CGLIB proxies should still consider @Transactional annotations on interface methods [SPR-14322] #1

> https://github.com/spring-projects/spring-framework/issues/18894

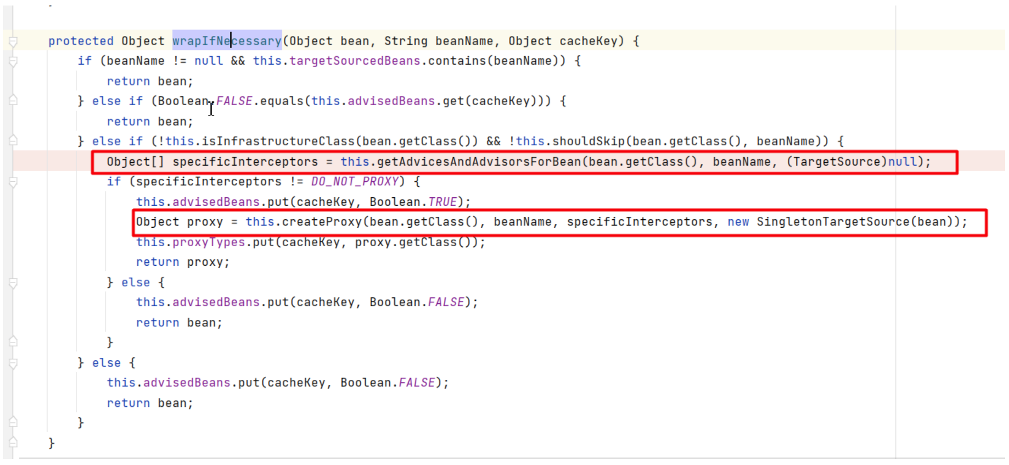
If an application components implements an interface whose methods carry annotations that are triggering interceptors (e.g. for transactions), enabling target class proxying will result in the interceptors for those annotations not being triggered anymore. Here's a sample:
如　个　件实　了　　个　口，　个　口　　　上使　了可以出发Interceptor　　　，　如事务
如　　了proxyTargetClass为true将会导　　　　发interceptor

```java
interface SomeComponent {

 @Transactional
 void init();
}

@Component
class SomeComponentImpl implements SomeComponent {

  @Override
  public void init() {
  if (!TransactionSynchronizationManager.isActualTransactionActive()) {
  throw new IllegalStateException("Expected transaction to be active!");
  }
  }
 }

 @Component
 class Invoker {

  public Invoker(List<SomeComponent> components) {
  components.forEach(SomeComponent::init);
  }
 }

 启动类上使用@EnableTransactionManagement(proxyTargetClass = false)
 来决定是用JDK动态代理还是cglib代理

```

If the above is bootstrapped with standard `@EnableTransactionManagement` the instances handed to the constructor of `Invoker` are JDK proxies and the lookup of the advice chain results in the interceptor for transactions being returned and thus activated. If `proxyTargetClass` is set to `true`, the instances received by the constructor are CGLib proxies and the lookup of the advice chain results in an empty one and thus no transaction is created in the first place.

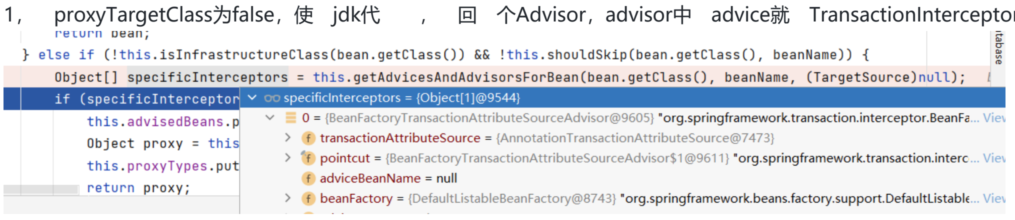
@Transactional定义在　口上　　候，如　　们使　　　　口代　，　么　　　回　advice中会包含TransactionInterceptor　但　如　　使　CGLib代　则　回了　个　　advice　　个　如何　　？

　们　　代　　入口　org.springframework.aop.framework.autoproxy.AbstractAutoProxyCreator#wrapIfNecessary
在　个wrapIfNecessary中
　先　　　getAdvicesAndAdvisorsForBean　　　取interceptors，　后如　　取到　interceptor不　　　　，则就会createProxy



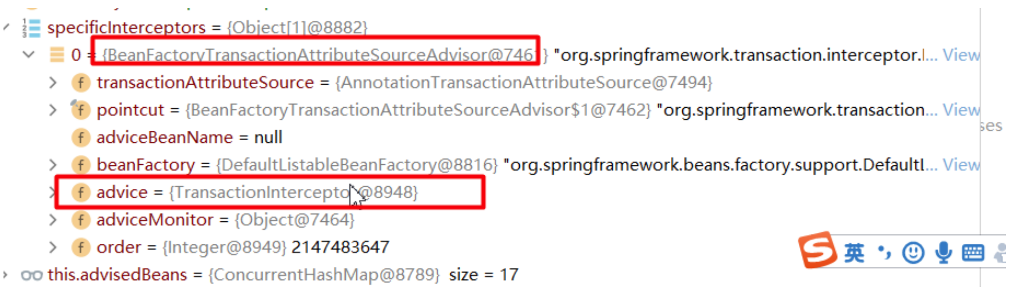
```
protected Object wrapIfNecessary(Object bean, String beanName, Object cacheKey) {
    if (beanName != null && this.targetSourcedBeans.contains(beanName)) {
        return bean;
    } else if (Boolean.FALSE.equals(this.advisedBeans.get(cacheKey))) {
        return bean;
    } else if (!this.isInfrastructureClass(bean.getClass()) && !this.shouldSkip(bean.getClass(), beanName)) {
        Object[] specificInterceptors = this.getAdvicesAndAdvisorsForBean(bean.getClass(), beanName, (TargetSource)null);
        if (specificInterceptors != DO_NOT_PROXY) {
            this.advisedBeans.put(cacheKey, Boolean.TRUE);
            Object proxy = this.createProxy(bean.getClass(), beanName, specificInterceptors, new SingletonTargetSource(bean));
            this.proxyTypes.put(cacheKey, proxy.getClass());
            return proxy;
        } else {
            this.advisedBeans.put(cacheKey, Boolean.FALSE);
            return bean;
        }
    } else {
        this.advisedBeans.put(cacheKey, Boolean.FALSE);
        return bean;
    }
}
```

1， proxyTargetClass为false，使 jdk代 ， 回 个Advisor，advisor中 advice就 TransactionInterceptor



```
    return bean;
} else if (!this.isInfrastructureClass(bean.getClass()) && !this.shouldSkip(bean.getClass(), beanName)) {
    Object[] specificInterceptors = this.getAdvicesAndAdvisorsForBean(bean.getClass(), beanName, (TargetSource)null);
    if (specificInterceptor
        this.advisedBeans.p
        Object proxy = this
        this.proxyTypes.put
        return proxy;
```

```
specificInterceptors = {Object[1]@9544}
  0 = {BeanFactoryTransactionAttributeSourceAdvisor@9605} "org.springframework.transaction.interceptor.BeanFa... View
    transactionAttributeSource = {AnnotationTransactionAttributeSource@7473}
    pointcut = {BeanFactoryTransactionAttributeSourceAdvisor$1@9611} "org.springframework.transaction.interc... View
    adviceBeanName = null
    beanFactory = {DefaultListableBeanFactory@8743} "org.springframework.beans.factory.support.DefaultListable... View
```

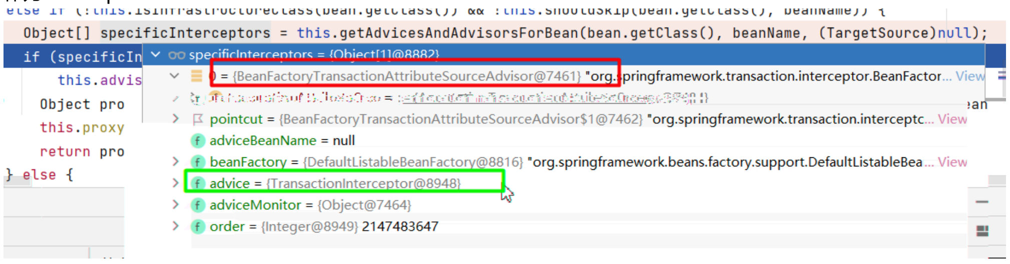


```
specificInterceptors = {Object[1]@8882}
  0 = {BeanFactoryTransactionAttributeSourceAdvisor@7461} "org.springframework.transaction.interceptor.I... View
    transactionAttributeSource = {AnnotationTransactionAttributeSource@7494}
    pointcut = {BeanFactoryTransactionAttributeSourceAdvisor$1@7462} "org.springframework.transaction... View
    adviceBeanName = null
    beanFactory = {DefaultListableBeanFactory@8816} "org.springframework.beans.factory.support.DefaultI... View
    advice = {TransactionInterceptor@8948}
    adviceMonitor = {Object@7464}
    order = {Integer@8949} 2147483647
this.advisedBeans = {ConcurrentHashMap@8789} size = 17
```

个BeanFactoryTransactionAttributeSourceAdvisor对 在ProxyTransactionManagementConfiguration 个 中 入

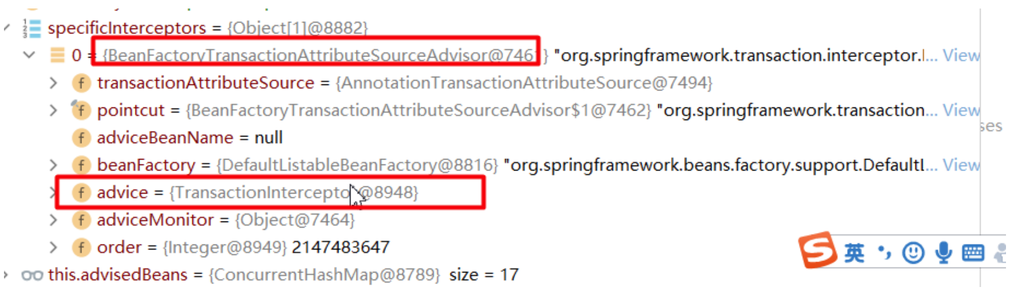
```
@Bean(name = TransactionManagementConfigUtils.TRANSACTION_ADVISOR_BEAN_NAME)
@Role(BeanDefinition.ROLE_INFRASTRUCTURE)
public BeanFactoryTransactionAttributeSourceAdvisor transactionAdvisor() {
  BeanFactoryTransactionAttributeSourceAdvisor advisor = new BeanFactoryTransactionAttributeSourceAdvisor();
  advisor.setTransactionAttributeSource(transactionAttributeSource());
  advisor.setAdvice(transactionInterceptor());
  advisor.setOrder(this.enableTx.<Integer>getNumber("order"));
  return advisor;
}
```

2， proxyTargetClass为true 候，上 getAdvicesAndAdvisorsForBean 也会 回 个BeanFactoryTransactionAttributeSourceAdvisor对 作为interceptor



```
} else if (!this.isInfrastructureClass(bean.getClass()) && !this.shouldSkip(bean.getClass(), beanName)) {
    Object[] specificInterceptors = this.getAdvicesAndAdvisorsForBean(bean.getClass(), beanName, (TargetSource)null);
    if (specificIn
        this.advis
      Object pro
      this.proxy
      return pro
} else {
```

```
specificInterceptors = {Object[1]@8882}
  0 = {BeanFactoryTransactionAttributeSourceAdvisor@7461} "org.springframework.transaction.interceptor.BeanFactor... View
    pointcut = {BeanFactoryTransactionAttributeSourceAdvisor$1@7462} "org.springframework.transaction.interceptc... View
    adviceBeanName = null
    beanFactory = {DefaultListableBeanFactory@8816} "org.springframework.beans.factory.support.DefaultListableBea... View
    advice = {TransactionInterceptor@8948}
    adviceMonitor = {Object@7464}
    order = {Integer@8949} 2147483647
```

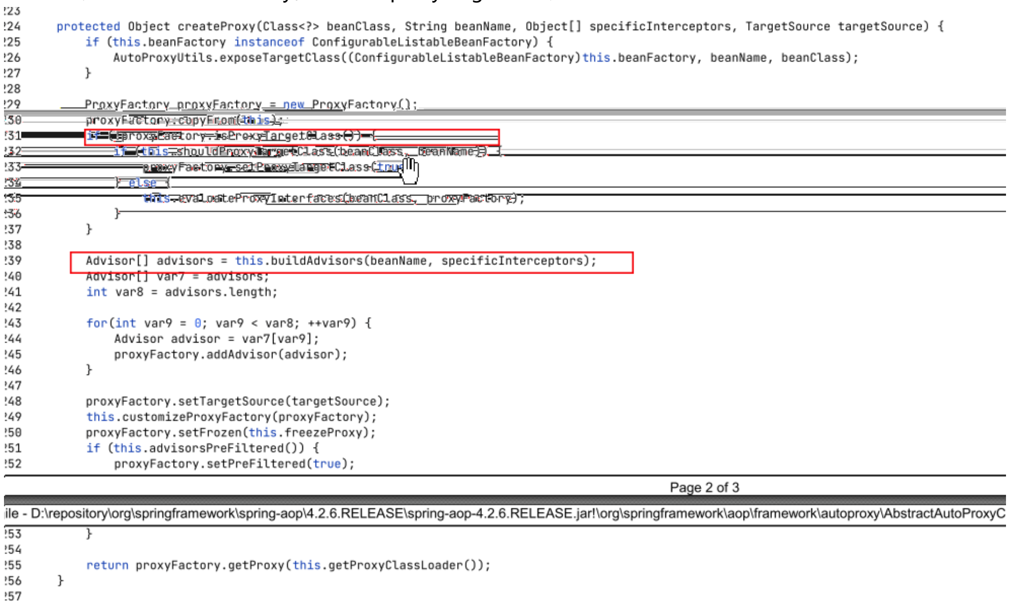


```
specificInterceptors = {Object[1]@8882}
  0 = {BeanFactoryTransactionAttributeSourceAdvisor@746} "org.springframework.transaction.interceptor.l... View
    transactionAttributeSource = {AnnotationTransactionAttributeSource@7494}
    pointcut = {BeanFactoryTransactionAttributeSourceAdvisor$1@7462} "org.springframework.transaction... View
    adviceBeanName = null
    beanFactory = {DefaultListableBeanFactory@8816} "org.springframework.beans.factory.support.Defaultl... View
    advice = {TransactionInterceptor@8948}
    adviceMonitor = {Object@7464}
    order = {Integer@8949} 2147483647
this.advisedBeans = {ConcurrentHashMap@8789}  size = 17
```

也就　　不　　proxyTargetClass为true　　false，　们　可以为SomeComponentImpl对　　到 specificInterceptors

8

后　们分　　二个 createProxy，主　　　proxyTargetClass为true　　况



```java
    protected Object createProxy(Class<?> beanClass, String beanName, Object[] specificInterceptors, TargetSource targetSource) {
        if (this.beanFactory instanceof ConfigurableListableBeanFactory) {
            AutoProxyUtils.exposeTargetClass((ConfigurableListableBeanFactory)this.beanFactory, beanName, beanClass);
        }

        ProxyFactory proxyFactory = new ProxyFactory();
        proxyFactory.copyFrom(this);
        if (!proxyFactory.isProxyTargetClass()) {
            if (this.shouldProxyTargetClass(beanClass, beanName)) {
                proxyFactory.setProxyTargetClass(true);
            } else {
                this.evaluateProxyInterfaces(beanClass, proxyFactory);
            }
        }

        Advisor[] advisors = this.buildAdvisors(beanName, specificInterceptors);
        Advisor[] var7 = advisors;
        int var8 = advisors.length;

        for(int var9 = 0; var9 < var8; ++var9) {
            Advisor advisor = var7[var9];
            proxyFactory.addAdvisor(advisor);
        }

        proxyFactory.setTargetSource(targetSource);
        this.customizeProxyFactory(proxyFactory);
        proxyFactory.setFrozen(this.freezeProxy);
        if (this.advisorsPreFiltered()) {
            proxyFactory.setPreFiltered(true);
```

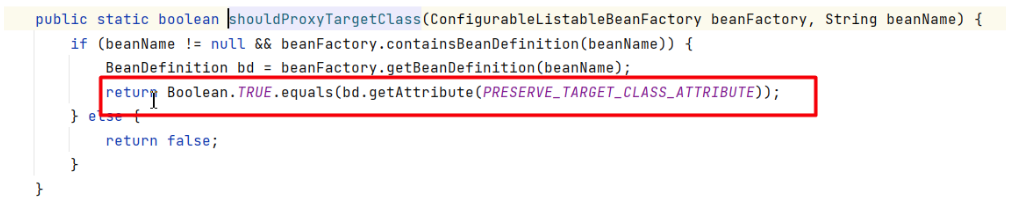
Page 2 of 3

ile - D:\repository\org\springframework\spring-aop\4.2.6.RELEASE\spring-aop-4.2.6.RELEASE.jar!\org\springframework\aop\framework\autoproxy\AbstractAutoProxyC

