实验三， 动态代理

```
DynamicLoggerProxyDemo
        |
        |--------tarena
                    |
                    |------------spring-config.xml
```

步骤一，创建接口IHello类

```
public interface IHello {
        // 业务方法声明
        public void hello(String name);
}
```

步骤二，创建实现类HelloSpeaker

```
public class HelloSpeaker implements IHello {

        public void hello(String name) {
                System.out.println("你好，" + name);
        }

}
```

步骤三，创建日志处理器LogHandler类

```
//动态代理，使用一个日志代理器，服务于多个对象
//必须实现InvocationHandlerr接口
public class LogHandler implements InvocationHandler {
```

```
        private Logger logger =
Logger.getLogger(this.getClass().getName());
        private Object delegate;

        // 绑定被代理对象，动态产生代理对象
        public Object bind(Object delegate) {
                this.delegate = delegate;
                return (
                                Proxy.newProxyInstance(

delegate.getClass().getClassLoader(),

delegate.getClass().getInterfaces(),
                                                this)
                );
        }

        // 回调方法
        public Object invoke(
                        Object proxy,
                        Method method,
                        Object[] args)
                        throws Throwable {

                Object result = null;

                logger.info("日志开始");
                result = method.invoke(delegate,args);
                logger.info("日志结束");

                return result;
        }

}
```

步骤四，创建测试类

```
public class Test {
```

```
        public static void main(String[] args) {

                //创建日志处理器
                LogHandler logHandler = new LogHandler();

                //绑定被代理对象，动态产生代理对象
                IHello proxy =
                        (IHello)logHandler.bind(new
HelloSpeaker());

                //调用业务方法
                proxy.hello("赵君");
        }

}
```

步骤五，结果
信息: 日志开始
你好，赵君
信息: 日志结束