01_反射、注解、设计模式综合案例

```
public interface UserDao {

    @SysLog(className = "com.qzw.dao.UserDaoImpl" , methodName = "addUser")
    void addUser() throws NoSuchMethodException;

    void deleteUser() throws NoSuchMethodException;

    void updateUser() throws NoSuchMethodException;
```

- }

- 如上图所示，需求如下：
  - 加了@SysLog注解的addUser方法在执行后，进行日志记录;
  - 其他没加@SysLog注解的方法，不进行日志记录;
  - 日志内容为"2020年04月15日 08:26:12 --- com.qzw.dao.UserDaoImpl类中 --- addUser()方法 --- 运行了"。
- 代码实现一

```
public class UserDaoWrapper implements UserDao {

    private UserDao userDao;

    public UserDaoWrapper(UserDao userDao) {
        this.userDao = userDao;
    }

    @Override
    public void addUser() throws NoSuchMethodException {
        userDao.addUser();
        printLog("addUser");
    }

    @Override
    public void deleteUser() throws NoSuchMethodException {
        userDao.deleteUser();
        printLog("deleteUser");
    }

    @Override
    public void updateUser() throws NoSuchMethodException {
        userDao.updateUser();
        printLog("updateUser");
    }

    private void printLog(String runMethodName) throws NoSuchMethodException {
        Class<? extends UserDao> clazz = userDao.getClass();
        Class<?>[] interfaces = clazz.getInterfaces();
```

```
        Method method = interfaces[0].getMethod(runMethodName);
        if (null != method) {
            boolean present = method.isAnnotationPresent(SysLog.class);
            if (present) {
                SysLog sysLog = method.getAnnotation(SysLog.class);
                String className = sysLog.className();
                String methodName = sysLog.methodName();
                SimpleDateFormat format = new SimpleDateFormat("yyyy年MM月dd日
hh:mm:ss");
                String currentTimeStr = format.format(new Date());
                System.out.println(currentTimeStr + " --- " + className + "类中 --- "
+ methodName + "方法 --- 运行了");
            }
        }
    }
}
```

- 以上代码，存在问题：printLog方法需要每个方法中调用，这是因为装饰者设计模式需要重写所有方法导致的！

- 代码实现二

```
UserDao userDao = new UserDaoImpl();
UserDao userDaoProxy = (UserDao) Proxy.newProxyInstance(
        userDao.getClass().getClassLoader(),
        userDao.getClass().getInterfaces(),
        new InvocationHandler() {
            @Override
            public Object invoke(Object proxy, Method method, Object[] args) throws
Throwable {
                System.out.println(method.getName());
                boolean present = method.isAnnotationPresent(SysLog.class);
                System.out.println("present : "+present);
                Object returnValue = null;
                if (present) {
                    returnValue = method.invoke(userDao,args);
                    SysLog sysLog = method.getAnnotation(SysLog.class);
                    String className = sysLog.className();
                    String methodName = sysLog.methodName();
                    SimpleDateFormat format = new SimpleDateFormat("yyyy年MM月dd日
hh:mm:ss");
                    String currentTimeStr = format.format(new Date());
                    System.out.println(currentTimeStr + " --- " + className + "类中 -
-- " + methodName + "方法 --- 运行了");
                } else {
                    returnValue = method.invoke(userDao,args);
                }
                return returnValue;
            }
        });
userDaoProxy.addUser();
```