



链滴

# java 代理的两种实现

作者: [tianyunperfect](#)

原文链接: <https://ld246.com/article/1552872855369>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



java的代理有两种实现方式

## java原生代理接口的实现类

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

```
/**  
 * 自定义代理增强  
 * T 为接口  
 * t 为接口的实现类  
 */  
public class proxyjdk {  
    public <T> T create(final T t) {  
        final Class<?> aClass = t.getClass();  
        Object o = Proxy.newProxyInstance(aClass.getClassLoader(),  
            aClass.getInterfaces(),  
            new InvocationHandler() {  
                public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {  
                    if ("method1".equalsIgnoreCase(method.getName())) {  
                        System.out.println("加强开始");  
                        Object invoke = method.invoke(t, args);  
                        System.out.println("加强结束");  
                        return invoke;  
                    }  
                    return method.invoke(t, args);  
                }  
            });  
        return (T) o;  
    }  
}
```

```
}  
}
```

## spring框架proxycglib代理一个对象

```
import org.springframework.cglib.proxy.Enhancer;  
import org.springframework.cglib.proxy.MethodInterceptor;  
import org.springframework.cglib.proxy.MethodProxy;  
  
import java.lang.reflect.Method;  
  
/**  
 * cglib代理增强（直接传入普通对象即可）  
 * t为普通对象  
 */  
public class proxycglib {  
    public <T> T create(final T t) {  
        Object o = Enhancer.create(t.getClass(), new MethodInterceptor() {  
            public Object intercept(Object o, Method method, Object[] objects, MethodProxy methodProxy) throws Throwable {  
                if ("method1".equalsIgnoreCase(method.getName())) {  
                    System.out.println("方法开始增强");  
                    Object invoke = method.invoke(t, objects);  
                    System.out.println("方法增强结束");  
                    return invoke;  
                }  
                return method.invoke(t, objects);  
            }  
        });  
        return (T) o;  
    }  
}
```