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

```
<dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib</artifactId>
    <version>3.3.0</version>
</dependency>
```

第二步：创建代理和代理对象

```
package com.kuangstudy.second.cglib;

import net.sf.cglib.proxy.Enhancer;
import net.sf.cglib.proxy.MethodInterceptor;
import net.sf.cglib.proxy.MethodProxy;

import java.lang.reflect.Method;
```

```java

//Cglib动态代理，实现MethodInterceptor接口
public class CglibProxy implements MethodInterceptor {

    private Object target;//需要代理的目标对象

    //重写拦截方法
    @Override
    public Object intercept(Object obj, Method method,
Object[] arr, MethodProxy proxy) throws Throwable {
        System.out.println("Cglib动态代理，监听开始！");
        Object invoke = method.invoke(target, arr);//方法执
行，参数：target 目标对象 arr参数数组
        System.out.println("Cglib动态代理，监听结束！");
        return invoke;
    }

    //定义获取代理对象方法
    public Object getCglibProxy(Object objectTarget) {
        //为目标对象target赋值
        this.target = objectTarget;

        // 创建代理对象
        Enhancer enhancer = new Enhancer();
        //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
        enhancer.setSuperclass(objectTarget.getClass());
        enhancer.setCallback(this);// 设置回调
        Object result = enhancer.create();//创建并返回代理对象
        return result;
    }
}
```

第三步：测试

```
package com.kuangstudy.second.cglib;

import com.kuangstudy.second.jdkproxy.UserServiceImpl;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/12/22 21:15
 */
public class MainTest {

    public static void main(String[] args) {
        CglibProxy cglib = new CglibProxy();//实例化CglibProxy对象
        UserServiceImpl user = (UserServiceImpl) cglib.getCglibProxy(new UserServiceImpl());//获取代理对象
        user.delUser(1L);//执行删除方法
    }
}

```

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

```
 //定义获取代理对象方法
    public Object getCglibProxy(Object objectTarget) {
        //为目标对象target赋值
        this.target = objectTarget;

        // 创建代理对象
        Enhancer enhancer = new Enhancer();
        //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
        enhancer.setSuperclass(objectTarget.getClass());
        enhancer.setCallback(this);// 设置回调
        Object result = enhancer.create();//创建并返回代理对象
        return result;
    }
```

- 代理对象执行方法

```
package com.kuangstudy.second.cglib;

import com.kuangstudy.second.jdkproxy.UserServiceImpl;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/12/22 21:15
 */
public class MainTest {

    public static void main(String[] args) {
        CglibProxy cglib = new CglibProxy();//实例化CglibProxy对象
```

```
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
```