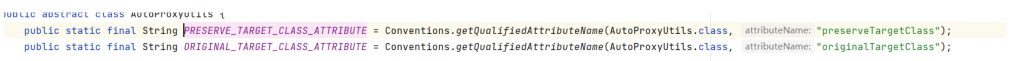
```java
        }

        return proxyFactory.getProxy(this.getProxyClassLoader());
    }
```

]

在createProxy　　中，　先　们会　　proxyFactory中　proxyTargetClass属　，同　会判　BeanDefinition中　preserveTargetClass属　，
　二个就　buildAdvisor，buildAdvisor　参　　前　getAdvicesAndAdvisorsForBean　　回值，
　　buildAdvisor　　回值　会　　作为Advisor 创　Proxy
　后在后　　proxyFactory.getProxy中会　　proxyTargetClass　判　　否使　cglib代



```java
public static boolean shouldProxyTargetClass(ConfigurableListableBeanFactory beanFactory, String beanName) {
    if (beanName != null && beanFactory.containsBeanDefinition(beanName)) {
        BeanDefinition bd = beanFactory.getBeanDefinition(beanName);
        return Boolean.TRUE.equals(bd.getAttribute(PRESERVE_TARGET_CLASS_ATTRIBUTE));
    } else {
        return false;
    }
}
```

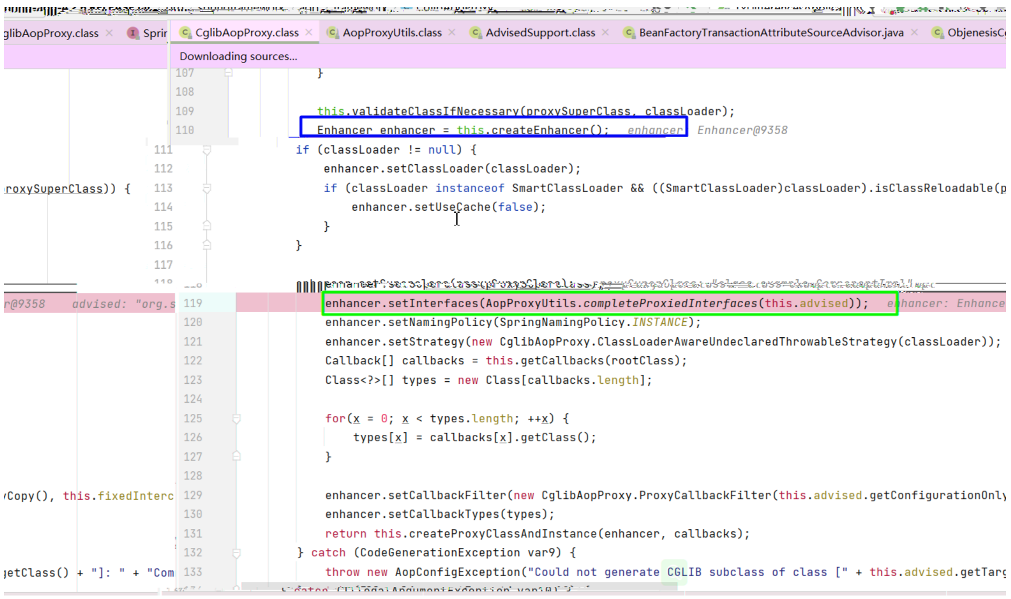


```java
public abstract class AutoProxyUtils {
    public static final String PRESERVE_TARGET_CLASS_ATTRIBUTE = Conventions.getQualifiedAttributeName(AutoProxyUtils.class,  attributeName: "preserveTargetClass");
    public static final String ORIGINAL_TARGET_CLASS_ATTRIBUTE = Conventions.getQualifiedAttributeName(AutoProxyUtils.class,  attributeName: "originalTargetClass");
```

　们　在　　　：不　　proxyTargetClass为true　　false，在getAdvicesAndAdvisorsForBean　和createProxy　　内　buildAdvisor 中
回了BeanFactoryTransactionAttributeSourceAdvisor作为Advisor，　么为什么proxyTargetClass为true　　候　　实　事务代　，　仅在
proxyTargetClass为false　　候　　SomeComponentImpl　　init　　　事务?

　：　proxyTargetClass 为false　　候使　jdk动　代

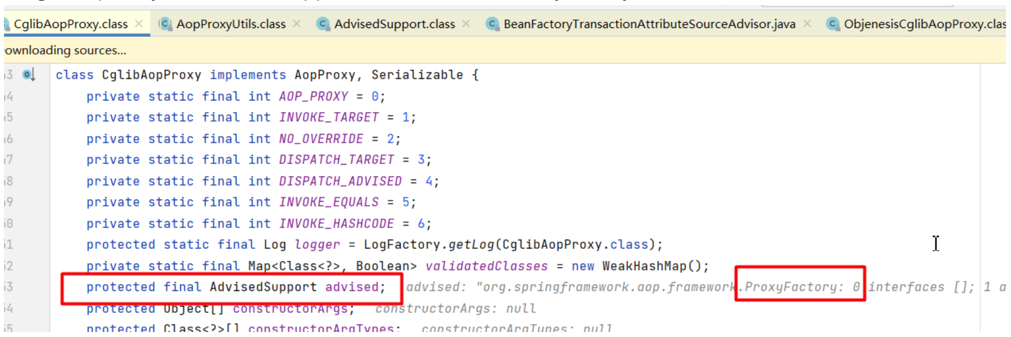
　二　proxyTargetClass为true　　偶 使　cglib代　，proxyFactory　getProxy　　会　发下　　getProxy
在org.springframework.aop.framework.CglibAopProxy#getProxy(java.lang.ClassLoader)



```
glibAopProxy.class × | Sprir | CglibAopProxy.class × | AopProxyUtils.class × | AdvisedSupport.class × | BeanFactoryTransactionAttributeSourceAdvisor.java × | ObjenesisC
Downloading sources...
            }

            this.validateClassIfNecessary(proxySuperClass, classLoader);
            Enhancer enhancer = this.createEnhancer();   enhancer  Enhancer@9358
        if (classLoader != null) {
            enhancer.setClassLoader(classLoader);
            if (classLoader instanceof SmartClassLoader && ((SmartClassLoader)classLoader).isClassReloadable(p
                enhancer.setUseCache(false);
            }
        }

            enhancer.setInterfaces(AopProxyUtils.completeProxiedInterfaces(this.advised));   enhancer: Enhance
            enhancer.setNamingPolicy(SpringNamingPolicy.INSTANCE);
            enhancer.setStrategy(new CglibAopProxy.ClassLoaderAwareUndeclaredThrowableStrategy(classLoader));
            Callback[] callbacks = this.getCallbacks(rootClass);
            Class<?>[] types = new Class[callbacks.length];

            for(x = 0; x < types.length; ++x) {
                types[x] = callbacks[x].getClass();
            }

            enhancer.setCallbackFilter(new CglibAopProxy.ProxyCallbackFilter(this.advised.getConfigurationOnly
            enhancer.setCallbackTypes(types);
            return this.createProxyClassAndInstance(enhancer, callbacks);
        } catch (CodeGenerationException var9) {
            throw new AopConfigException("Could not generate CGLIB subclass of class [" + this.advised.getTarg
```

其中　前对　　ObjenesisCglibAopProxy（class ObjenesisCglibAopProxy extends CglibAopProxy ），proxyFactory.getProxy　　中会创　个AopProxy，　个AopProxy可　JdkDynamicAopProxy　ObjenesisCglibAopProxy　在ObjenesisCglibAopProxy对　，同　ProxyFactory在创　AopProxy　候会将　传　AopProxy，

在CglibAopProxy中　AdvisedSupport advised;属　就　ProxyFactory

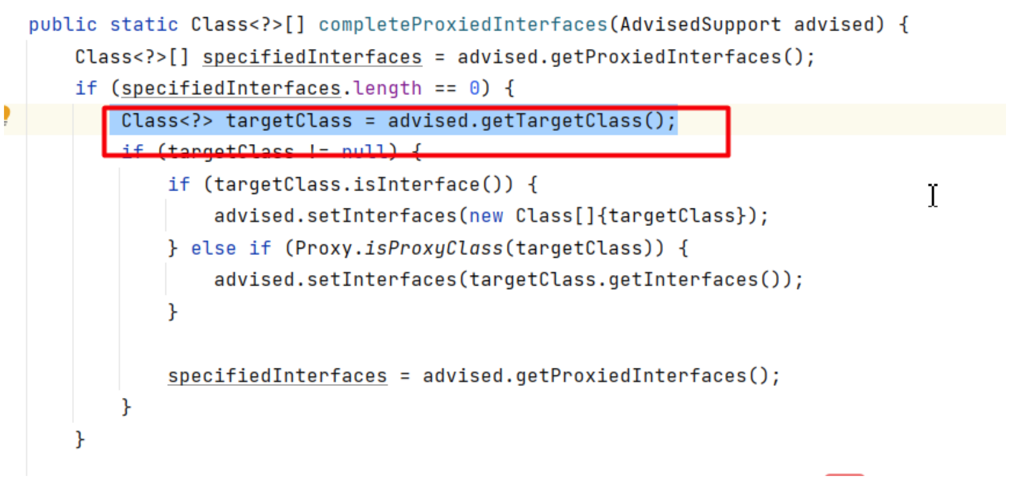


```
CglibAopProxy.class × | AopProxyUtils.class × | AdvisedSupport.class × | BeanFactoryTransactionAttributeSourceAdvisor.java × | ObjenesisCglibAopProxy.clas
ownloading sources...
class CglibAopProxy implements AopProxy, Serializable {
    private static final int AOP_PROXY = 0;
    private static final int INVOKE_TARGET = 1;
    private static final int NO_OVERRIDE = 2;
    private static final int DISPATCH_TARGET = 3;
    private static final int DISPATCH_ADVISED = 4;
    private static final int INVOKE_EQUALS = 5;
    private static final int INVOKE_HASHCODE = 6;
    protected static final Log logger = LogFactory.getLog(CglibAopProxy.class);
    private static final Map<Class<?>, Boolean> validatedClasses = new WeakHashMap();
    protected final AdvisedSupport advised;   advised: "org.springframework.aop.framework.ProxyFactory: 0 interfaces []; 1 a
    protected Object[] constructorArgs;   constructorArgs: null
    protected Class<?>[] constructorArgTypes;   constructorArgTypes: null
```

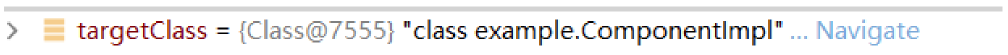
因　之　　this.advised 就　ProxyFactory对

enhancer.setInterfaces(AopProxyUtils.*completeProxiedInterfaces*(this.advised));

在completeProxiedInterfaces　　中会advised.getTargetClass();　取到　前创　　Bean　class，



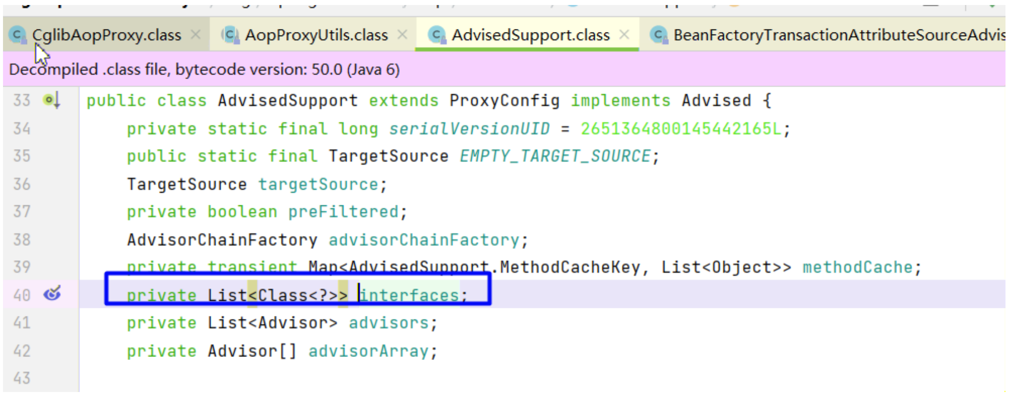
```
public static Class<?>[] completeProxiedInterfaces(AdvisedSupport advised) {
    Class<?>[] specifiedInterfaces = advised.getProxiedInterfaces();
    if (specifiedInterfaces.length == 0) {
        Class<?> targetClass = advised.getTargetClass();
        if (targetClass != null) {
            if (targetClass.isInterface()) {
                advised.setInterfaces(new Class[]{targetClass});
            } else if (Proxy.isProxyClass(targetClass)) {
                advised.setInterfaces(targetClass.getInterfaces());
            }

            specifiedInterfaces = advised.getProxiedInterfaces();
        }
    }
```



```
> targetClass = {Class@7555} "class example.ComponentImpl" ... Navigate
```

ProxyFactory中 targetClass 在上 wrapIfNecessary 中 了createProxy 中创 了ProxyFactory，在创 ProxyFactory 候 了 些属

在上 completeProxiedInterfaces 中，getTargetClasss之后 又将 个targetClass 到 advised interfaces属 中，也就 advised.setInterfaces(new Class[]{targetClass});
对


CglibAopProxy.class × | AopProxyUtils.class × | AdvisedSupport.class × | BeanFactoryTransactionAttributeSourceAdvis

Decompiled .class file, bytecode version: 50.0 (Java 6)

```java
public class AdvisedSupport extends ProxyConfig implements Advised {
    private static final long serialVersionUID = 2651364800145442165L;
    public static final TargetSource EMPTY_TARGET_SOURCE;
    TargetSource targetSource;
    private boolean preFiltered;
    AdvisorChainFactory advisorChainFactory;
    private transient Map<AdvisedSupport.MethodCacheKey, List<Object>> methodCache;
    private List<Class<?>> interfaces;
    private List<Advisor> advisors;
    private Advisor[] advisorArray;
```

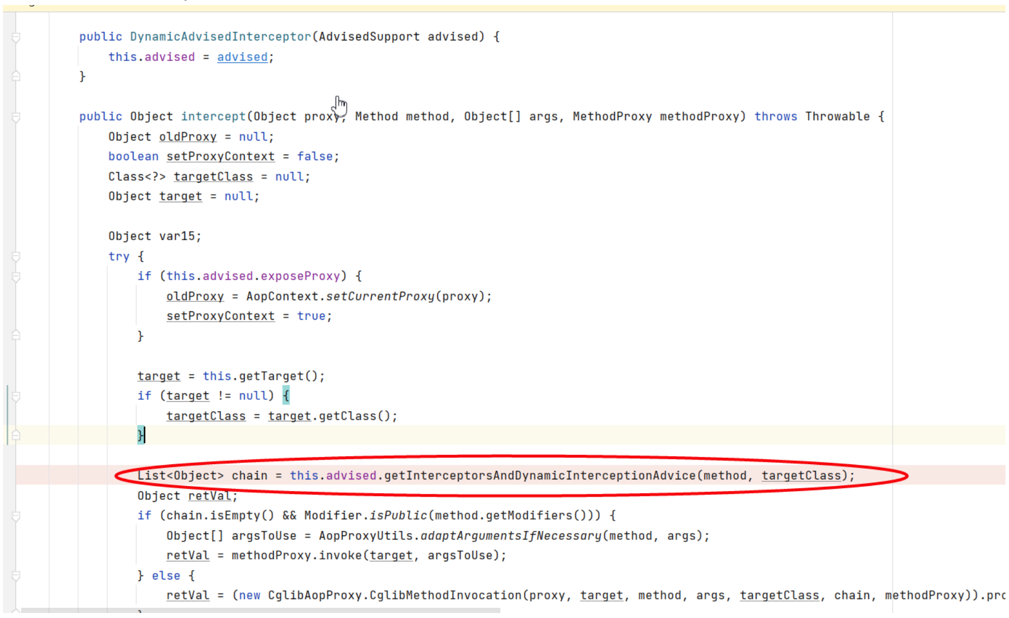
个interfaces会 作为enhancer interfaces属
enhancer.setInterfaces(AopProxyUtils.*completeProxiedInterfaces*(this.advised));

后又 enhancer 属

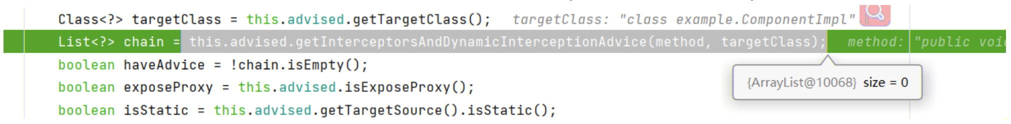
```java
Callback[] callbacks = this.getCallbacks(rootClass);
Class<?>[] types = new Class[callbacks.length];

for(x = 0; x < types.length; ++x) {
 types[x] = callbacks[x].getClass();
}

enhancer.setCallbackFilter(new CglibAopProxy.ProxyCallbackFilter(this.advised.getConfiguratior
OnlyCopy(), this.fixedInterceptorMap, this.fixedInterceptorOffset));
```

getCallbacks 中会 个 interceptor CglibAopProxy.DynamicAdvisedInterceptor(this.advised);

