**3.3.2  Cglib2AopProxy的intercept拦截**

在分析Cglib2AopProxy的AopProxy代理对象生成的时候，我们知道对于AOP的拦截调用，其回调是在DynamicAdvisedInterceptor对象中实现的，这个回调的实现在intercept方法中，如代码清单3-19所示。Cglib2AopProxy的intercept回调方法的实现和JdkDynamicAopProxy的回调实现是非常类似的，只是在Cglib2AopProxy中构造的是CglibMethodInvocation对象来完成拦截器链的调用，而在JdkDynamicAopProxy中是通过构造ReflectiveMethodInvocation对象来完成这个功能的。

代码清单3-19  DynamicAdvisedInterceptor的intercept

1. **public** Object intercept(Object proxy, Method method,
2. Object[] args, MethodProxy
3. methodProxy) **throws** Throwable {
4. Object oldProxy = **null**;
5. **boolean** setProxyContext = **false**;
6. Class targetClass = **null**;
7. Object target = **null**;
8. **try** {
9. **if** (**this**.advised.exposeProxy) {
10. // Make invocation available if
11. necessary.
12. oldProxy =
13. AopContext.setCurrentProxy(proxy);
14. setProxyContext = **true**;
15. }
16. /\*\*
17. \* May be <code>null</code>. Get as late as
18. possible to minimize the time we
19. \* "own" the target, in case it comes from a
20. pool.
21. \*/
22. target = getTarget();
23. **if** (target != **null**) {
24. targetClass = target.getClass();
25. }
26. //从advised中取得配置好的AOP通知。
27. List<Object> chain =
28. **this**.advised.getInterceptorsAndDynamicInterceptionAdvice
29. (method, targetClass);
30. Object retVal = **null**;
31. /\*\*
32. \* Check whether we only have one
33. InvokerInterceptor: that is,
34. \* no real advice, but just reflective
35. invocation of the target.
36. \*/
37. //
38. 如果没有AOP通知配置，那么直接调用target对象的调用方法。
39. **if** (chain.isEmpty() &&
40. Modifier.isPublic(method.getModifiers())) {
41. /\*\*
42. \* We can skip creating a
43. MethodInvocation: just invoke the target directly.
44. \* Note that the final invoker must
45. be an InvokerInterceptor, so we know
46. \* it does nothing but a reflective
47. operation on the target, and no hot
48. \* swapping or fancy proxying.
49. \*/
50. retVal = methodProxy.invoke(target,
51. args);
52. }
53. **else** {
55. //通过CglibMethodInvocation来启动advice通知。
56. retVal = **new**
57. CglibMethodInvocation(proxy, target, method, args,
58. targetClass, chain, methodProxy).proceed();
59. }
60. retVal = massageReturnTypeIfNecessary(proxy,
61. target, method, retVal);
62. **return** retVal;
63. }
64. **finally** {
65. **if** (target != **null**) {
66. releaseTarget(target);
67. }
68. **if** (setProxyContext) {
69. // Restore old proxy.
71. AopContext.setCurrentProxy(oldProxy);
72. }
73. }
74. }