java静态代理和动态代理的区别

JAVA的静态代理与动态代理比较  
1.静态代理类：  
由程序员创建或由特定工具自动生成源代码，再对其编译。在程序运行前，代理类的.class文件就已经存在了。动态代理类：在程序运行时，运用反射机制动态创建而成。  
由此可见，代理类可以为委托类预处理消息、把消息转发给委托类和事后处理消息等。  
例程1 HelloService.java  
package proxy;  
import java.util.Date;  
public interface HelloService{  
public String echo(String msg);  
public Date getTime();  
}  
2.动态代理类  
与静态代理类对照的是动态代理类，动态代理类的字节码在程序运行时由Java反射机制动态生成，无需程序员手工编写它的源代码。动态代理类不仅简化了编程工作，而且提高了软件系统的可扩展性，因为Java 反射机制可以生成任意类型的动态代理类。java.lang.reflect 包中的Proxy类和InvocationHandler 接口提供了生成动态代理类的能力。  
  
Proxy类提供了创建动态代理类及其实例的静态方法。  
（1）getProxyClass()静态方法负责创建动态代理类，它的完整定义如下：  
public static Class<?> getProxyClass(ClassLoader loader, Class<?>[] interfaces) throws IllegalArgumentException  
参数loader 指定动态代理类的类加载器，参数interfaces 指定动态代理类需要实现的所有接口。  
（2）newProxyInstance()静态方法负责创建动态代理类的实例，它的完整定义如下：  
public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces, InvocationHandler handler) throws  
IllegalArgumentException  
参数loader 指定动态代理类的类加载器，参数interfaces 指定动态代理类需要实现的所有接口，参数handler 指定与动态代理类关联的 InvocationHandler 对象。  
以下两种方式都创建了实现Foo接口的动态代理类的实例：  
/\*\*\*\* 方式一 \*\*\*\*/  
//创建InvocationHandler对象  
InvocationHandler handler = new MyInvocationHandler(...);  
//创建动态代理类  
Class proxyClass = Proxy.getProxyClass(Foo.class.getClassLoader(), new Class[] { Foo.class });  
//创建动态代理类的实例  
Foo foo = (Foo) proxyClass.getConstructor(new Class[] { InvocationHandler.class }).  
newInstance(new Object[] { handler });  
/\*\*\*\* 方式二 \*\*\*\*/  
//创建InvocationHandler对象  
InvocationHandler handler = new MyInvocationHandler(...);  
//直接创建动态代理类的实例  
Foo foo = (Foo) Proxy.newProxyInstance(Foo.class.getClassLoader(),new Class[] { Foo.class }, handler);  
由Proxy类的静态方法创建的动态代理类具有以下特点：  
动态代理类是public、final和非抽象类型的；  
动态代理类继承了java.lang.reflect.Proxy类；  
动态代理类的名字以“$Proxy”开头；  
动态代理类实现getProxyClass()和newProxyInstance()方法中参数interfaces指定的所有接口。