在 个 interceptor中会 个


```java
    public DynamicAdvisedInterceptor(AdvisedSupport advised) {
        this.advised = advised;
    }

    public Object intercept(Object proxy, Method method, Object[] args, MethodProxy methodProxy) throws Throwable {
        Object oldProxy = null;
        boolean setProxyContext = false;
        Class<?> targetClass = null;
        Object target = null;

        Object var15;
        try {
            if (this.advised.exposeProxy) {
                oldProxy = AopContext.setCurrentProxy(proxy);
                setProxyContext = true;
            }

            target = this.getTarget();
            if (target != null) {
                targetClass = target.getClass();
            }

            List<Object> chain = this.advised.getInterceptorsAndDynamicInterceptionAdvice(method, targetClass);
            Object retVal;
            if (chain.isEmpty() && Modifier.isPublic(method.getModifiers())) {
                Object[] argsToUse = AopProxyUtils.adaptArgumentsIfNecessary(method, args);
                retVal = methodProxy.invoke(target, argsToUse);
            } else {
                retVal = (new CglibAopProxy.CglibMethodInvocation(proxy, target, method, args, targetClass, chain, methodProxy)).prc
```

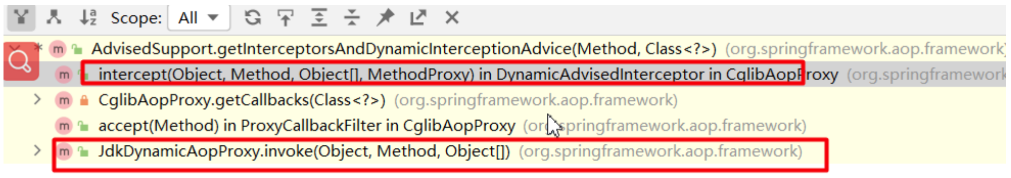
上 getInterceptorsAndDynamicInterceptionAdvice size为0，也就 将TransactionInterceptor作为interceptor， 就会导 cglib代 对 其 中并 TransactionInterceptor对 作为Interceptor



```
Class<?> targetClass = this.advised.getTargetClass();   targetClass: "class example.ComponentImpl"
List<?> chain = this.advised.getInterceptorsAndDynamicInterceptionAdvice(method, targetClass);   method: "public voi
boolean haveAdvice = !chain.isEmpty();
boolean exposeProxy = this.advised.isExposeProxy();
boolean isStatic = this.advised.getTargetSource().isStatic();
```

{ArrayList@10068} size = 0

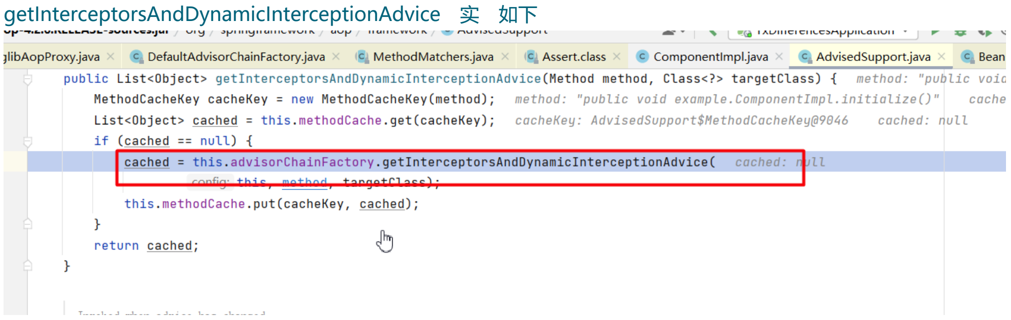
个getInterceptorsAndDynamicInterceptionAdvice ，他在两个地 8 到，分别对 Cglib AOp实 和JDK动 代 Aop实 ，也就



Scope: All

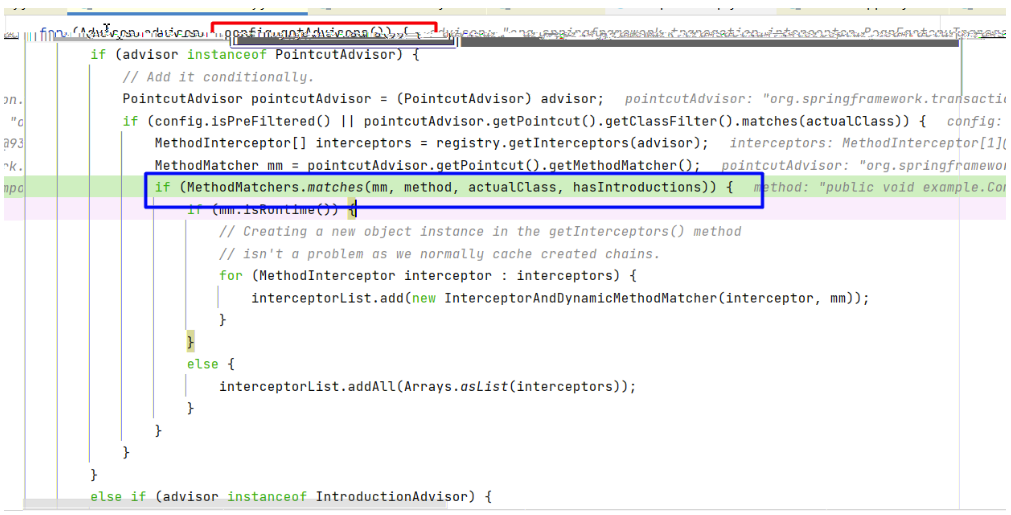
```
AdvisedSupport.getInterceptorsAndDynamicInterceptionAdvice(Method, Class<?>) (org.springframework.aop.framework)
  intercept(Object, Method, Object[], MethodProxy) in DynamicAdvisedInterceptor in CglibAopProxy (org.springframework
  CglibAopProxy.getCallbacks(Class<?>) (org.springframework.aop.framework)
  accept(Method) in ProxyCallbackFilter in CglibAopProxy (org.springframework.aop.framework)
  JdkDynamicAopProxy.invoke(Object, Method, Object[]) (org.springframework.aop.framework)
```

getInterceptorsAndDynamicInterceptionAdvice 实 如下



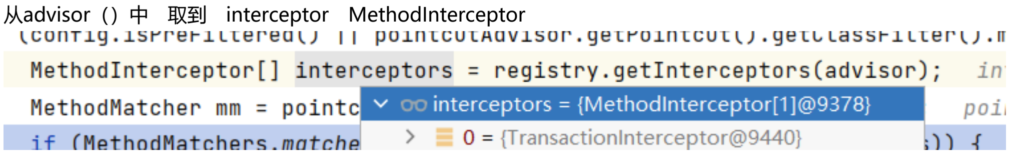
glibAopProxy.java × DefaultAdvisorChainFactory.java × MethodMatchers.java × Assert.class × ComponentImpl.java × AdvisedSupport.java × Bean

```
public List<Object> getInterceptorsAndDynamicInterceptionAdvice(Method method, Class<?> targetClass) {   method: "public voi
    MethodCacheKey cacheKey = new MethodCacheKey(method);   method: "public void example.ComponentImpl.initialize()"   cache
    List<Object> cached = this.methodCache.get(cacheKey);   cacheKey: AdvisedSupport$MethodCacheKey@9046   cached: null
    if (cached == null) {
        cached = this.advisorChainFactory.getInterceptorsAndDynamicInterceptionAdvice(   cached: null
                config: this, method, targetClass);
        this.methodCache.put(cacheKey, cached);
    }
    return cached;
}
```



```
for (Advisor advisor : config.getAdvisors()) {
    if (advisor instanceof PointcutAdvisor) {
        // Add it conditionally.
        PointcutAdvisor pointcutAdvisor = (PointcutAdvisor) advisor;   pointcutAdvisor: "org.springframework.transacti
        if (config.isPreFiltered() || pointcutAdvisor.getPointcut().getClassFilter().matches(actualClass)) {   config:
            MethodInterceptor[] interceptors = registry.getInterceptors(advisor);   interceptors: MethodInterceptor[1](
            MethodMatcher mm = pointcutAdvisor.getPointcut().getMethodMatcher();   pointcutAdvisor: "org.springframewo
            if (MethodMatchers.matches(mm, method, actualClass, hasIntroductions)) {   method: "public void example.Co
                if (mm.isRuntime()) {
                    // Creating a new object instance in the getInterceptors() method
                    // isn't a problem as we normally cache created chains.
                    for (MethodInterceptor interceptor : interceptors) {
                        interceptorList.add(new InterceptorAndDynamicMethodMatcher(interceptor, mm));
                    }
                }
                else {
                    interceptorList.addAll(Arrays.asList(interceptors));
                }
            }
        }
    }
    else if (advisor instanceof IntroductionAdvisor) {
```

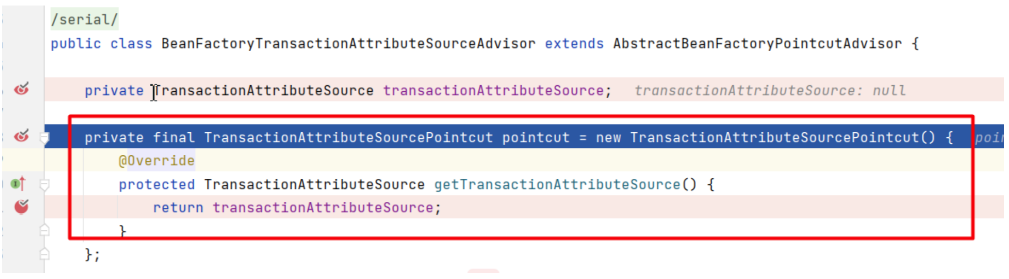
在上图中 们发 实 中 历 个Advisor，其中config就 ProxyFactory，其中从advisor（）中 取到 interceptor MethodInterceptor



```
(config.isPreFiltered() || pointcutAdvisor.getPointcut().getClassFilter().m
    MethodInterceptor[] interceptors = registry.getInterceptors(advisor);   in
    MethodMatcher mm = pointc
    if (MethodMatchers.matche                                             )) {
```

```
interceptors = {MethodInterceptor[1]@9378}
  0 = {TransactionInterceptor@9440}
```

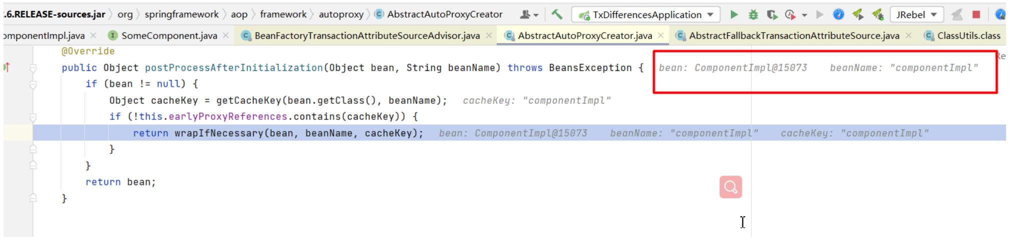
PointCut



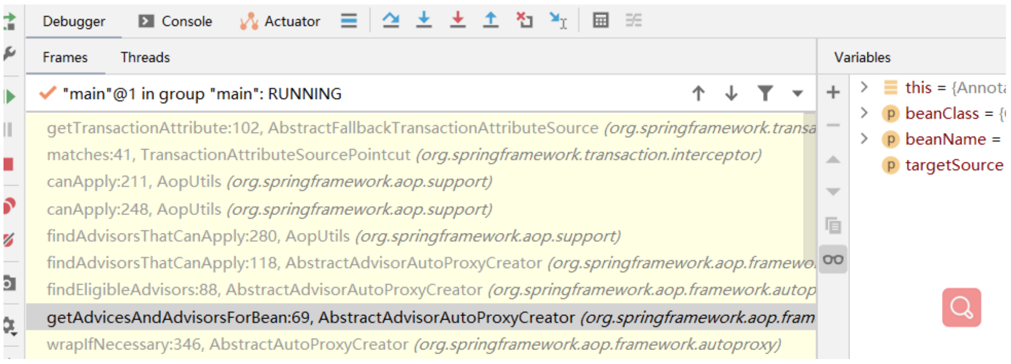
```java
/serial/
public class BeanFactoryTransactionAttributeSourceAdvisor extends AbstractBeanFactoryPointcutAdvisor {

    private TransactionAttributeSource transactionAttributeSource;   transactionAttributeSource: null

    private final TransactionAttributeSourcePointcut pointcut = new TransactionAttributeSourcePointcut() {   poi
        @Override
        protected TransactionAttributeSource getTransactionAttributeSource() {
            return transactionAttributeSource;
        }
    };
```

二

插　　，在wrapIfNecessary　　内会　　getAdvicesAndAdvisorsForBean

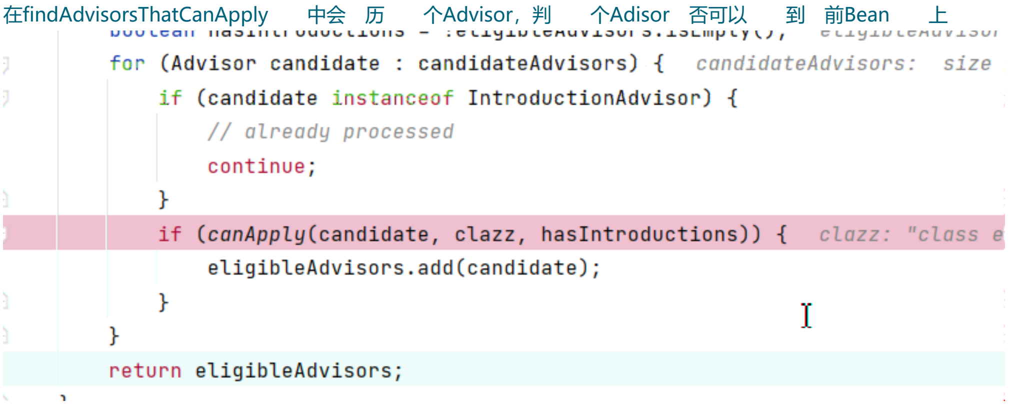


```java
@Override
public Object postProcessAfterInitialization(Object bean, String beanName) throws BeansException {   bean: ComponentImpl@15073    beanName: "componentImpl"
    if (bean != null) {
        Object cacheKey = getCacheKey(bean.getClass(), beanName);   cacheKey: "componentImpl"
        if (!this.earlyProxyReferences.contains(cacheKey)) {
            return wrapIfNecessary(bean, beanName, cacheKey);   bean: ComponentImpl@15073    beanName: "componentImpl"    cacheKey: "componentImpl"
        }
    }
    return bean;
}
```



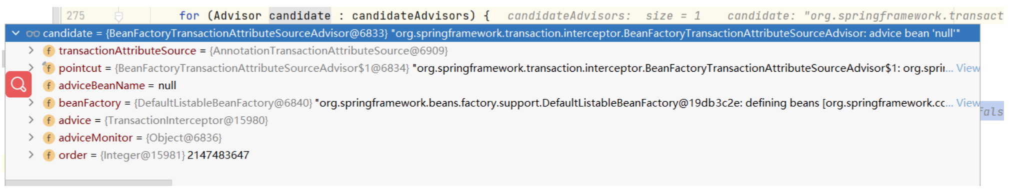
```
"main"@1 in group "main": RUNNING
getTransactionAttribute:102, AbstractFallbackTransactionAttributeSource (org.springframework.transa
matches:41, TransactionAttributeSourcePointcut (org.springframework.transaction.interceptor)
canApply:211, AopUtils (org.springframework.aop.support)
canApply:248, AopUtils (org.springframework.aop.support)
findAdvisorsThatCanApply:280, AopUtils (org.springframework.aop.support)
findAdvisorsThatCanApply:118, AbstractAdvisorAutoProxyCreator (org.springframework.aop.framewo
findEligibleAdvisors:88, AbstractAdvisorAutoProxyCreator (org.springframework.aop.framework.autop
getAdvicesAndAdvisorsForBean:69, AbstractAdvisorAutoProxyCreator (org.springframework.aop.fram
wrapIfNecessary:346, AbstractAutoProxyCreator (org.springframework.aop.framework.autoproxy)
```

