

CGlib代码

代理要素

- 目标对象
- 代理类
 - 必须实现：MethodInterceptor
- 代理对象

第一步：引入cglib的依赖

```
1 <dependency>
2     <groupId>cglib</groupId>
3     <artifactId>cglib</artifactId>
4     <version>3.3.0</version>
5 </dependency>
```

第二步：创建代理和代理对象

```
1 package com.kuangstudy.second.cglib;
2
3 import net.sf.cglib.proxy.Enhancer;
4 import net.sf.cglib.proxy.MethodInterceptor;
5 import net.sf.cglib.proxy.MethodProxy;
6
7 import java.lang.reflect.Method;
```

```
8
9 //Cglib动态代理，实现MethodInterceptor接口
10 public class CglibProxy implements MethodInterceptor {
11
12     private Object target;//需要代理的目标对象
13
14     //重写拦截方法
15     @Override
16     public Object intercept(Object obj, Method method,
17 object[] arr, MethodProxy proxy) throws Throwable {
18         System.out.println("Cglib动态代理，监听开始！");
19         Object invoke = method.invoke(target, arr);//方法执
20         行，参数：target 目标对象 arr参数数组
21         System.out.println("Cglib动态代理，监听结束！");
22         return invoke;
23     }
24
25     //定义获取代理对象方法
26     public Object getCglibProxy(Object objectTarget) {
27         //为目标对象target赋值
28         this.target = objectTarget;
29
30         // 创建代理对象
31         Enhancer enhancer = new Enhancer();
32         //设置父类,因为Cglib是针对指定的类生成一个子类，所以需要指定父
33         类
34         enhancer.setSuperclass(objectTarget.getClass());
35         enhancer.setCallback(this);// 设置回调
36         Object result = enhancer.create();//创建并返回代理对象
37         return result;
38     }
39 }
```

第三步：测试

```
1 package com.kuangstudy.second.cglib;
2
3 import com.kuangstudy.second.jdkproxy.UserServiceImpl;
4
5 /**
6  * @author 飞哥
7  * @Title: 学相伴出品
8  * @Description: 飞哥B站地址:
9  * https://space.bilibili.com/490711252
10  * 记得关注和三连哦!
11  * @Description: 我们有一个学习网站: https://www.kuangstudy.com
12  * @date 2021/12/22 21:15
13  */
14 public class MainTest {
15     public static void main(String[] args) {
16         CglibProxy cglib = new CglibProxy();//实例化CglibProxy
17         //对象
18         UserServiceImpl user = (UserServiceImpl)
19         cglib.getCglibProxy(new UserServiceImpl());//获取代理对象
20         user.delUser(1L);//执行删除方法
21     }
22 }
```

总结：

- 和jdk动态代理思想几乎是一模一样。只不过代理类实现的接口是：MethodInterceptor 覆盖的方法是:intercept方法
- 根据目标对象创建代理对象

```

1 //定义获取代理对象方法
2     public Object getCglibProxy(Object objectTarget) {
3         //为目标对象target赋值
4         this.target = objectTarget;
5
6         // 创建代理对象
7         Enhancer enhancer = new Enhancer();
8         //设置父类,因为Cglib是针对指定的类生成一个子类,所以需要指
        定父类
9         enhancer.setSuperclass(objectTarget.getClass());
10        enhancer.setCallback(this); // 设置回调
11        Object result = enhancer.create(); //创建并返回代理
        对象
12        return result;
13    }

```

- 代理对象执行方法

```

1 package com.kuangstudy.second.cglib;
2
3 import com.kuangstudy.second.jdkproxy.UserServiceImpl;
4
5 /**
6  * @author 飞哥
7  * @Title: 学相伴出品
8  * @Description: 飞哥B站地址:
9  * https://space.bilibili.com/490711252
10  * 记得关注和三连哦!
11  * @Description: 我们有一个学习网站:
12  * https://www.kuangstudy.com
13  * @date 2021/12/22 21:15
14  */
15 public class MainTest {
16
17     public static void main(String[] args) {
18         CglibProxy cglib = new CglibProxy(); //实例化
19         CglibProxy对象
20     }
21 }

```

```
17         UserServiceImpl user = (UserServiceImpl)
    cglib.getCglibProxy(new UserServiceImpl()); // 获取代理对象
18         user.delUser(1L); // 执行删除方法
19     }
20 }
21
```