# java的动态代理机制详解

在java的动态代理机制中，有两个重要的类或接口，一个是InvocationHandler(Interface)、另一个则是Proxy(Class)，这一个类和接口是实现我们动态代理所必须用到的。

每一个动态代理类都必须要实现InvocationHandler这个接口，并且每个代理类的实例都关联到了一个handler，当我们通过代理对象调用一个方法的时候，这个方法的调用就会被转发为由InvocationHandler这个接口的invoke方法来进行调用。下面看看InvocationHandler这个接口的唯一一个方法invoke方法：

```
Object invoke(Object proxy,Method method,Object[] args) throws Throwable

//proxy: 指代我们所代理的那个真实对象
//method: 指代的是我们所要调用真实对象的某个方法的Method对象
//args: 指代的是调用真实对象某个方法时接受的参数
```

**接下来看看Proxy这个类**

Proxy这个类的作用就是用来动态创建一个代理对象的类，它提供了许多的方法，但是我们用的最多的就是newProxyInstance这个方法：

```
public static Object newProxyInstance(ClassLoader loader,Class<?>
  interfaces,InvocationHandler h) throws IllegalArgumentException
//
//参数含义：
//loader:     一个ClassLoader对象，定义了由哪个ClassLoader对象来对生成的代理对象进行加载
//interfaces: 一个Interfaces对象的数组，表示的是将要给需要代理的对象提供一组什么接口，如果提供了
          一组接口给它，那么这个代理对象就宣称实现了该接口，这样就能调用这组接口中的方法
//h:          一个InvocationHandler对象，表示的是当这个动态代理对象在调用方法的时候，会关联到哪一
        个InvocationHandler对象上。
```

该方法的作用就是得到一个动态的代理对象。

下面通过一个实例来看看动态代理是什么样子的：

首先定义一个Subject类型的接口，为其声明了两个方法：

```
public interface Subject{
    public void rent();
    public void hello(String str);
}
```

接着定义一个类来实现这个接口

```
public class SubjectImpl implements Subject{
    @Override
    public void rent() {
        System.out.println("rent.....");
    }

    @Override
    public void hello(String str) {
        System.out.println(str+"hello.....");
    }
}
```

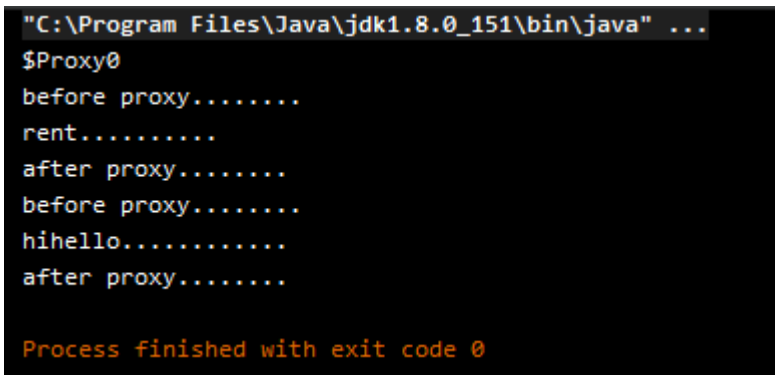
下面就要定义一个动态代理类了，每一个动态代理类都必须要实现InvocationHandler这个接口

```
public class DynamicProxy implements InvocationHandler{
    private Object subject;//我们要代理的真实对象
    public DynamicProxy(Object subject){
        this.subject = subject;
    }
    @Override
    public Object invoke(Object object,Method method,Object[] args) throws Throwable{
        //do something before
         System.out.println("before proxy.....");
        method.invoke(subject,args);
        //do something after
         System.out.println("after proxy.....");
        return null;
    }
}
```

最后来看看Client类

```
public class Client{
    Subject realSubject = new RealSubject();//要代理的真实对象
    InvocationHandler handler = new DynamicProxy(realSubject);
    Subject subject =
(Subject)Proxy.newProxyInstance(handler.class.getClassLoader(),realSubject.getClass().getInterfa
ces(),handler);
    System.out.println(subject.getClass().getName());
    subject.rent();
    subject.hello("hi");
}
```

**输出结果如下：**




通过Proxy.newProxyInstance创建的代理对象是在jvm运行时动态生成的一个对象，它并不是我们的InvocationHandler类型，也不是我们定义的那组接口的类型，而是在运行时动态生成的一个对象，并且命名方式都是以$开头，Proxy为中，最后一个数字表示对象的标号，可以递增。