. Used when the target is dynamic or when the
 * proxy is not frozen.
 */
```



1.1. 跟踪方法

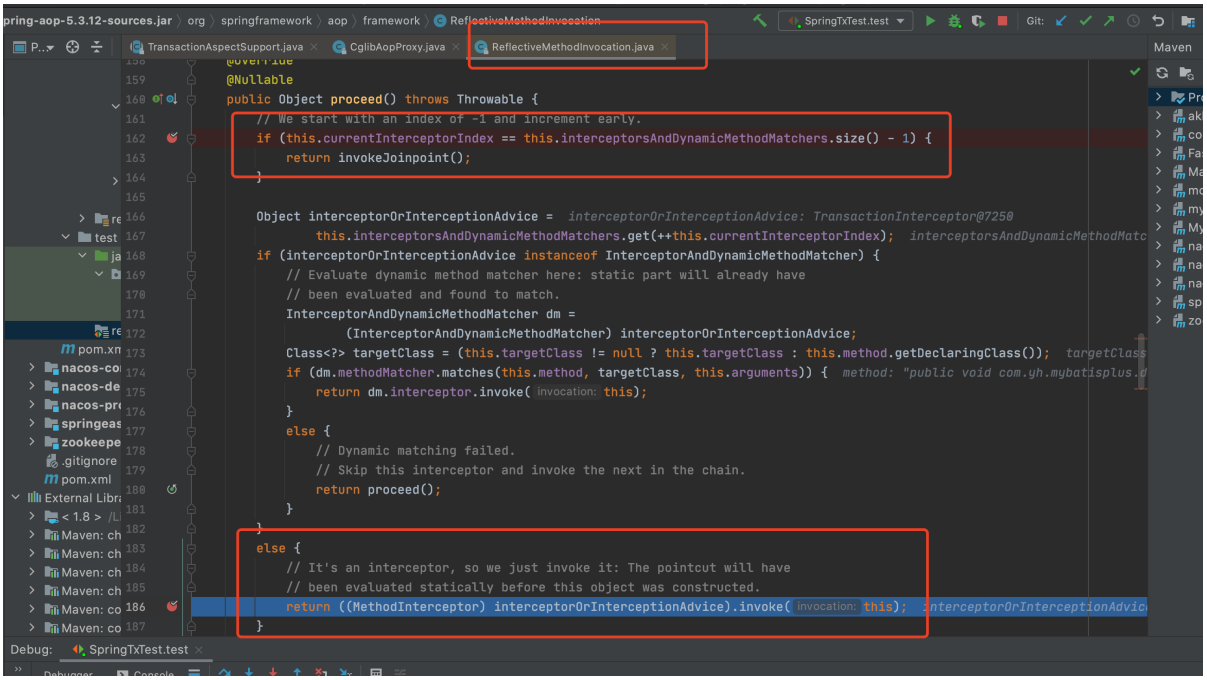
```
// We need to create a method invocation...
retVal = new CglibMethodInvocation(proxy, target, method, args, targetClass, chain, methodProxy).proceed();
```

2

ReflectiveMethodInvocation

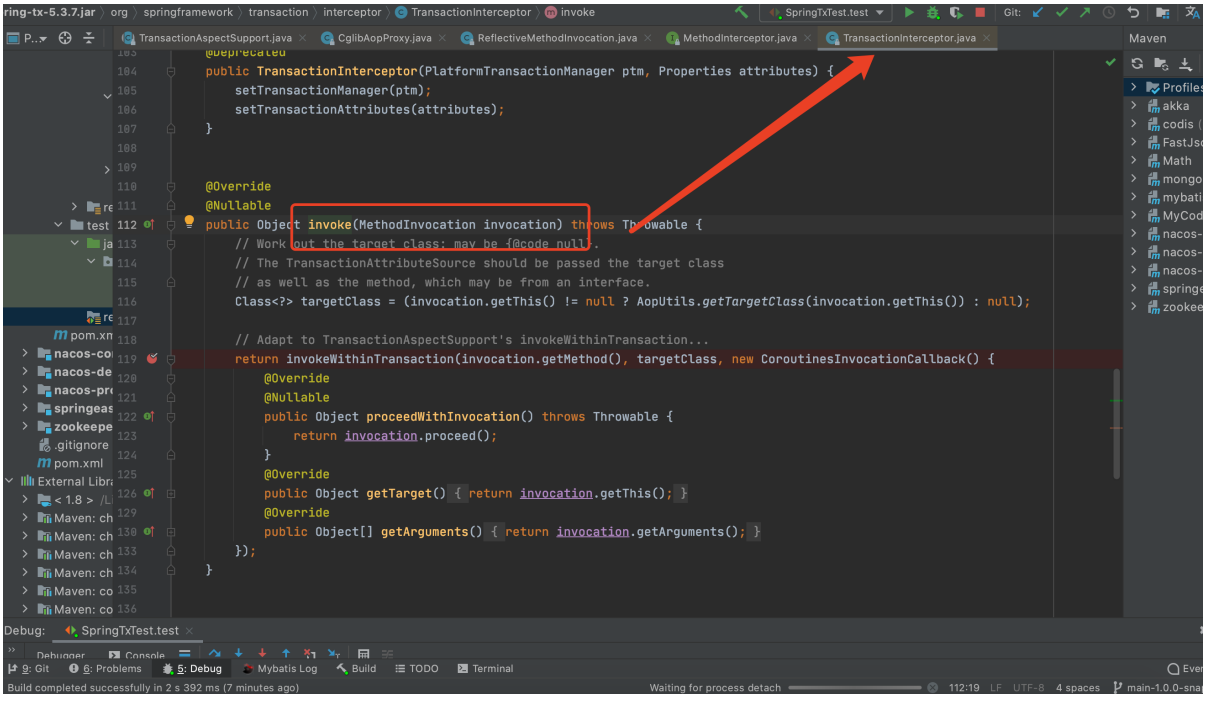
```
// We start with an index of -1 and increment early.
