动态代理 原理简析

一、概述

1.动态编译 JavaCompiler.CompilationTask 动态编译想理解自己查 API 文档

2.反射被代理类 主要使用 Method.invoke(Object o,Object... args);对带有指定参数的指定对象调用由此 Method 对象表示的底层方法。

3.类的加载 URLClassLoader 可以加载硬盘任意位置的.java 文件。class.getClassLoader 只能加载 classPath 目录下的类。

动态代理可以理解为 动态生成发射代理的类。这其中可以动态增加逻辑操作。比如日志的打印，事物的处理等。spring 的 AOP 操作也是动态代理的。

二、创建业务接口

假设我们有一个接口 GrowAble 可成长的。

```
1.package com.cn;
2.
3.public interface GrowAble {
4.    void growUp();
5.}
```

一棵小树苗实现了这个接口

```
1.package com.cn;
2.public class Tree implements GrowAble {
3.  @Override
4.  public void growUp() {
5.      System.out.println('I am a tree , I'm grow up!');
6.  }
7.
8.}
```

这时我们想不在不改变源码的情况下想知道树长了多少这个操作？

我们需要一个转换接口。

```
1.package com.cn;
2.import java.lang.reflect.Method;
3.
4.public interface InvactionHandle {
5.  void invoke(Object o,Method m);
6.}
```

一个实现接口类。

```
01.package com.cn;
02.import java.lang.reflect.Method;
03.import java.util.Random;
04.
05.public class HeightInvactionHandle implements InvactionHandle {
06.     @Override
07.     public void invoke(Object c, Method m) {
08.         try {
09.             m.invoke(this.o);
10.             System.out.println('这棵树长了' + new Random().nextInt(9527)+'
米！！！' );
11.         } catch (Exception e) {
12.             e.printStackTrace();
13.         }
14.     }
15.     private Object o;
16.     public HeightInvactionHandle(Object o) {
17.         super();
18.         this.o = o;
19.     }
20.}
```

三、其他重要类

现在最重要的 Proxy 类了。把上述两个接口接口起来。

```
01.package com.cn;
```

```
02.import java.io.File;
03.import java.io.FileWriter;
04.import java.lang.reflect.Constructor;
05.import java.lang.reflect.Method;
06.import java.net.URL;
07.import java.net.URLClassLoader;
08.import javax.tools.JavaCompiler;
09.import javax.tools.JavaFileObject;
10.import javax.tools.StandardJavaFileManager;
11.import javax.tools.ToolProvider;
12.import javax.tools.JavaCompiler.CompilationTask;
13./**
14.* 动态代理
15.* @author 灵台方寸小道士
16.*/
17.public class Proxy {
18.public static Object getNewInstance(Class<?> c,Object
object) throws Exception {
19.String path = System.getProperty('user.dir') + File.separator + 'mybin'
20.+ File.separator + 'com' + File.separator + 'cn'
21.+ File.separator;
22.String fileName = '$Proxy.java';
23.String nextLine = System.getProperty('line.separator');
24.// create java File
25.String fileValue = 'package com.cn;'+nextLine+
26.'import com.cn.*;'+nextLine+
27.'import java.lang.reflect.Method;'+nextLine+
28.'public class $Proxy implements '+ c.getName() +'{'+nextLine+
29.'    private InvactionHandle h;'+nextLine+
30.'    public $Proxy(InvactionHandle hin)'+ nextLine+
31.'     {'+nextLine+
32.'           this.h = hin;'+nextLine+
33.'     }'+nextLine;
34.Method[] methods = c.getDeclaredMethods();
35.for (Method m:methods) {
36.fileValue += '    public '+ m.getReturnType()+' '+m.getName()+'()'+nextLine+
```

```
37.'     {'+nextLine+
38.'          try{             '+nextLine+
39.//测试方法不带参数  所以 new Class<?>[]{}空参数传入
40.'          Method me =
'+c.getName()+'.class.getDeclaredMethod(''+m.getName()+'',  new
Class<?>[]{});'+nextLine+
41.'          h.invoke(this,me);'+nextLine+
42.'          }catch(Exception e){ '+nextLine+
43.'           e.printStackTrace(); }'+nextLine+
44.'     }'+nextLine;
45.}
46.fileValue +='}'+nextLine;
47.File f =  new File(path);//是否存在此目录
48.if (!f.exists())
49.f.mkdirs();
50.FileWriter writer = new FileWriter(new File(f,fileName));
51.writer.write(fileValue);
52.writer.flush();
53.writer.close();
54.System.out.println('***************     create java file
over      ******************');
55.// compiler 生成 class 文件  调取 javac 编译
56.JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
57.StandardJavaFileManager manager =
compiler.getStandardFileManager(null,null, null);
58.Iterable<? extends JavaFileObject> in = manager.getJavaFileObjects(path+
fileName);
59.CompilationTask task = compiler.getTask(null, manager, null, null,null, in);
60.task.call();
61.System.out.println('***************     complier class file
over      ******************');
62.
63.// loader 加载 class 文件 的第一种方法 URLClassLoader 可以 load 任意目录下的类!
64.URL[] urls = new URL[] { new URL('file:/' + System.getProperty('user.dir')
+ File.separator + 'mybin'+ File.separator) };
65.URLClassLoader loader = new URLClassLoader(urls);
```

```
66.Class<?> d = loader.loadClass('com.cn.$Proxy');
67.System.out.println('***************     loader class file
over      ******************');
68.
69.// newInstance class JVM
70.Constructor<?> con = d.getConstructor(InvactionHandle.class);
71.Object o = con.newInstance(object);
72.// newInstance...
73./**
74.加载 class 文件 的第二种方法 ClassLoader 只能 load 位于 classpath（src 目录）下的类
75.Class<?> second = Proxy.class.getClassLoader().loadClass('com.cn.$Proxy');
76.System.out.println(second.getSimpleName());
77.*/
78.return o;
79.}
80.}
```