在findAdvisorsThatCanApply　　中会　历　　个Advisor，判　　个Adisor　否可以　　到　前Bean　　上



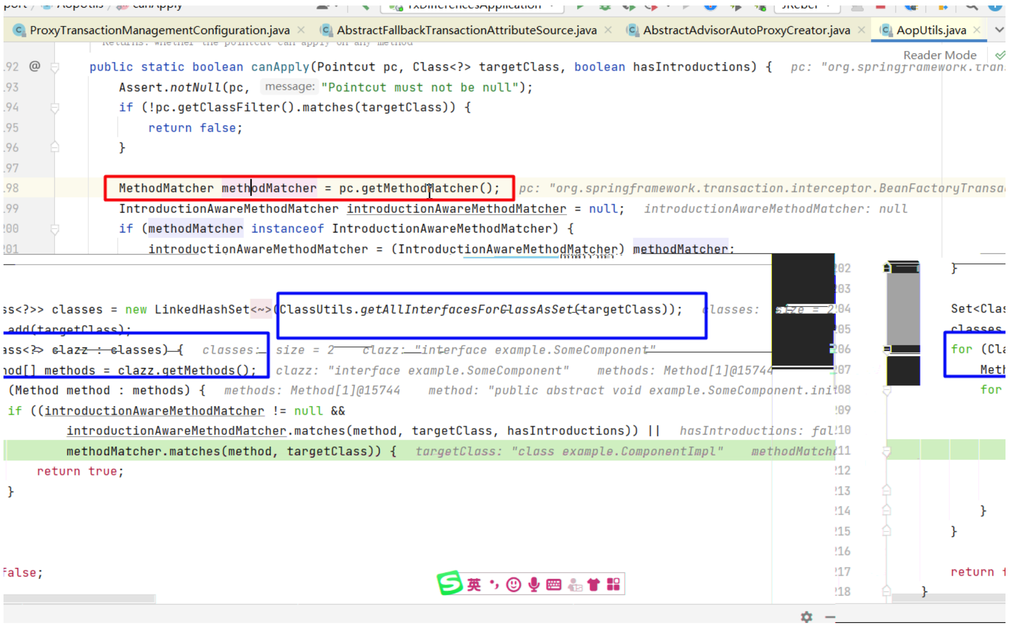
```java
for (Advisor candidate : candidateAdvisors) {   candidateAdvisors:  size
    if (candidate instanceof IntroductionAdvisor) {
        // already processed
        continue;
    }
    if (canApply(candidate, clazz, hasIntroductions)) {   clazz: "class e
        eligibleAdvisors.add(candidate);
    }
}
return eligibleAdvisors;
```

其中　　个Advisor就　BeanFactorytransactionAttributeSourceAdvisor，属　advice　MethodInterceptor，pointcut　TransactionAttributeSourcePointcut



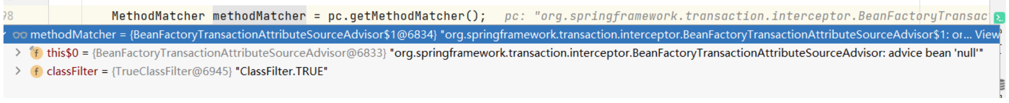
```
for (Advisor candidate : candidateAdvisors) {   candidateAdvisors:  size = 1    candidate: "org.springframework.transact
candidate = {BeanFactoryTransactionAttributeSourceAdvisor@6833} "org.springframework.transaction.interceptor.BeanFactoryTransactionAttributeSourceAdvisor: advice bean 'null'"
  transactionAttributeSource = {AnnotationTransactionAttributeSource@6909}
  pointcut = {BeanFactoryTransactionAttributeSourceAdvisor$1@6834} "org.springframework.transaction.interceptor.BeanFactoryTransactionAttributeSourceAdvisor$1: org.spri... View
  adviceBeanName = null
  beanFactory = {DefaultListableBeanFactory@6840} "org.springframework.beans.factory.support.DefaultListableBeanFactory@19db3c2e: defining beans [org.springframework.cc... View
  advice = {TransactionInterceptor@15980}
  adviceMonitor = {Object@6836}
  order = {Integer@15981} 2147483647
```

在canApply　　中会　对class　　Interfaces中　　个　　判　MethodMatcher　否match

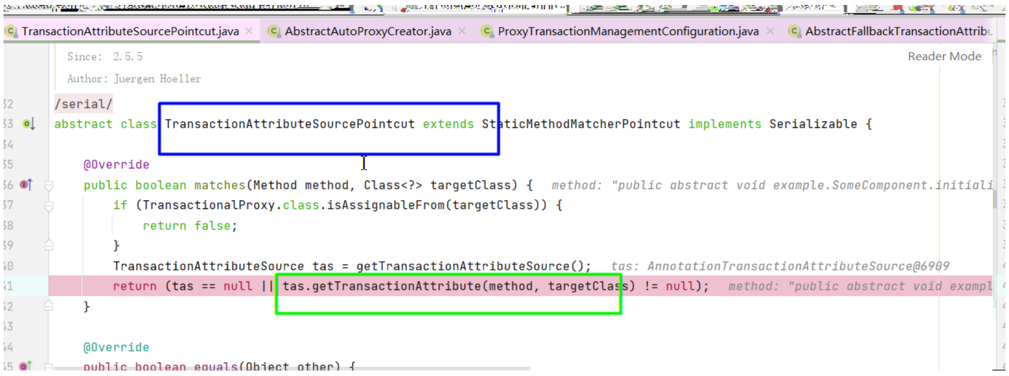


```
public static boolean canApply(Pointcut pc, Class<?> targetClass, boolean hasIntroductions) {   pc: "org.springframework.tran
    Assert.notNull(pc,  message: "Pointcut must not be null");
    if (!pc.getClassFilter().matches(targetClass)) {
        return false;
    }

    MethodMatcher methodMatcher = pc.getMethodMatcher();   pc: "org.springframework.transaction.interceptor.BeanFactoryTransa
    IntroductionAwareMethodMatcher introductionAwareMethodMatcher = null;   introductionAwareMethodMatcher: null
    if (methodMatcher instanceof IntroductionAwareMethodMatcher) {
        introductionAwareMethodMatcher = (IntroductionAwareMethodMatcher) methodMatcher;

ss<?>> classes = new LinkedHashSet<~>(ClassUtils.getAllInterfacesForClassAsSet(targetClass));   classes:  size = 2
add(targetClass);
ass<?> clazz : classes) {   classes:  size = 2   clazz: "interface example.SomeComponent"
hod[] methods = clazz.getMethods();   clazz: "interface example.SomeComponent"   methods: Method[1]@15744
(Method method : methods) {   methods: Method[1]@15744   method: "public abstract void example.SomeComponent.ini
if ((introductionAwareMethodMatcher != null &&
        introductionAwareMethodMatcher.matches(method, targetClass, hasIntroductions)) ||   hasIntroductions: fal
        methodMatcher.matches(method, targetClass)) {   targetClass: "class example.ComponentImpl"   methodMatche
    return true;
}

false;
```

否match　依　　判　　　上　否　@Transactional　　，　个methodMatcher如下

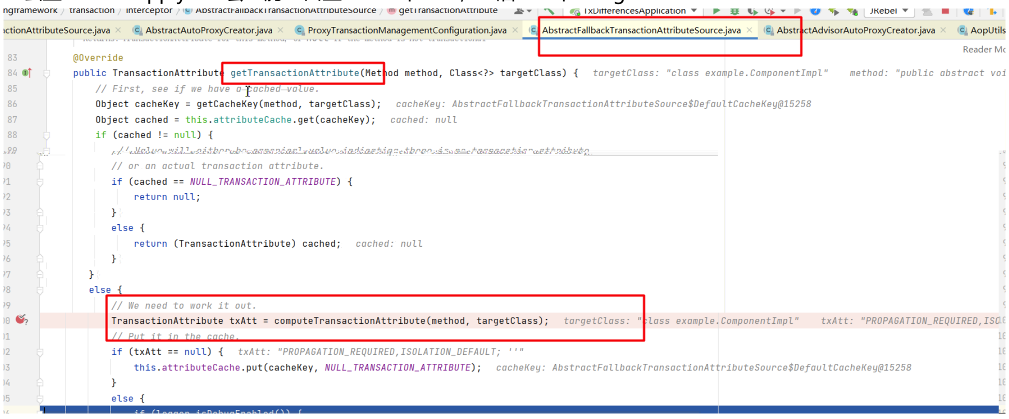


```
MethodMatcher methodMatcher = pc.getMethodMatcher();   pc: "org.springframework.transaction.interceptor.BeanFactoryTransac
∞ methodMatcher = {BeanFactoryTransactionAttributeSourceAdvisor$1@6834} "org.springframework.transaction.interceptor.BeanFactoryTransactionAttributeSourceAdvisor$1: or... View
 > this$0 = {BeanFactoryTransactionAttributeSourceAdvisor@6833} "org.springframework.transaction.interceptor.BeanFactoryTransactionAttributeSourceAdvisor: advice bean 'null'"
 > classFilter = {TrueClassFilter@6945} "ClassFilter.TRUE"
```



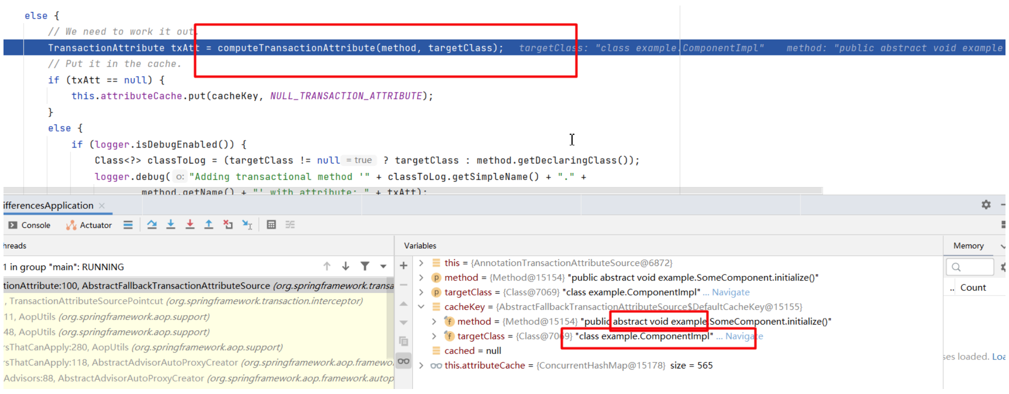
```
Since: 2.5.5
Author: Juergen Hoeller
/serial/
abstract class TransactionAttributeSourcePointcut extends StaticMethodMatcherPointcut implements Serializable {

    @Override
    public boolean matches(Method method, Class<?> targetClass) {   method: "public abstract void example.SomeComponent.initiali
        if (TransactionalProxy.class.isAssignableFrom(targetClass)) {
            return false;
        }
        TransactionAttributeSource tas = getTransactionAttributeSource();   tas: AnnotationTransactionAttributeSource@6909
        return (tas == null || tas.getTransactionAttribute(method, targetClass) != null);   method: "public abstract void exampl
    }

    @Override
    public boolean equals(Object other) {
```

假　　们在　口　　　上使　了@Transactional　　，　且　定使　Cglib代　（proxyTargetClass为true）
　么上　　canApply　　会　历　口上　　　个　，　后　　下　　getTransactionAttribute



```
@Override
public TransactionAttribute getTransactionAttribute(Method method, Class<?> targetClass) {   targetClass: "class example.ComponentImpl"   method: "public abstract voi
    // First, see if we have a cached value.
    Object cacheKey = getCacheKey(method, targetClass);   cacheKey: AbstractFallbackTransactionAttributeSource$DefaultCacheKey@15258
    Object cached = this.attributeCache.get(cacheKey);   cached: null
    if (cached != null) {
        // Value will either be canonical value indicating there is no transaction attribute,
        // or an actual transaction attribute.
        if (cached == NULL_TRANSACTION_ATTRIBUTE) {
            return null;
        }
        else {
            return (TransactionAttribute) cached;   cached: null
        }
    }
    else {
        // We need to work it out.
        TransactionAttribute txAtt = computeTransactionAttribute(method, targetClass);   targetClass: "class example.ComponentImpl"   txAtt: "PROPAGATION_REQUIRED,ISO
        // Put it in the cache.
        if (txAtt == null) {   txAtt: "PROPAGATION_REQUIRED,ISOLATION_DEFAULT; ''"
            this.attributeCache.put(cacheKey, NULL_TRANSACTION_ATTRIBUTE);   cacheKey: AbstractFallbackTransactionAttributeSource$DefaultCacheKey@15258
        }
        else {
```

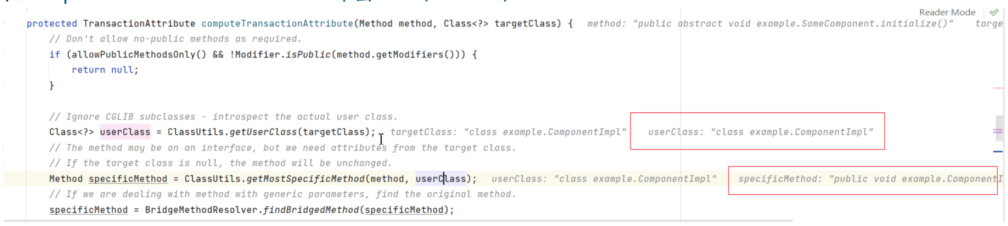
在　历　　　中会　历　口example.SomeComponent#initialize中　initialize　　，　后分　　个　　上@Transactional　　，　　下　　method
　存在abstract修



```
else {
    // We need to work it out.
    TransactionAttribute txAtt = computeTransactionAttribute(method, targetClass);   targetClass: "class example.ComponentImpl"    method: "public abstract void example
    // Put it in the cache.
    if (txAtt == null) {
        this.attributeCache.put(cacheKey, NULL_TRANSACTION_ATTRIBUTE);
    }
    else {
        if (logger.isDebugEnabled()) {
            Class<?> classToLog = (targetClass != null = true ? targetClass : method.getDeclaringClass());
            logger.debug( o: "Adding transactional method '" + classToLog.getSimpleName() + "." +
                    method.getName() + "' with attribute: " + txAtt);
```

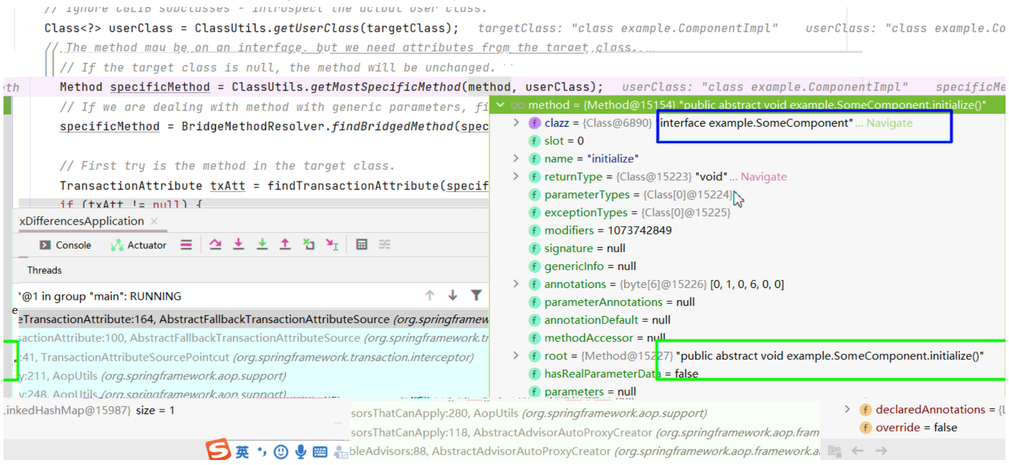
```
ifferencesApplication ×
Console   Actuator
hreads
1 in group "main": RUNNING
tionAttribute:100, AbstractFallbackTransactionAttributeSource (org.springframework.transa
, TransactionAttributeSourcePointcut (org.springframework.transaction.interceptor)
11, AopUtils (org.springframework.aop.support)
48, AopUtils (org.springframework.aop.support)
sThatCanApply:280, AopUtils (org.springframework.aop.support)
sThatCanApply:118, AbstractAdvisorAutoProxyCreator (org.springframework.aop.framewo
Advisors:88, AbstractAdvisorAutoProxyCreator (org.springframework.aop.framework.autop

Variables
> this = {AnnotationTransactionAttributeSource@6872}
> method = {Method@15154} "public abstract void example.SomeComponent.initialize()"
> targetClass = {Class@7069} "class example.ComponentImpl" ... Navigate
v cacheKey = {AbstractFallbackTransactionAttributeSource$DefaultCacheKey@15155}
    > method = {Method@15154} "public abstract void example.SomeComponent.initialize()"
    > targetClass = {Class@7069} "class example.ComponentImpl" ... Navigate
  cached = null
> this.attributeCache = {ConcurrentHashMap@15178}  size = 565

Memory
.. Count
ses loaded. Loa
```

