# 说明

本模块的目的是为了学习框架技术进行知识储备，回顾一路走来遇到过的基础知识，对基础知识点进行罗列和探索，了解每种技术能完成的工作和极限，并且结合在日常开发过程中碰到的疑问进行探讨解答。

# 1.java反射技术

参考链接:

<https://blog.csdn.net/sinat_38259539/article/details/71799078>

反射技术在框架中无处不在，是框架技术一个非常重要的手段，要使用反射技术，先要取得类的Class。现在，我们先来看看Class类的jdk说明：



总的来说，Class类的jdk方法可以总结出来：

1. 获取Class的注释（annotation）；
2. 获取类加载器；
3. 获取构造器；
4. 获取声明的方法、属性和构造器；
5. 获取自身的信息，例如：名称、包裹路径；
6. 判断是否存在注释、是否是接口，是否是枚举，是共有还是私有，是本地类还是成员类，是否是数组等等；
7. 创建实例；
8. 获取类实现了那些接口；
9. 等等，例如：获取资源文件（详见jdk文档）。。。；

可以归分为几类：Class类、Interface类、Field类、Method类和Constructor类。因此，根据上面可知，如果获取到一个类的class，就拥有了以上操作的权限。

Java demo：

FrameDigger\reflectDigger\src\main\java\com\vi\demo\Demo1.java



从代码中 可以联想 spring的bean要求必须带有无参的构造函数，在本demo中就可以得知。

# 2.java代理

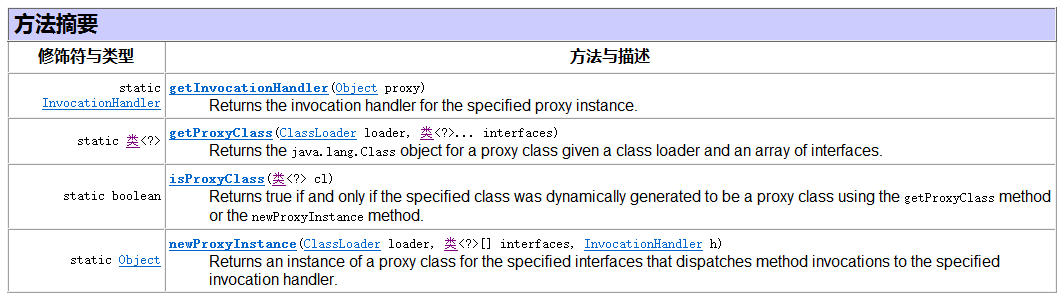
参考链接：

<https://www.cnblogs.com/flying607/p/3464268.html>

代理在spring中非常重要，是一个无处不在的东西。

* java.lang.reflect.Proxy类；
* java.lang.reflect.InvocationHandler接口；

上面的类和接口是代理中一个非常重要的内容，要了解代理，必须对该类和接口进行探讨。



**要注意的是，代理有两种，一种是jdk的代理，一种是cglib的代理，两种的区别是jdk代理类必须实现接口，而cglib则不需要，cglib是针对类来实现代理，在spring的aop中，两种代理都使用到。**

Spring Aop的代理应用说明如下：

* spring中实现AOP用的是JDK代理和CGLIB代理；
* JDK动态代理只针对实现了接口的类生成代理；
* CGlib代理针对类实现代理，主要是指定的类生成的一个子类，覆盖其中所有的方法，该类的方法不能声明为final；
* 如果目标没有实现接口，则会默认采用cglib代理；

下面通过demo来了解两种代理的具体实现。

## 2.1 jdk代理

Jdk代理有静态代理和动态代理两种。

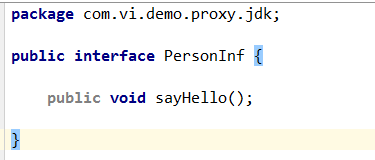
* **静态代理：**

Java Demo：

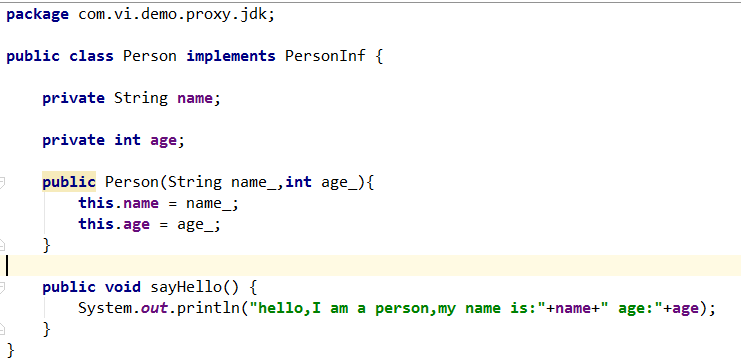
FrameDigger\reflectDigger\src\main\java\com\vi\demo\proxy\jdk\PersonProxyStatic.java