JavaCompiler 是用于编译生成的 java 代码。在用 URLClassLoader 将 class 文件加载进内存。在实例化。

下面一个测试类 Client

```
01.package com.cn;
02.public class Client {
03.
04.public static void main(String[] args) throws Exception {
05.Tree tree = new Tree();
06.InvactionHandle handle = new HeightInvactionHandle(tree);
07.GrowAble gro = (GrowAble)Proxy.getNewInstance(GrowAble.class, handle);
08.gro.growUp();
09.System.out.println('测试结束');
10.}
11.}
```

运行结果

```
1.***************     create java file over      ******************
2.***************     complier class file over      ******************
3.***************     loader class file over      ******************
4.I am a tree , I'm grow up!
5.这棵树长了2174米！！！
6.测试结束
```

四、使用 JDK 来做

```
01.package com.cn;
02.import java.lang.reflect.InvocationHandler;
03.import java.lang.reflect.Method;
04.import java.util.Random;
05.
06.public class JDKInvocationHandle implements InvocationHandler {
07.
08.private Object o;
09.public JDKInvocationHandle(Object o)
10.{
11.super();
12.this.o = o;
13.}
14.@Override
15.public Object invoke(Object proxy, Method method, Object[] args)
16.throws Throwable {
17.Object result =  null;
18.try {
19.result = method.invoke(o,args);
20.System.out.println('这棵树长了' + new Random().nextInt(9527)+'米！！！' );
21.} catch (Exception e) {
22.e.printStackTrace();
23.}
24.return result;
25.}
26.}
```

测试类

```
01.package com.cn;
02.public class Client2 {
03.
04.public static void main(String[] args) {
05.java.lang.reflect.InvocationHandler h
= new JDKInvocationHandle(new Tree());
06.GrowAble gro = (GrowAble) java.lang.reflect.Proxy.newProxyInstance(
07.GrowAble.class.getClassLoader(),
08.new Class[] { GrowAble.class },
09.h);
10.gro.growUp();
11.System.out.println('测试结束');
12.}
13.}
```

运行结果

```
view sourceprint?
1.I am a tree , I'm grow up!
2.这棵树长了 726 米！！！
3.测试结束
```