在computeTransactionAttribute　　中会　　下　　　个



```
protected TransactionAttribute computeTransactionAttribute(Method method, Class<?> targetClass) {   method: "public abstract void example.SomeComponent.initialize()"    targe
    // Don't allow no-public methods as required.
    if (allowPublicMethodsOnly() && !Modifier.isPublic(method.getModifiers())) {
        return null;
    }

    // Ignore CGLIB subclasses - introspect the actual user class.
    Class<?> userClass = ClassUtils.getUserClass(targetClass);   targetClass: "class example.ComponentImpl"    userClass: "class example.ComponentImpl"
    // The method may be on an interface, but we need attributes from the target class.
    // If the target class is null, the method will be unchanged.
    Method specificMethod = ClassUtils.getMostSpecificMethod(method, userClass);   userClass: "class example.ComponentImpl"    specificMethod: "public void example.ComponentI
    // If we are dealing with method with generic parameters, find the original method.
    specificMethod = BridgeMethodResolver.findBridgedMethod(specificMethod);
```

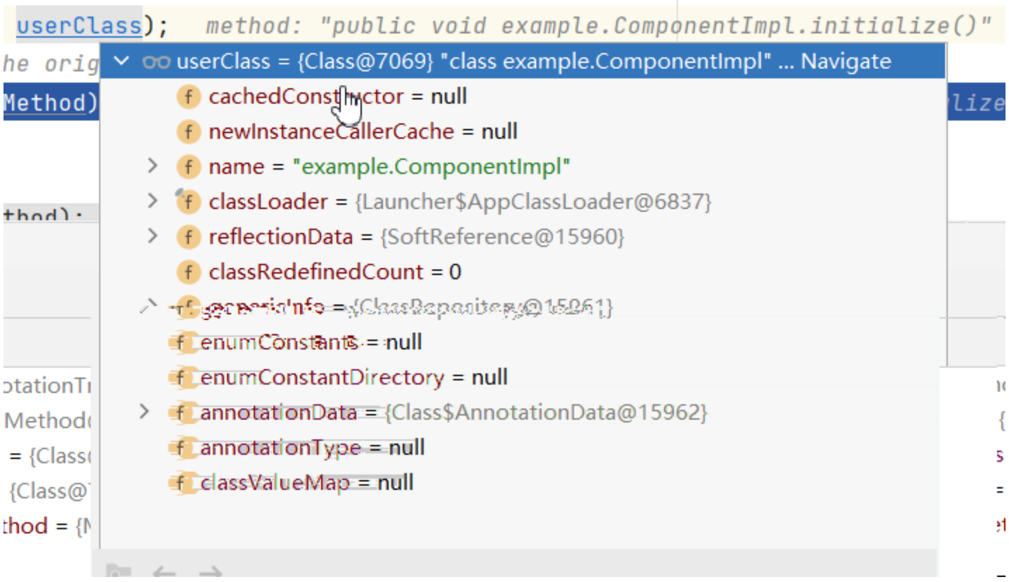
其中ClassUtils.*getUserClass*(targetClass);　　　targetClass就　Bean　实　　example.ComponentImpl
2
　后method　　　　口class　取到　method对　，



```
    // Ignore CGLIB subclasses - introspect the actual user class.
    Class<?> userClass = ClassUtils.getUserClass(targetClass);   targetClass: "class example.ComponentImpl"    userClass: "class example.Co
    // The method may be on an interface, but we need attributes from the target class.
    // If the target class is null, the method will be unchanged.
    Method specificMethod = ClassUtils.getMostSpecificMethod(method, userClass);   userClass: "class example.ComponentImpl"    specificMe
    // If we are dealing with method with generic parameters, fi
    specificMethod = BridgeMethodResolver.findBridgedMethod(spec

    // First try is the method in the target class.
    TransactionAttribute txAtt = findTransactionAttribute(specif
    if (txAtt != null) {
```

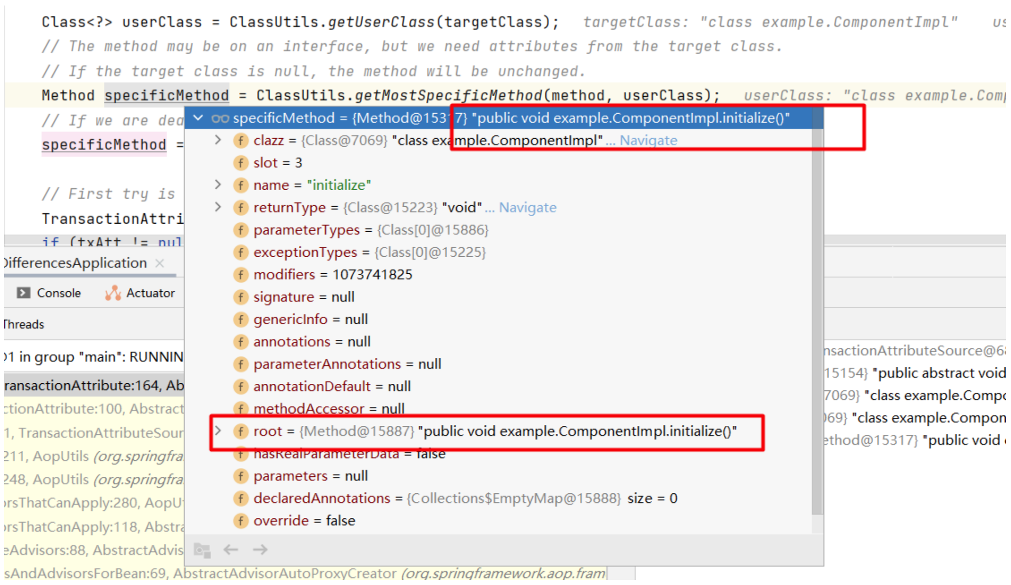
```
xDifferencesApplication ×
Console   Actuator
Threads
'@1 in group "main": RUNNING
eTransactionAttribute:164, AbstractFallbackTransactionAttributeSource (org.springframew
actionAttribute:100, AbstractFallbackTransactionAttributeSource (org.springframework.tr
:41, TransactionAttributeSourcePointcut (org.springframework.transaction.interceptor)
y:211, AopUtils (org.springframework.aop.support)
y:248, AopUtils (org.springframework.aop.support)
sorsThatCanApply:280, AopUtils (org.springframework.aop.support)
sorsThatCanApply:118, AbstractAdvisorAutoProxyCreator (org.springframework.aop.fram
bleAdvisors:88, AbstractAdvisorAutoProxyCreator (org.springframework.aop.framework.a
.inkedHashMap@15987}  size = 1

v method = {Method@15154} "public abstract void example.SomeComponent.initialize()"
    > clazz = {Class@6890} "interface example.SomeComponent" ... Navigate
      slot = 0
    > name = "initialize"
    > returnType = {Class@15223} "void" ... Navigate
      parameterTypes = {Class[0]@15224}
      exceptionTypes = {Class[0]@15225}
      modifiers = 1073742849
      signature = null
      genericInfo = null
    > annotations = {byte[6]@15226} [0, 1, 0, 6, 0, 0]
      parameterAnnotations = null
      annotationDefault = null
      methodAccessor = null
    > root = {Method@15227} "public abstract void example.SomeComponent.initialize()"
      hasRealParameterData = false
      parameters = null
    > declaredAnnotations = {L
      override = false
```

userClass



```
userClass);   method: "public void example.ComponentImpl.initialize()"
v userClass = {Class@7069} "class example.ComponentImpl" ... Navigate
      cachedConstructor = null
      newInstanceCallerCache = null
    > name = "example.ComponentImpl"
    > classLoader = {Launcher$AppClassLoader@6837}
    > reflectionData = {SoftReference@15960}
      classRedefinedCount = 0
    > genericInfo = {ClassRepository@15961}
      enumConstants = null
      enumConstantDirectory = null
    > annotationData = {Class$AnnotationData@15962}
      annotationType = null
      classValueMap = null
```

ClassUtils.*getMostSpecificMethod*(method, userClass);　　就　在　口　实　　上　取　　　　实
　以specificMethod就　下　　个具体　实



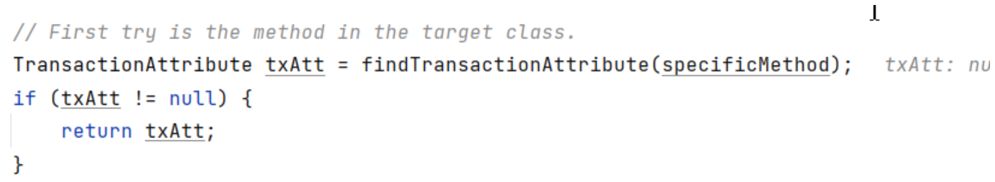
```
Class<?> userClass = ClassUtils.getUserClass(targetClass);   targetClass: "class example.ComponentImpl"    us
// The method may be on an interface, but we need attributes from the target class.
// If the target class is null, the method will be unchanged.
Method specificMethod = ClassUtils.getMostSpecificMethod(method, userClass);   userClass: "class example.Com
// If we are dea
specificMethod =

// First try is
TransactionAttri
if (txAtt != nul
```

```
specificMethod = {Method@153?7} "public void example.ComponentImpl.initialize()"
  clazz = {Class@7069} "class example.ComponentImpl" ... Navigate
  slot = 3
  name = "initialize"
  returnType = {Class@15223} "void" ... Navigate
  parameterTypes = {Class[0]@15886}
  exceptionTypes = {Class[0]@15225}
  modifiers = 1073741825
  signature = null
  genericInfo = null
  annotations = null
  parameterAnnotations = null
  annotationDefault = null
  methodAccessor = null
  root = {Method@15887} "public void example.ComponentImpl.initialize()"
  hasRealParameterData = false
  parameters = null
  declaredAnnotations = {Collections$EmptyMap@15888} size = 0
  override = false
```

```
DifferencesApplication
Console   Actuator
Threads
1 in group "main": RUNNIN
ransactionAttribute:164, Ab
ctionAttribute:100, Abstract
1, TransactionAttributeSour
211, AopUtils (org.springfra
248, AopUtils (org.springfra
rsThatCanApply:280, AopU
rsThatCanApply:118, Abstra
eAdvisors:88, AbstractAdvis
sAndAdvisorsForBean:69, AbstractAdvisorAutoProxyCreator (org.springframework.aop.fram
```

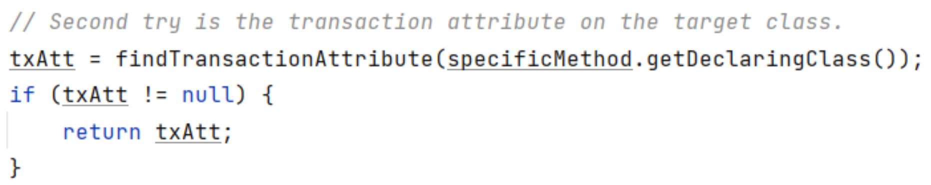
```
nsactionAttributeSource@6
15154} "public abstract void
7069} "class example.Compo
069} "class example.Compon
ethod@15317} "public void
```

后 们 先在实 上寻 TransactionAttribute，如 实 上 Attribute则 存在事务



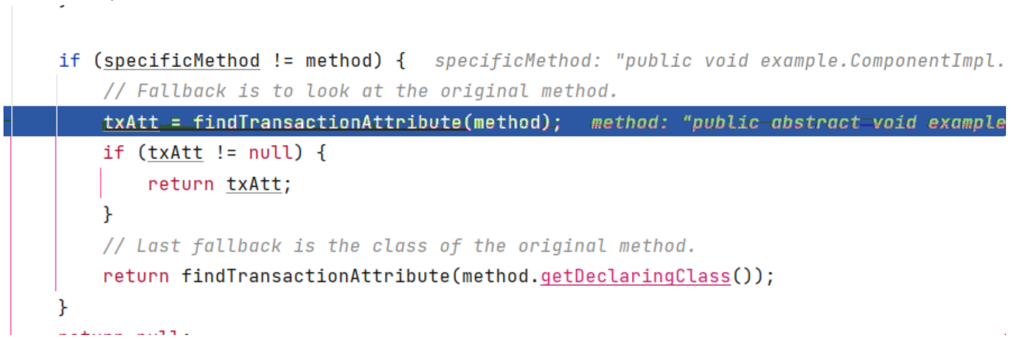
```
// First try is the method in the target class.
TransactionAttribute txAtt = findTransactionAttribute(specificMethod);   txAtt: nu
if (txAtt != null) {
    return txAtt;
}
```

如 实 上 ，则 们在实 上寻 TransactionAttribute



```
// Second try is the transaction attribute on the target class.
txAtt = findTransactionAttribute(specificMethod.getDeclaringClass());
if (txAtt != null) {
    return txAtt;
}
```

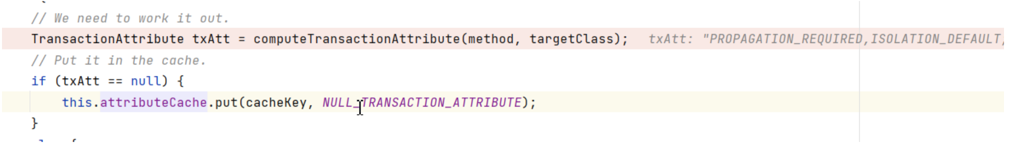
后，如 specificMethod不 于method，则就在method上 ， method就 口 中 abstract method



```
if (specificMethod != method) {   specificMethod: "public void example.ComponentImpl.
    // Fallback is to look at the original method.
    txAtt = findTransactionAttribute(method);   method: "public abstract void example
    if (txAtt != null) {
        return txAtt;
    }
    // Last fallback is the class of the original method.
    return findTransactionAttribute(method.getDeclaringClass());
}
```

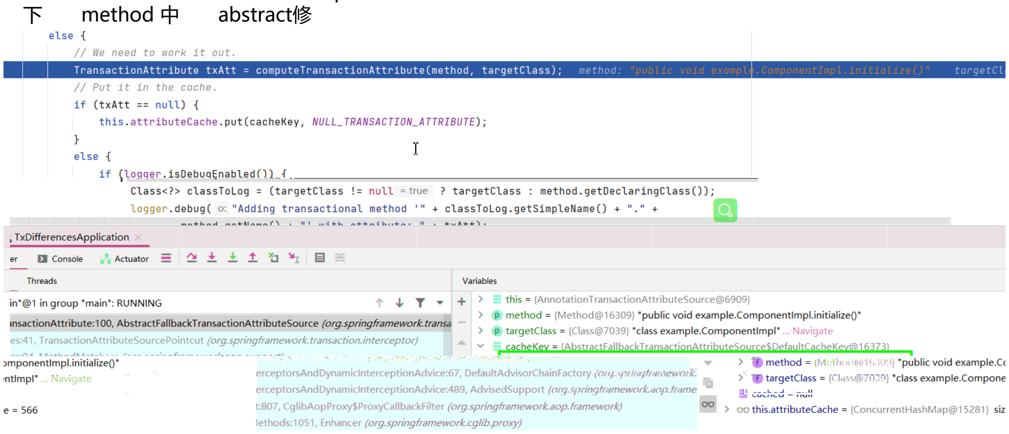
因 口 上存在 候， 们以 口中 取TransactionAttribute 候 可以 取到TransactionAttribute

在Bean 创 中会分 Bean 口上 个 ， 个 在实 上 否 ，在实 了上 否 ，在 口上 否 ， 会 到 org.springframework.transaction.interceptor.AbstractFallbackTransactionAttributeSource#attributeCache属 中，因 个 位 否 事务



```
// We need to work it out.
TransactionAttribute txAtt = computeTransactionAttribute(method, targetClass);   txAtt: "PROPAGATION_REQUIRED,ISOLATION_DEFAULT,
// Put it in the cache.
if (txAtt == null) {
    this.attributeCache.put(cacheKey, NULL_TRANSACTION_ATTRIBUTE);
}
```