静态代理实际上是利用设计模式，把接口当做属性对象写在代理类中，然后在实现的时候，指定接口的真正实现类，在这个过程中并没有使用使用到proxy这些jdk的代理类，仅仅通过一些设计模式来实现。

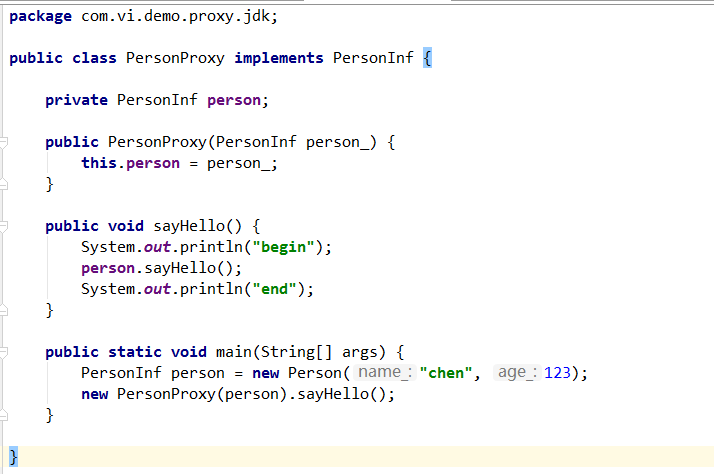
先定义一个PersonInf接口：



然后实现这个接口类Person：



最后编写代理类，并在代理类中使用PersonInf接口作为属性对象。Main方法中指定接口对象为真正的对象Person，实现代理。



整个过程实际上是使用了设计模式，并没有使用到jdk的Proxy等代理类。

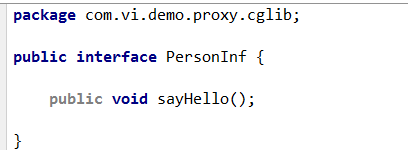
* 动态代理：

Java demo：

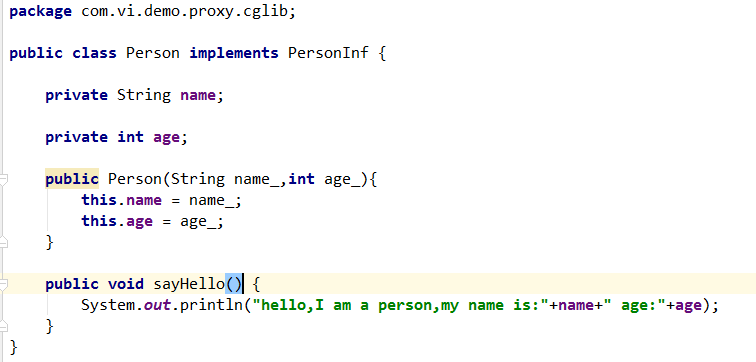
FrameDigger\reflectDigger\src\main\java\com\vi\demo\proxy\jdk\PersonProxyActive.java