if (this.currentInterceptorIndex == this.interceptorsAndDynamicMethodMatchers.size() - 1) {
   return invokeJoinpoint();
}
```

1 2 3 /



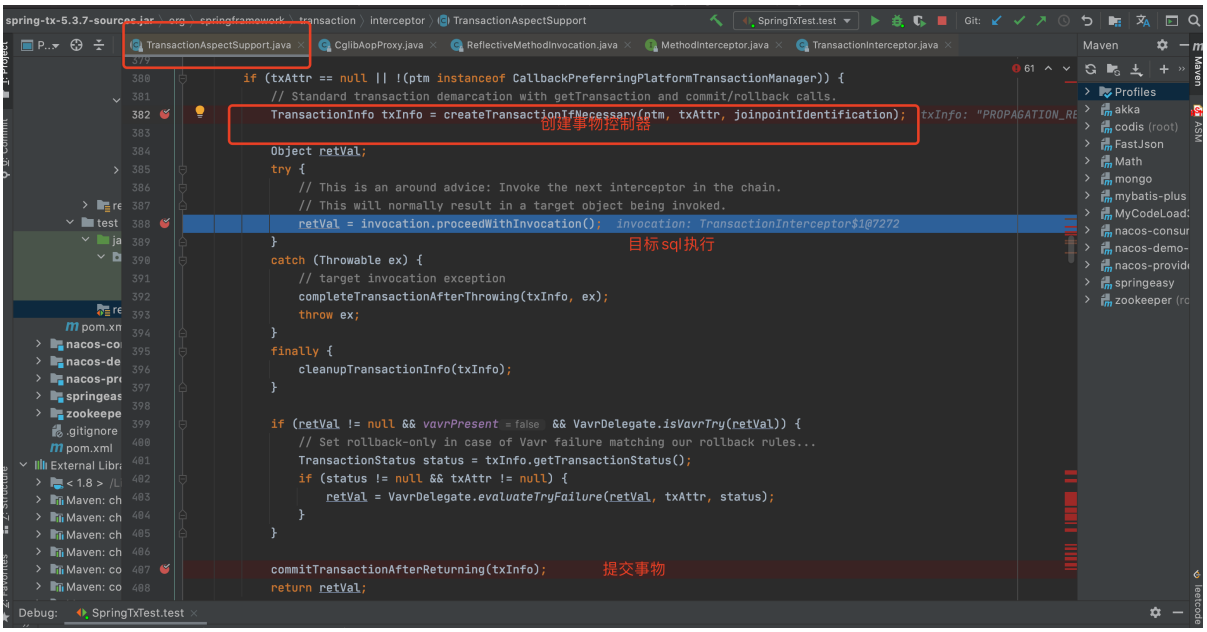
2.1

TransactionInterceptor

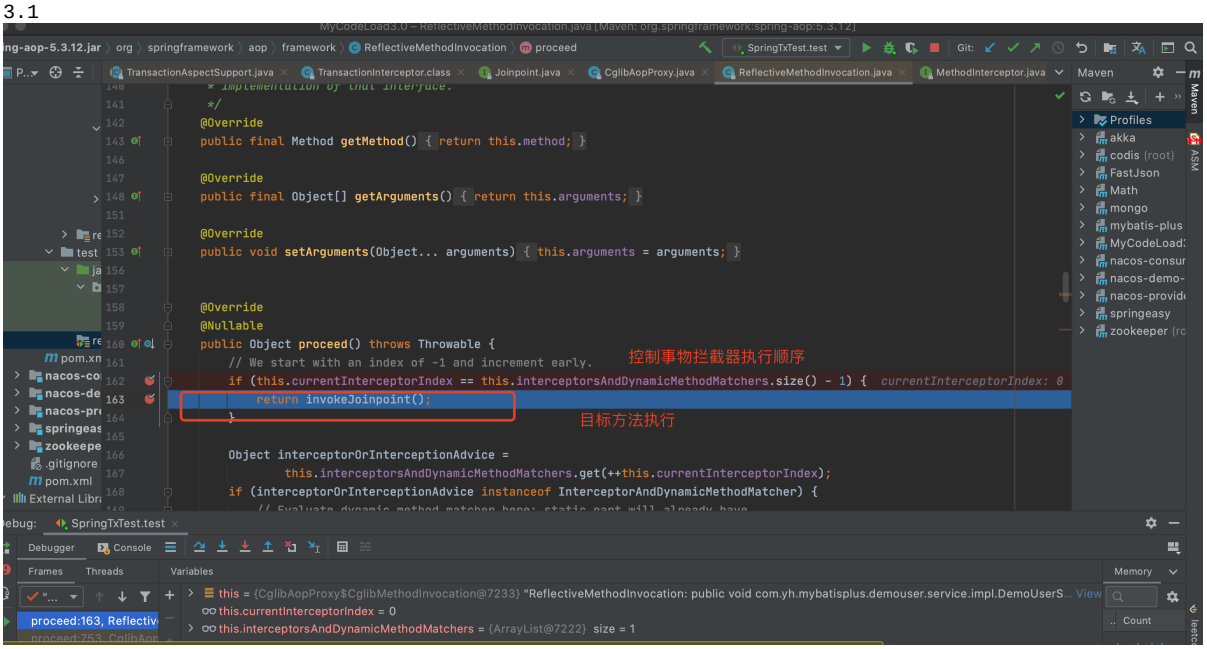