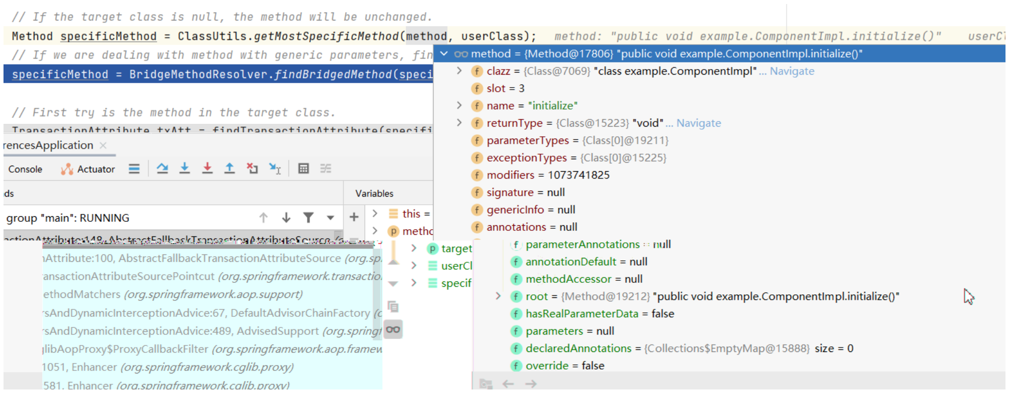
在使 了cglib代 况下 如 们以ComponentImpl 中 initialize 去分 否 TransactionAttribute，会 回null，也就 事务，
4
下 method 中 abstract修



```
else {
    // We need to work it out.
    TransactionAttribute txAtt = computeTransactionAttribute(method, targetClass);   method: "public void example.ComponentImpl.initialize()"   targetCl
    // Put it in the cache.
    if (txAtt == null) {
        this.attributeCache.put(cacheKey, NULL_TRANSACTION_ATTRIBUTE);
    }
    else {
        if (logger.isDebugEnabled()) {
            Class<?> classToLog = (targetClass != null = true ? targetClass : method.getDeclaringClass());
            logger.debug( o: "Adding transactional method '" + classToLog.getSimpleName() + "." +
```

```
TxDifferencesApplication
er   Console   Actuator
Threads
in"@1 in group "main": RUNNING
ınsactionAttribute:100, AbstractFallbackTransactionAttributeSource (org.springframework.transa
es:41, TransactionAttributeSourcePointcut (org.springframework.transaction.interceptor)
erceptorsAndDynamicInterceptionAdvice:67, DefaultAdvisorChainFactory (org.springframework.
erceptorsAndDynamicInterceptionAdvice:489, AdvisedSupport (org.springframework.aop.frame
t:807, CglibAopProxy$ProxyCallbackFilter (org.springframework.aop.framework)
lethods:1051, Enhancer (org.springframework.cglib.proxy)

Variables
> this = {AnnotationTransactionAttributeSource@6909}
> method = {Method@16309} "public void example.ComponentImpl.initialize()"
> targetClass = {Class@7039} "class example.ComponentImpl" ... Navigate
v cacheKey = {AbstractFallbackTransactionAttributeSource$DefaultCacheKey@16373}
    > method = {Method@16309} "public void example.Co
    > targetClass = {Class@7039} "class example.Compone
      cached = null
> this.attributeCache = {ConcurrentHashMap@15281} siz
```

为什么会 不到事务 ？ 个 候computeTransactionAttribute 入参method

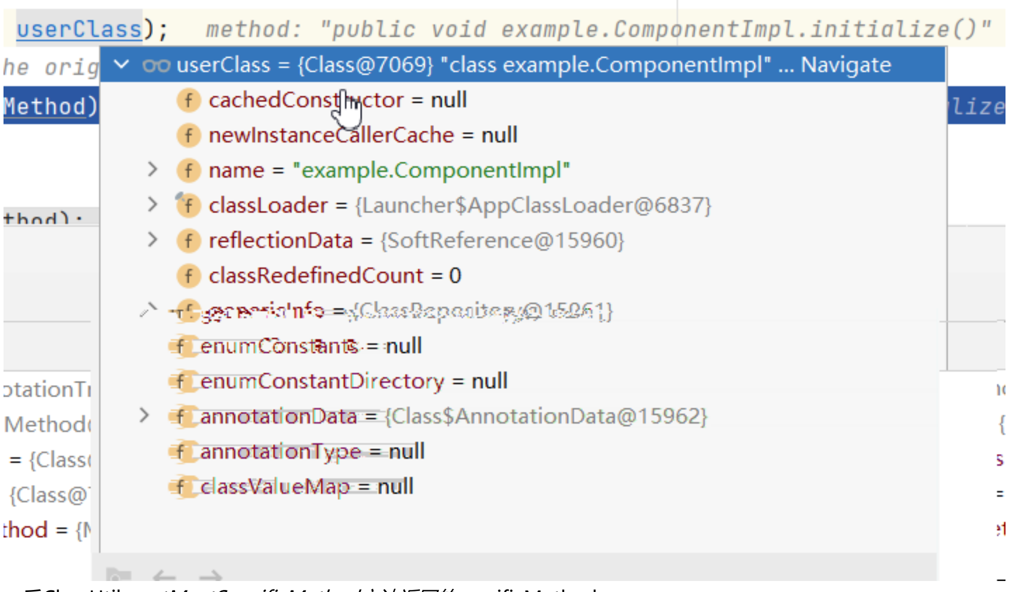


```
// If the target class is null, the method will be unchanged.
Method specificMethod = ClassUtils.getMostSpecificMethod(method, userClass);   method: "public void example.ComponentImpl.initialize()"   userCl
// If we are dealing with method with generic parameters, fin
specificMethod = BridgeMethodResolver.findBridgedMethod(speci

// First try is the method in the target class.
```

```
v method = {Method@17806} "public void example.ComponentImpl.initialize()"
  > clazz = {Class@7069} "class example.ComponentImpl" ... Navigate
    slot = 3
  > name = "initialize"
  > returnType = {Class@15223} "void" ... Navigate
    parameterTypes = {Class[0]@19211}
    exceptionTypes = {Class[0]@15225}
    modifiers = 1073741825
    signature = null
    genericInfo = null
    annotations = null
    parameterAnnotations = null
    annotationDefault = null
    methodAccessor = null
  > root = {Method@19212} "public void example.ComponentImpl.initialize()"
    hasRealParameterData = false
    parameters = null
    declaredAnnotations = {Collections$EmptyMap@15888}  size = 0
    override = false
```

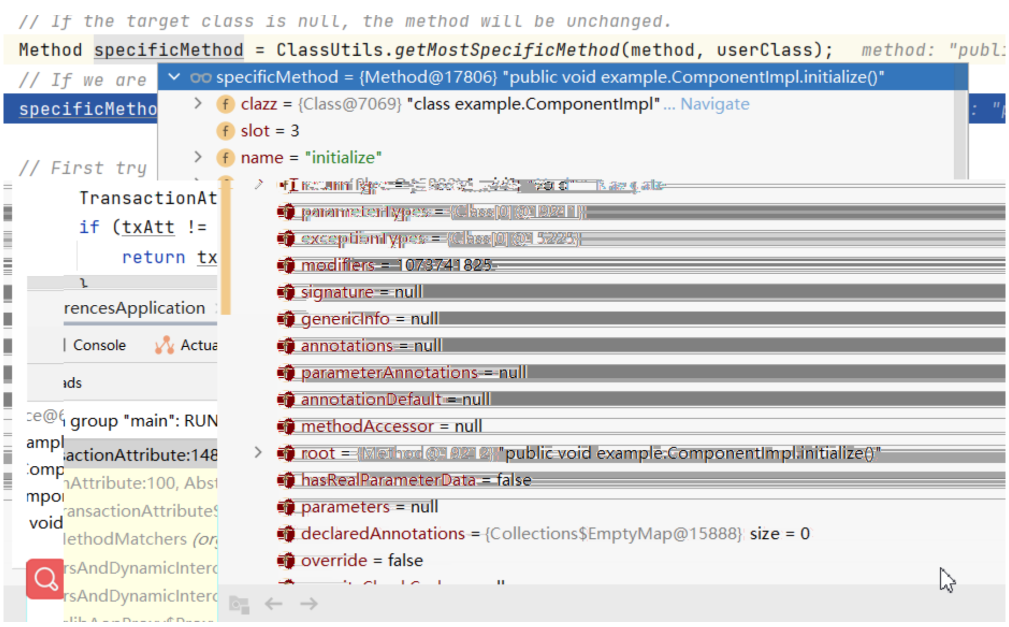
```
rencesApplication
Console   Actuator
ds
group "main": RUNNING
ıAttribute:100, AbstractFallbackTransactionAttributeSource (org.sp
ansactionAttributeSourcePointcut (org.springframework.transactio
ethodMatchers (org.springframework.aop.support)
sAndDynamicInterceptionAdvice:67, DefaultAdvisorChainFactory (c
sAndDynamicInterceptionAdvice:489, AdvisedSupport (org.spring
libAopProxy$ProxyCallbackFilter (org.springframework.aop.framew
1051, Enhancer (org.springframework.cglib.proxy)
581, Enhancer (org.springframework.cglib.proxy)
```

userClass



```
userClass);   method: "public void example.ComponentImpl.initialize()"
v userClass = {Class@7069} "class example.ComponentImpl" ... Navigate
    cachedConstructor = null
    newInstanceCallerCache = null
  > name = "example.ComponentImpl"
  > classLoader = {Launcher$AppClassLoader@6837}
  > reflectionData = {SoftReference@15960}
    classRedefinedCount = 0
  > genericInfo = {ClassRepository@15961}
    enumConstants = null
    enumConstantDirectory = null
  > annotationData = {Class$AnnotationData@15962}
    annotationType = null
    classValueMap = null
```

后ClassUtils.*getMostSpecificMethod方法返回的*specificMethod

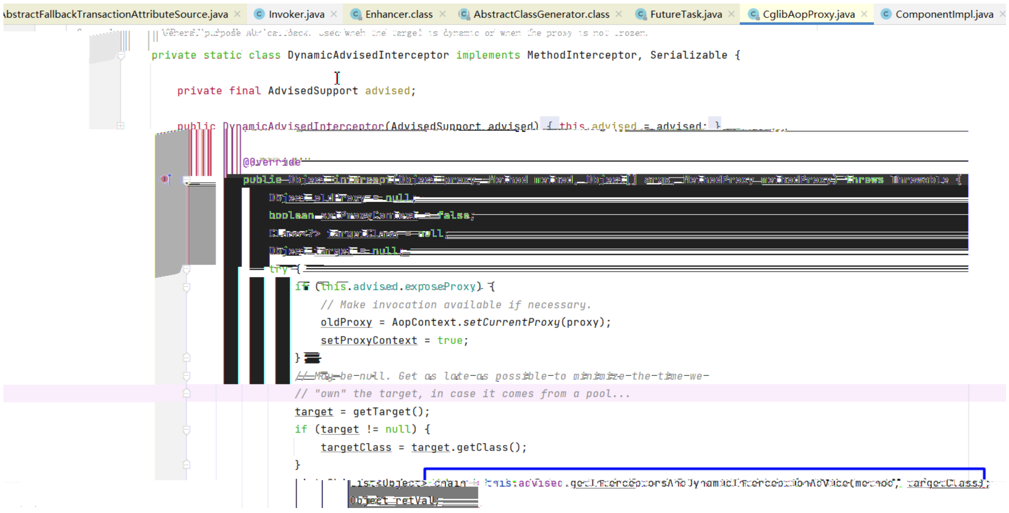


```
// If the target class is null, the method will be unchanged.
Method specificMethod = ClassUtils.getMostSpecificMethod(method, userClass);   method: "publ
// If we are
specificMetho

// First try
    TransactionAt
    if (txAtt !=
        return tx
```

```
specificMethod = {Method@17806} "public void example.ComponentImpl.initialize()"
  clazz = {Class@7069} "class example.ComponentImpl" ... Navigate
  slot = 3
  name = "initialize"
  parameterTypes = {Class[0]@
  exceptionTypes = {Class[0]@
  modifiers = 1073741825
  signature = null
  genericInfo = null
  annotations = null
  parameterAnnotations = null
  annotationDefault = null
  methodAccessor = null
  root = {Method@     } "public void example.ComponentImpl.initialize()"
  hasRealParameterData = false
  parameters = null
  declaredAnnotations = {Collections$EmptyMap@15888} size = 0
  override = false
```

后 们就会发 specificMethod 于method， 个实 上并 @Transactional ，因 不到TransactionalAttribute

们 Cglib对 候会 DynamicAdvisedInterceptor intercept ，在内 会 取匹 advisor advice，因为对于事务 Advisor ， 到 上 TransactionalAttribute 以 cglibg 回 chain size为0，也就 TransactionalInterceptor，因 Cglib 代 对 事务



```java
// General purpose AOP callback. Used when the target is dynamic or when the proxy is not frozen.
private static class DynamicAdvisedInterceptor implements MethodInterceptor, Serializable {

    private final AdvisedSupport advised;

    public DynamicAdvisedInterceptor(AdvisedSupport advised) { this.advised = advised; }

    @Override
    public Object intercept(Object proxy, Method method, Object[] args, MethodProxy methodProxy) throws Throwable {
        Object oldProxy = null;
        boolean setProxyContext = false;
        Class<?> targetClass = null;
        Object target = null;
        try {
            if (this.advised.exposeProxy) {
                // Make invocation available if necessary.
                oldProxy = AopContext.setCurrentProxy(proxy);
                setProxyContext = true;
            }
            // May be null. Get as late as possible to minimize the time we
            // "own" the target, in case it comes from a pool...
            target = getTarget();
            if (target != null) {
                targetClass = target.getClass();
            }
            List<Object> chain = this.advised.getInterceptorsAndDynamicInterceptionAdvice(method, targetClass);
            Object retVal;
```

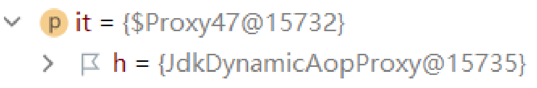
对于Cglib代 ，在wrapIfNecessary 中会 getProxy，在getProxy 中会使 enhancer.createClass();，

在 class 中会 个 ，org.springframework.cglib.proxy.Enhancer#getMethods(java.lang.Class, java.lang.Class[], java.util.List, java.util.List, java.util.Set)

```
1
2 computeTransactionAttribute:148, AbstractFallbackTransactionAttributeSource (org.springframework.transaction.interceptor)
3 getTransactionAttribute:100, AbstractFallbackTransactionAttributeSource (org.springframework.transaction.interceptor)
4 matches:41, TransactionAttributeSourcePointcut (org.springframework.transaction.interceptor)
5 matches:94, MethodMatchers (org.springframework.aop.support)
```

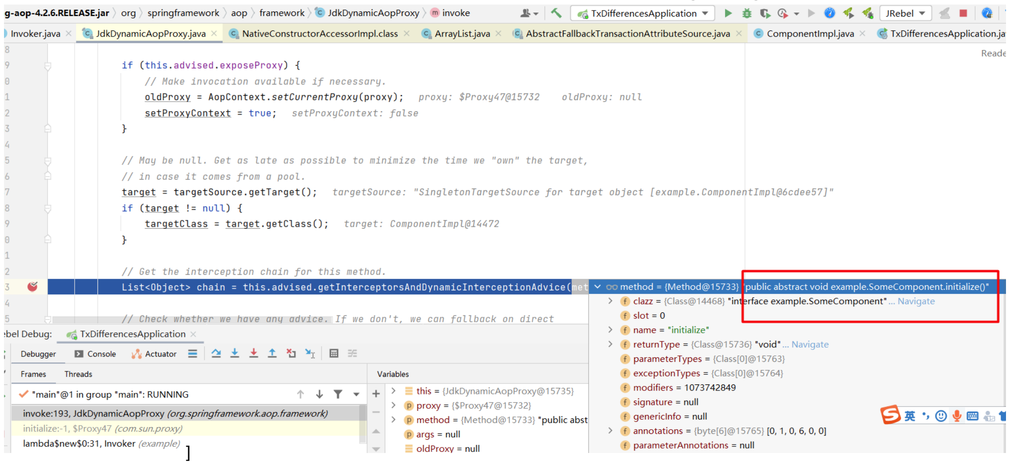
```
getInterceptorsAndDynamicInterceptionAdvice:67, DefaultAdvisorChainFactory (org.springframework.aop.framework)
getInterceptorsAndDynamicInterceptionAdvice:489, AdvisedSupport (org.springframework.aop.framework)
accept:807, CglibAopProxy$ProxyCallbackFilter (org.springframework.aop.framework)
emitMethods:1051, Enhancer (org.springframework.cglib.proxy)
generateClass:581, Enhancer (org.springframework.cglib.proxy)
generateClass:33, TransformingClassGenerator (org.springframework.cglib.transform)
generate:25, DefaultGeneratorStrategy (org.springframework.cglib.core)
generate:990, CglibAopProxy$ClassLoaderAwareUndeclaredThrowableStrategy (org.springframework.aop.framework)
generate:312, AbstractClassGenerator (org.springframework.cglib.core)
generate:445, Enhancer (org.springframework.cglib.proxy)
apply:85, AbstractClassGenerator$ClassLoaderData$3 (org.springframework.cglib.core)
apply:83, AbstractClassGenerator$ClassLoaderData$3 (org.springframework.cglib.core)
call:54, LoadingCache$2 (org.springframework.cglib.core.internal)
run:266, FutureTask (java.util.concurrent)
createEntry:61, LoadingCache (org.springframework.cglib.core.internal)
get:34, LoadingCache (org.springframework.cglib.core.internal)
get:105, AbstractClassGenerator$ClassLoaderData (org.springframework.cglib.core)
create:278, AbstractClassGenerator (org.springframework.cglib.core)
createHelper:433, Enhancer (org.springframework.cglib.proxy)
createClass:338, Enhancer (org.springframework.cglib.proxy)
createProxyClassAndInstance:55, ObjenesisCglibAopProxy (org.springframework.aop.framework)
getProxy:203, CglibAopProxy (org.springframework.aop.framework)
getProxy:109, ProxyFactory (org.springframework.aop.framework)
createProxy:468, AbstractAutoProxyCreator (org.springframework.aop.framework.autoproxy)
wrapIfNecessary:349, AbstractAutoProxyCreator (org.springframework.aop.framework.autoproxy)
```