首先同样需要建立相同的PersonInf接口和接口实现类Person：

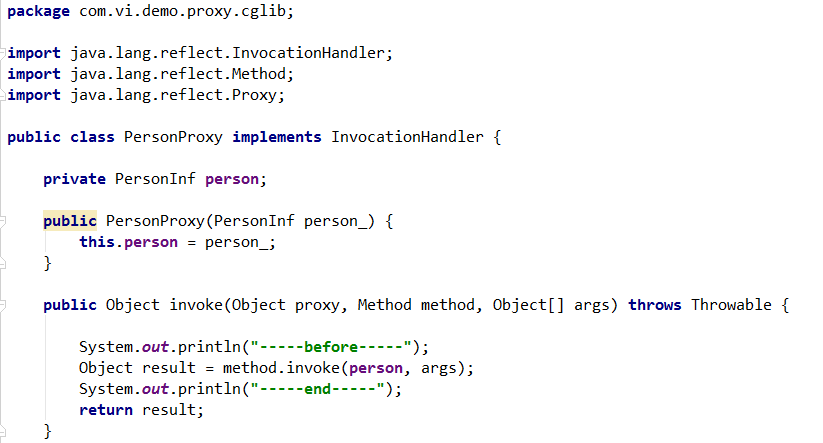
PersonInf：

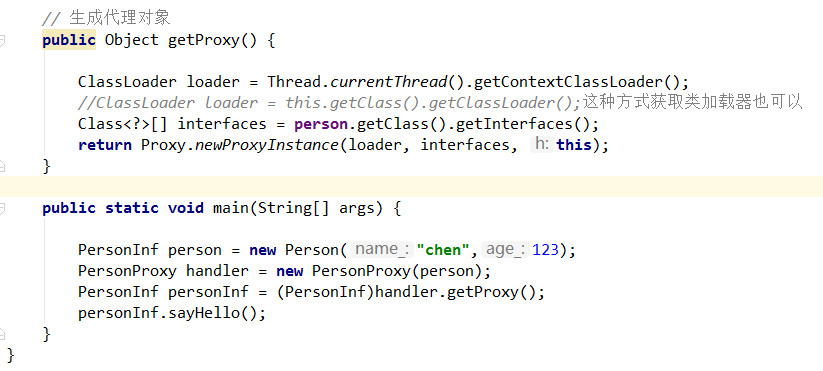


Person：



接着实现代理类，其中代理类必须需要实现InvocationHandler接口：





实现的时候，需要指定被代理类，并且被代理类需要通过

Proxy.*newProxyInstance*(loader, interfaces, **this**) 的方式进行加载，该方法有三个参数

* 第一个参数：类加载器，可以获取当前线程使用的类加载器，也可以获取当前类的类加载器；
* 第二个参数：被代理类实现了的接口；
* 第三个参数：代理类；

## 2.2 cglidb代理

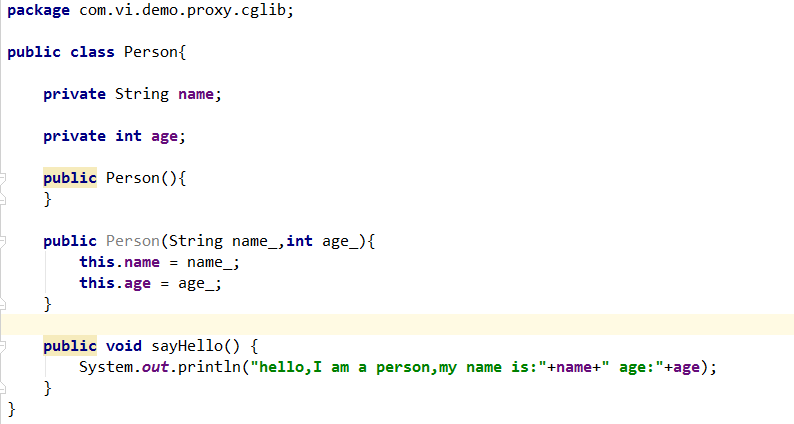
Cglib动态代理跟jdk不一样的地方：

* Cglib被代理类可以不实现接口；
* Cglib的代理类必须要有无参的构造函数；
* Jdk的代理类需要实现接口InvocationHandler，cglib的代理类需要实现接口MethodInterceptor；

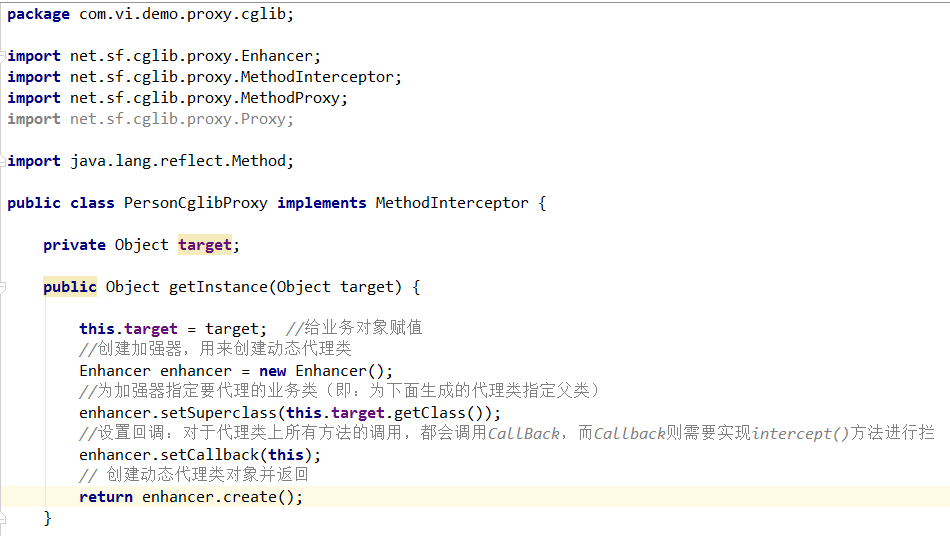
Java demo：

FrameDigger\reflectDigger\src\main\java\com\vi\demo\proxy\cglib\PersonCglibProxy.java

先编写要进行代理的类Person：



然后编写代理类PersonCglibProxy，代理类需要实现接口：MethodInterceptor：





PersonCglibProxy类要实现方法intercept，该方法可以在调用被代理类的前后进行一些操作（跟jdk代理类似）。

另外有个特别的地方是，在PersonCglibProxy的main方法中，调用getInstance的方法实现代理时，该方法有一个步骤：

enhancer.setSuperclass(**this**.**target**.getClass());

要求，传入的target对象（这里指Person），必须要又一个无参的构造器，否则会报错。