3

TransactionAspectSupport.invokeWithinTransaction()



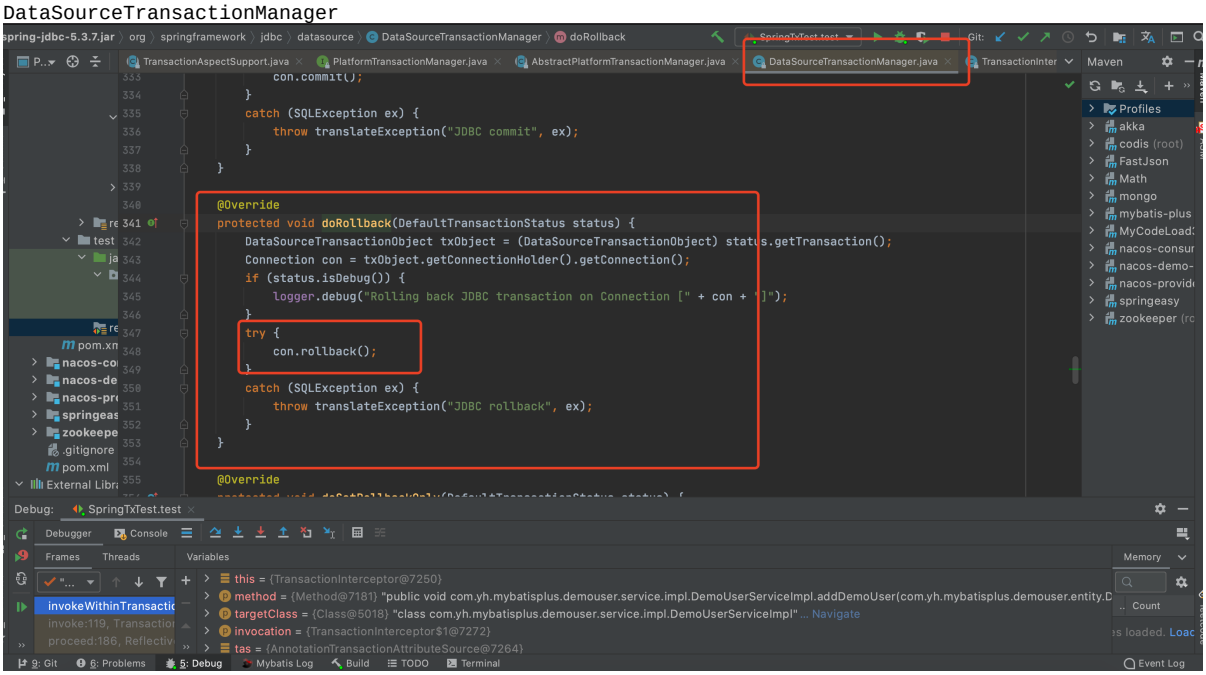
3.1



3.2

```
completeTransactionAfterThrowing(txInfo, ex);
```

DataSourceTransactionManager



4 end

GitHub