们proxyTargetClass为false，使 jdk动 代 ， 代 对 口 ，代 对 也 口 实 对 ，it instanceof SomeComponent 口为ture，



```
p it = {$Proxy47@15732}
  h = {JdkDynamicAopProxy@15735}
```

口对 ，将会 InvocationHandler invoke ，也就 org.springframework.aop.framework.JdkDynamicAopProxy#invoke

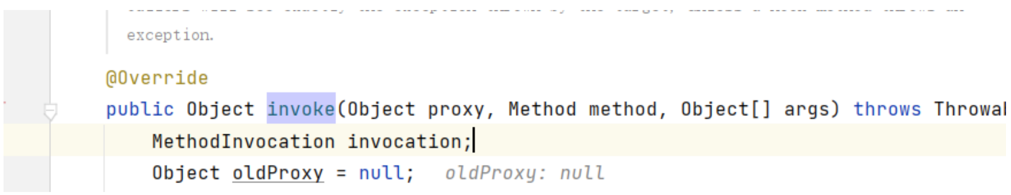


```
g-aop-4.2.6.RELEASE.jar > org > springframework > aop > framework > JdkDynamicAopProxy > invoke

if (this.advised.exposeProxy) {
    // Make invocation available if necessary.
    oldProxy = AopContext.setCurrentProxy(proxy);   proxy: $Proxy47@15732    oldProxy: null
    setProxyContext = true;   setProxyContext: false
}

// May be null. Get as late as possible to minimize the time we "own" the target,
// in case it comes from a pool.
target = targetSource.getTarget();   targetSource: "SingletonTargetSource for target object [example.ComponentImpl@6cdee57]"
if (target != null) {
    targetClass = target.getClass();   target: ComponentImpl@14472
}

// Get the interception chain for this method.
List<Object> chain = this.advised.getInterceptorsAndDynamicInterceptionAdvice(met

// Check whether we have any advice. If we don't, we can fallback on direct
```

```
Frames
"main"@1 in group "main": RUNNING
invoke:193, JdkDynamicAopProxy (org.springframework.aop.framework)
initialize:-1, $Proxy47 (com.sun.proxy)
lambda$new$0:31, Invoker (example)

Variables
this = {JdkDynamicAopProxy@15735}
proxy = {$Proxy47@15732}
method = {Method@15733} "public abst
args = null
oldProxy = null

method = {Method@15733} "public abstract void example.SomeComponent.initialize()"
  clazz = {Class@14468} "interface example.SomeComponent" ... Navigate
  slot = 0
  name = "initialize"
  returnType = {Class@15736} "void" ... Navigate
  parameterTypes = {Class[0]@15763}
  exceptionTypes = {Class[0]@15764}
  modifiers = 1073742849
  signature = null
  genericInfo = null
  annotations = {byte[6]@15765} [0, 1, 0, 6, 0, 0]
  parameterAnnotations = null
```

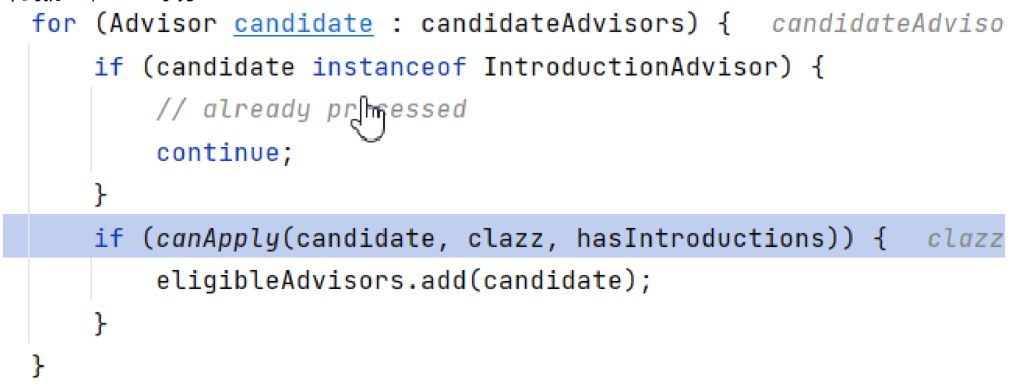
]

invoke 受 参 Method 名 class 口， abstract，

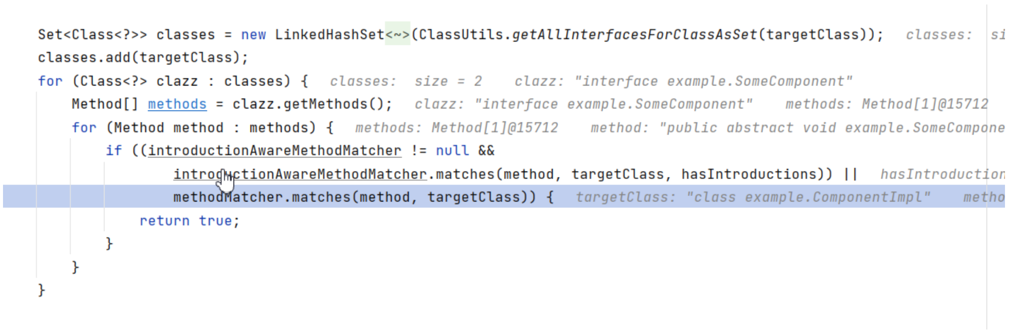


```
    exception.

@Override
public Object invoke(Object proxy, Method method, Object[] args) throws Throwal
    MethodInvocation invocation;
    Object oldProxy = null;  oldProxy: null
```

在invoke 中 了getInterceptorsAndDynamicInterceptionAdvice， 个 名 们 可以 TransactionalAttribute，因为在创 bean 候会 wrapIfNecessary， 个 内会 历 个Advisor，对 个Advisor 历对 口中 个 ， 后 Advisor中 Pointcut判 Advisor 否匹 个 ，如 匹 么就取出Advisor中 Advice，对于事务Advisor ，其pointcut在 个 上寻 TransactionAttribute，并将 和 到 TransactionAttribute 存 ，如 不到就 味 事务

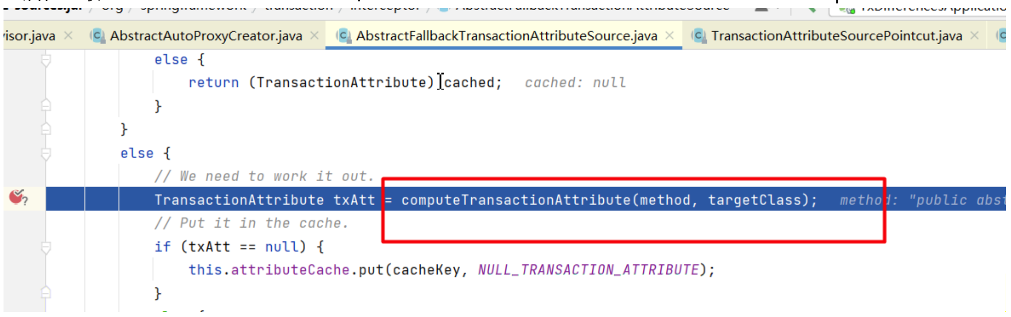


```
for (Advisor candidate : candidateAdvisors) {  candidateAdviso
    if (candidate instanceof IntroductionAdvisor) {
        // already processed
        continue;
    }
    if (canApply(candidate, clazz, hasIntroductions)) {  clazz
        eligibleAdvisors.add(candidate);
    }
}
```



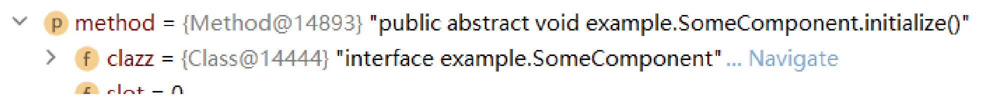
```
Set<Class<?>> classes = new LinkedHashSet<~>(ClassUtils.getAllInterfacesForClassAsSet(targetClass));  classes:  si
classes.add(targetClass);
for (Class<?> clazz : classes) {  classes:  size = 2   clazz: "interface example.SomeComponent"
    Method[] methods = clazz.getMethods();  clazz: "interface example.SomeComponent"   methods: Method[1]@15712
    for (Method method : methods) {  methods: Method[1]@15712   method: "public abstract void example.SomeCompone
        if ((introductionAwareMethodMatcher != null &&
                introductionAwareMethodMatcher.matches(method, targetClass, hasIntroductions)) ||  hasIntroductior
                methodMatcher.matches(method, targetClass)) {  targetClass: "class example.ComponentImpl"   metho
            return true;
        }
    }
}
```

后在 取TransactionAttribute 中AbstractFallbackTransactionAttributeSource computeTransactionAttribute



```
isor.java ×   AbstractAutoProxyCreator.java ×   AbstractFallbackTransactionAttributeSource.java ×   TransactionAttributeSourcePointcut.java ×

            else {
                return (TransactionAttribute) cached;  cached: null
            }
        }
        else {
            // We need to work it out.
            TransactionAttribute txAtt = computeTransactionAttribute(method, targetClass);  method: "public abst
            // Put it in the cache.
            if (txAtt == null) {
                this.attributeCache.put(cacheKey, NULL_TRANSACTION_ATTRIBUTE);
            }
```

如 传入 Method 名 口中 (class为 口 ，methodname为 abstract method)



```
˅ p method = {Method@14893} "public abstract void example.SomeComponent.initialize()"
› f clazz = {Class@14444} "interface example.SomeComponent" ... Navigate
```

computeTransactionAttribute 中 先 取targetClass， 个targetClass其实就 Bean对 class，也就 实

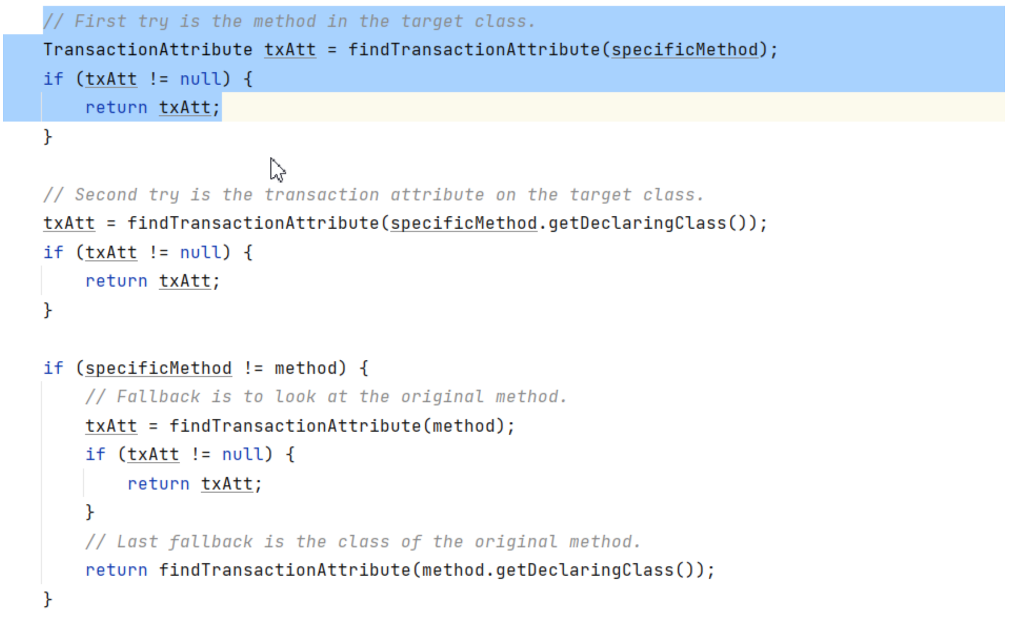
```
// Ignore CGLIB subclasses - introspect the actual user class.
Class<?> userClass = ClassUtils.getUserClass(targetClass);
```

后 method 取在实 上 实 specificMethod

```
// The method may be on an interface,
but we need attributes from the target class.
// If the target class is null, the method will be unchanged.
Method specificMethod = ClassUtils.
```

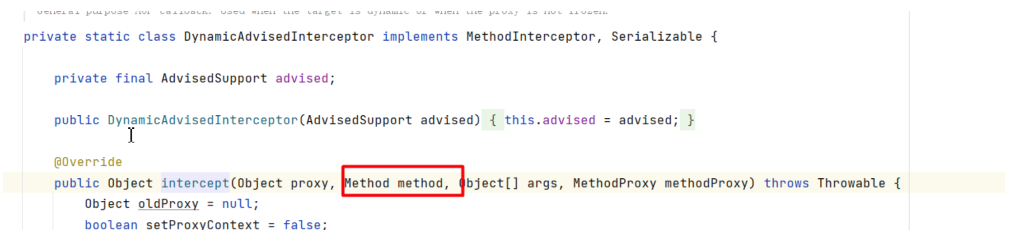
```
getMostSpecificMethod(method, userClass);
```

后在实 上 取 定 TransactionAttribute，如 ，就在实 上 取TransactionAttribute，如 ，判 前 到 method和SpecificMethod 否 同，如 不同则 取 前 到 method上 TransactionAttribute 在 口 上 @Transactional 且使 jdk代 候， 前 method 名信 class 口，method 口 abstract ，因为jdk代 对 实 了 口；



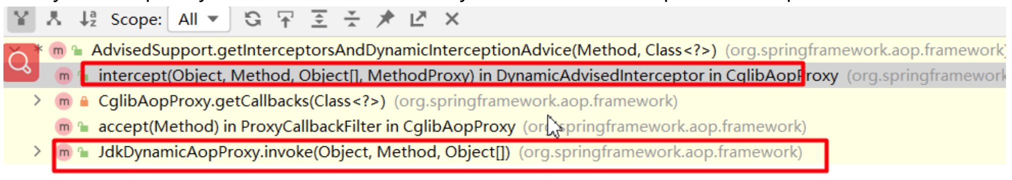
```
// First try is the method in the target class.
TransactionAttribute txAtt = findTransactionAttribute(specificMethod);
if (txAtt != null) {
    return txAtt;
}

// Second try is the transaction attribute on the target class.
txAtt = findTransactionAttribute(specificMethod.getDeclaringClass());
if (txAtt != null) {
    return txAtt;
}

if (specificMethod != method) {
    // Fallback is to look at the original method.
    txAtt = findTransactionAttribute(method);
    if (txAtt != null) {
        return txAtt;
    }
    // Last fallback is the class of the original method.
    return findTransactionAttribute(method.getDeclaringClass());
}
```

在 口 上 且proxyTargetClass为true使 cglib代 ， 会 DynamicAdvisedInterceptor 到，其中method 名 class 为实 对 名， 为ComponentImpl public void initialize()



```
General purpose AOP callback. Used when the target is dynamic or when the proxy is not frozen.
private static class DynamicAdvisedInterceptor implements MethodInterceptor, Serializable {

    private final AdvisedSupport advised;

    public DynamicAdvisedInterceptor(AdvisedSupport advised) { this.advised = advised; }

    @Override
    public Object intercept(Object proxy, Method method, Object[] args, MethodProxy methodProxy) throws Throwable {
        Object oldProxy = null;
        boolean setProxyContext = false;
```

不 cglibg动 代 jdk动 代 ，对于代 对 其 候会 InvocationHandler invoke ，也就 JDKDynamicAopProxy invoke ， DynamicAdvisedInterceptor Interceptor



Scope: All

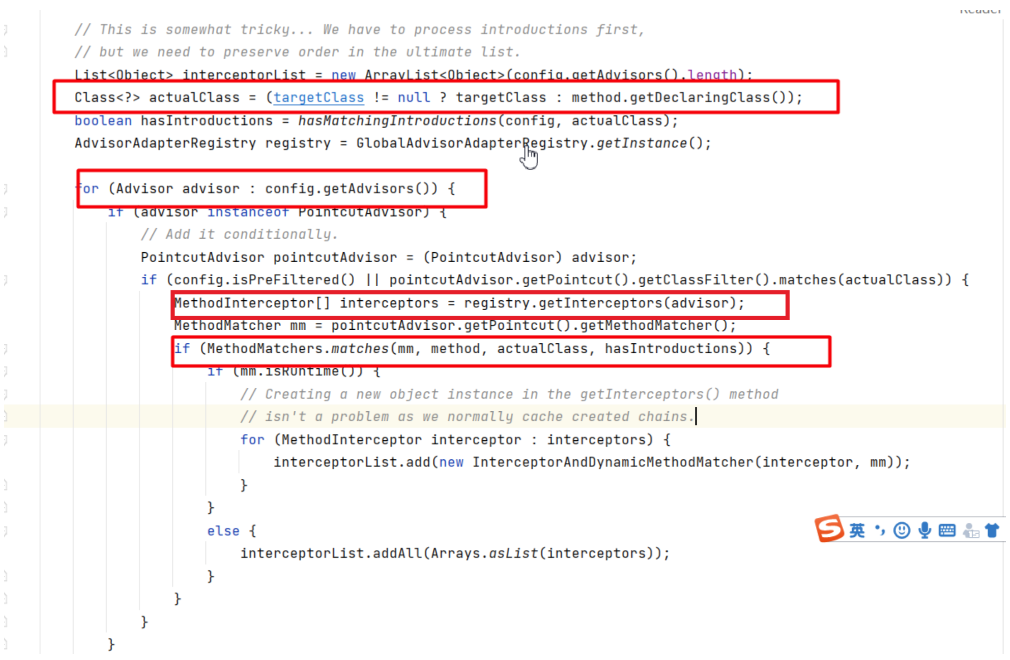
```
AdvisedSupport.getInterceptorsAndDynamicInterceptionAdvice(Method, Class<?>) (org.springframework.aop.framework)
  intercept(Object, Method, Object[], MethodProxy) in DynamicAdvisedInterceptor in CglibAopProxy (org.springframework
  CglibAopProxy.getCallbacks(Class<?>) (org.springframework.aop.framework)
  accept(Method) in ProxyCallbackFilter in CglibAopProxy (org.springframework.aop.framework)
  JdkDynamicAopProxy.invoke(Object, Method, Object[]) (org.springframework.aop.framework)
```

在 两个 中 会 AdvisedSupport getInterceptorsAndDynamicInterceptionAdvice 个

// Get the interception chain for this method.

List<Object> chain = this.advised.getInterceptorsAndDynamicInterceptionAdvice(method, targetClass);

在 org.springframework.aop.framework.DefaultAdvisorChainFactory#getInterceptorsAndDynamicInterceptionAdvice 中， 历 个Advisor， Advisor中 Pointcut 去匹 个

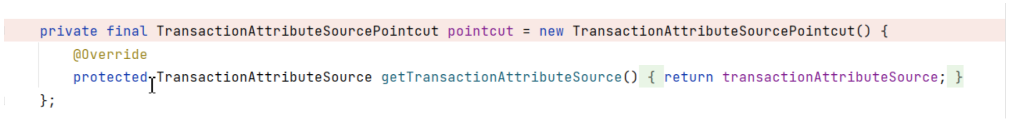


```
    // This is somewhat tricky... We have to process introductions first,
    // but we need to preserve order in the ultimate list.
    List<Object> interceptorList = new ArrayList<Object>(config.getAdvisors().length);
    Class<?> actualClass = (targetClass != null ? targetClass : method.getDeclaringClass());
    boolean hasIntroductions = hasMatchingIntroductions(config, actualClass);
    AdvisorAdapterRegistry registry = GlobalAdvisorAdapterRegistry.getInstance();

    for (Advisor advisor : config.getAdvisors()) {
        if (advisor instanceof PointcutAdvisor) {
            // Add it conditionally.
            PointcutAdvisor pointcutAdvisor = (PointcutAdvisor) advisor;
            if (config.isPreFiltered() || pointcutAdvisor.getPointcut().getClassFilter().matches(actualClass)) {
                MethodInterceptor[] interceptors = registry.getInterceptors(advisor);
                MethodMatcher mm = pointcutAdvisor.getPointcut().getMethodMatcher();
                if (MethodMatchers.matches(mm, method, actualClass, hasIntroductions)) {
                    if (mm.isRuntime()) {
                        // Creating a new object instance in the getInterceptors() method
                        // isn't a problem as we normally cache created chains.
                        for (MethodInterceptor interceptor : interceptors) {
                            interceptorList.add(new InterceptorAndDynamicMethodMatcher(interceptor, mm));
                        }
                    }
                    else {
                        interceptorList.addAll(Arrays.asList(interceptors));
                    }
                }
            }
        }
```

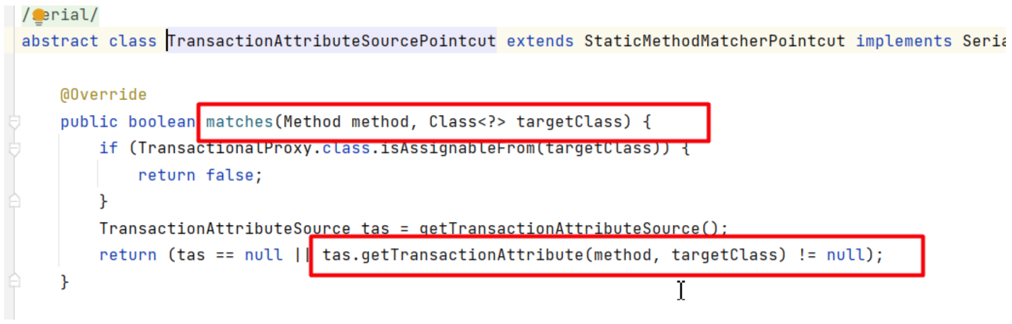
 如BeanFactoryTransactionAttributeSourceAdvisor 个advisor，

MethodInterceptor[] interceptors = registry.getInterceptors(advisor);将会 回MethodInterceptor

其Pointcut实 match   会判   上 否存在TransactionAttribute属



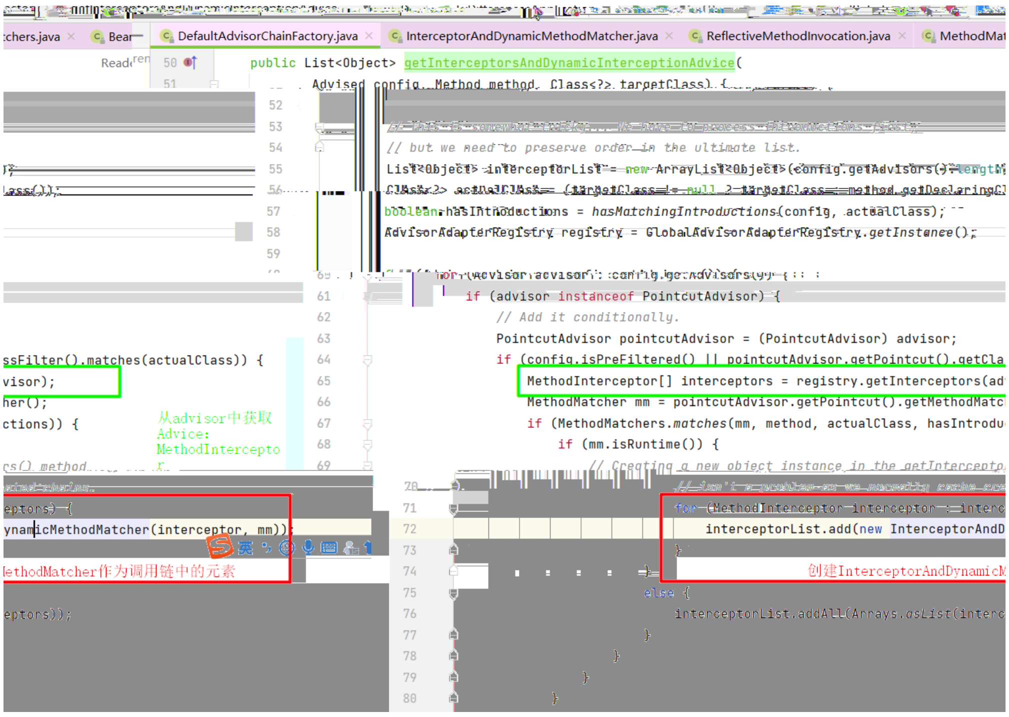
```
    private final TransactionAttributeSourcePointcut pointcut = new TransactionAttributeSourcePointcut() {
        @Override
        protected TransactionAttributeSource getTransactionAttributeSource() { return transactionAttributeSource; }
    };
```



```
/erial/
abstract class TransactionAttributeSourcePointcut extends StaticMethodMatcherPointcut implements Seria

    @Override
    public boolean matches(Method method, Class<?> targetClass) {
        if (TransactionalProxy.class.isAssignableFrom(targetClass)) {
            return false;
        }
        TransactionAttributeSource tas = getTransactionAttributeSource();
        return (tas == null || tas.getTransactionAttribute(method, targetClass) != null);
    }
```

如   method对 ， 出 个method上存在 @Transactional，则 个   个事务   否则就J 事务 ，对于事务

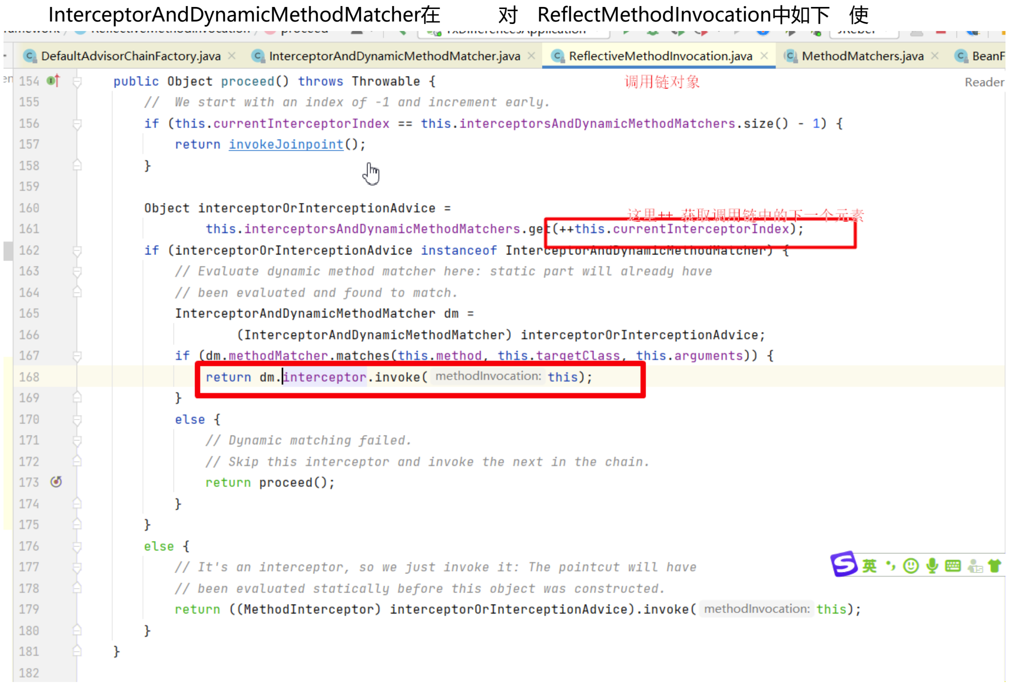
，会 回MethodInterceptor作为 器



```java
public List<Object> getInterceptorsAndDynamicInterceptionAdvice(
        Advised config, Method method, Class<?> targetClass) {

    // but we need to preserve order in the ultimate list.
    List<Object> interceptorList = new ArrayList<Object>(config.getAdvisors().length
    Class<?> actualClass = (targetClass != null ? targetClass : method.getDeclaringCl
    boolean hasIntroductions = hasMatchingIntroductions(config, actualClass);
    AdvisorAdapterRegistry registry = GlobalAdvisorAdapterRegistry.getInstance();

    for (Advisor advisor : config.getAdvisors()) {
        if (advisor instanceof PointcutAdvisor) {
            // Add it conditionally.
            PointcutAdvisor pointcutAdvisor = (PointcutAdvisor) advisor;
            if (config.isPreFiltered() || pointcutAdvisor.getPointcut().getCla
                MethodInterceptor[] interceptors = registry.getInterceptors(ad
                MethodMatcher mm = pointcutAdvisor.getPointcut().getMethodMatc
                if (MethodMatchers.matches(mm, method, actualClass, hasIntrodu
                    if (mm.isRuntime()) {
                        // Creating a new object instance in the getIntercepto
                        // isn't a problem as we normally cache created chains.
                        for (MethodInterceptor interceptor : interc
                            interceptorList.add(new InterceptorAndD
                        }
                    }
                    else {
                        interceptorList.addAll(Arrays.asList(interc
                    }
                }
            }
```

从advisor中获取Advice: MethodInterceptor
ethodMatcher作为调用链中的元素
创建InterceptorAndDynamicM

InterceptorAndDynamicMethodMatcher在　　　对　ReflectMethodInvocation中如下　使

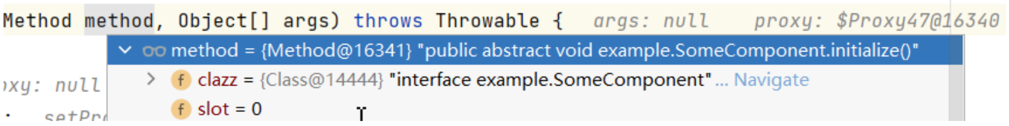


```java
public Object proceed() throws Throwable {
    //  We start with an index of -1 and increment early.
    if (this.currentInterceptorIndex == this.interceptorsAndDynamicMethodMatchers.size() - 1) {
        return invokeJoinpoint();
    }

    Object interceptorOrInterceptionAdvice =
            this.interceptorsAndDynamicMethodMatchers.get(++this.currentInterceptorIndex);
    if (interceptorOrInterceptionAdvice instanceof InterceptorAndDynamicMethodMatcher) {
        // Evaluate dynamic method matcher here: static part will already have
        // been evaluated and found to match.
        InterceptorAndDynamicMethodMatcher dm =
                (InterceptorAndDynamicMethodMatcher) interceptorOrInterceptionAdvice;
        if (dm.methodMatcher.matches(this.method, this.targetClass, this.arguments)) {
            return dm.interceptor.invoke( methodInvocation: this);
        }
        else {
            // Dynamic matching failed.
            // Skip this interceptor and invoke the next in the chain.
            return proceed();
        }
    }
    else {
        // It's an interceptor, so we just invoke it: The pointcut will have
        // been evaluated statically before this object was constructed.
        return ((MethodInterceptor) interceptorOrInterceptionAdvice).invoke( methodInvocation: this);
    }
}
```

]

么　们　　另外　个　　：

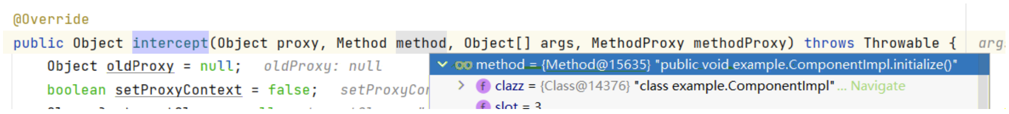
（1）jdk动　代　　org.springframework.aop.framework.JdkDynamicAopProxy#invoke　　参　method　　　到　　，　个　　　名信　，class为　口，method为　口中　　　　个发



```
Method method, Object[] args) throws Throwable {   args: null    proxy: $Proxy47@16340
    method = {Method@16341} "public abstract void example.SomeComponent.initialize()"
        clazz = {Class@14444} "interface example.SomeComponent" ... Navigate
        slot = 0
```

]

（2）cglib代　org.springframework.aop.framework.CglibAopProxy.DynamicAdvisedInterceptor#intercept　　　参　method　　　到　，　个method　信　如下

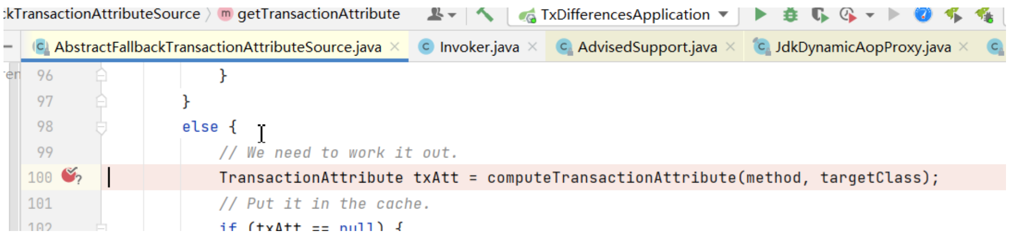


```java
@Override
public Object intercept(Object proxy, Method method, Object[] args, MethodProxy methodProxy) throws Throwable {
    Object oldProxy = null;   oldProxy: null
    boolean setProxyContext = false;   setProxyCo
```

```
method = {Method@15635} "public void example.ComponentImpl.initialize()"
    clazz = {Class@14376} "class example.ComponentImpl" ... Navigate
    slot = 3
```

另外　个　，　　　到　method　定　　上　否存在TransactionAttribute，从　判　　　否　事务　，　个　　在哪　实　？

org.springframework.transaction.interceptor.AbstractFallbackTransactionAttributeSource#getTransactionAttribute 中会 computeTransactionAttribute org.springframework.transaction.interceptor.AbstractFallbackTransactionAttributeSource#computeTransactionAttribute



```
			}
		}
		else {
			// We need to work it out.
			TransactionAttribute txAtt = computeTransactionAttribute(method, targetClass);
			// Put it in the cache.
			if (txAtt == null) {
```