作者：Intopass  
链接：https://www.zhihu.com/question/20794107/answer/75164285  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

① 首先你要明白静态代理的作用  
我们有一个字体提供类，有多种实现（从磁盘，从网络，从系统）

public interface FontProvider {

Font getFont(String name);

}

public abstract class ProviderFactory {

public static FontProvider getFontProvider() {

return new FontProviderFromDisk();

}

}

public class Main() {

public static void main(String[] args) {

FontProvider fontProvider = ProviderFactory.getFontProvider();

Font font = fontProvider.getFont("微软雅黑");

......

}

}

现在我们希望给他加上一个缓存功能，我们可以用静态代理来完成

public class CachedFontProvider implements FontProvider {

private FontProvider fontProvider;

private Map<String, Font> cached;

public CachedFontProvider(FontProvider fontProvider) {

this.fontProvider = fontProvider;

}

public Font getFont(String name) {

Font font = cached.get(name);

if (font == null) {

font = fontProvider.getFont(name);

cached.put(name, font);

}

return font;

}

}

/\* 对工厂类进行相应修改，代码使用处不必进行任何修改。

这也是面向接口编程以及工厂模式的一个好处 \*/

public abstract class ProviderFactory {

public static FontProvider getFontProvider() {

return new CachedFontProvider(new FontProviderFromDisk());

}

}

当然，我们直接修改FontProviderFromDisk类也可以实现目的，但是我们还有FontProviderFromNet, FontProviderFromSystem等多种实现类，一一修改太过繁琐且易出错。  
况且将来还可能添加日志，权限检查，异常处理等功能显然用代理类更好一点。  
  
② 然而为什么要用动态代理？  
考虑以下各种情况，有多个提供类，每个类都有getXxx(String name)方法，每个类都要加入缓存功能，使用静态代理虽然也能实现，但是也是略显繁琐，需要手动一一创建代理类。

public abstract class ProviderFactory {

public static FontProvider getFontProvider() {...}

public static ImageProvider getImageProvider() {...}

public static MusicProvider getMusicProvider() {...}

......

}

使用动态代理怎么完成呢？

public class CachedProviderHandler implements InvocationHandler {

private Map<String, Object> cached = new HashMap<>();

private Object target;

public CachedProviderHandler(Object target) {

this.target = target;

}

public Object invoke(Object proxy, Method method, Object[] args)

throws Throwable {

Type[] types = method.getParameterTypes();

if (method.getName().matches("get.+") && (types.length == 1) &&

(types[0] == String.class)) {

String key = (String) args[0];

Object value = cached.get(key);

if (value == null) {

value = method.invoke(target, args);

cached.put(key, value);

}

return value;

}

return method.invoke(target, args);

}

}

public abstract class ProviderFactory {

public static FontProvider getFontProvider() {

Class<FontProvider> targetClass = FontProvider.class;

return (FontProvider) Proxy.newProxyInstance(targetClass.getClassLoader(),

new Class[] { targetClass },

new CachedProviderHandler(new FontProviderFromDisk()));

}

}

③ 这也是为什么Spring这么受欢迎的一个原因  
Spring容器代替工厂，Spring AOP代替JDK动态代理，让面向切面编程更容易实现。  
在Spring的帮助下轻松添加，移除动态代理，且对源代码无任何影响。

[https://pic4.zhimg.com/8bba599f5af49ac53e5e8f4b27b7b4c3_s.jpg](http://www.zhihu.com/people/yuan-yi-jing-61)

[小叶子](http://www.zhihu.com/people/yuan-yi-jing-61)

头像好评\_(•̀ω•́ 」∠)\_

0 赞7 月前回复赞举报

[https://pic4.zhimg.com/cf75ea622ce5efd6e717da98107c9463_s.jpg](http://www.zhihu.com/people/crackme)

[CrackMe](http://www.zhihu.com/people/crackme)

前面那个不应该是装饰模式么？添加了新功能啊

0 赞7 月前回复赞举报

[https://pic1.zhimg.com/1a35d7494dd69e955a42e8b4039ed370_s.jpg](http://www.zhihu.com/people/intopass)

[Intopass](http://www.zhihu.com/people/intopass)（作者） 回复 [CrackMe](http://www.zhihu.com/people/crackme)查看对话

装饰模式本身就可以理解为静态代理。

1 赞7 月前回复赞举报

[https://pic3.zhimg.com/73147ec74283516d5771066ec5b0eac6_s.jpg](http://www.zhihu.com/people/guo-wu-xin)

[郭无心](http://www.zhihu.com/people/guo-wu-xin)

CachedFontProvider 在静态代理的例子当中我觉得应该是static的，Map对象也应该设置成static的，不然你的工厂类每次都new 一个新的，起不到cache缓存的作用

0 赞7 月前回复赞举报

[https://pic1.zhimg.com/1a35d7494dd69e955a42e8b4039ed370_s.jpg](http://www.zhihu.com/people/intopass)

[Intopass](http://www.zhihu.com/people/intopass)（作者） 回复 [郭无心](http://www.zhihu.com/people/guo-wu-xin)查看对话

只是举个例子，不过如果用Spring管理的话，用单例模式+非静态也行。

0 赞7 月前回复赞举报

[https://pic4.zhimg.com/7994af253_s.jpg](http://www.zhihu.com/people/liu-duo)

[刘铎](http://www.zhihu.com/people/liu-duo)

静态代理例子中的CachedFontProvider类应该是实现getFont()方法，而不是getFontProvider()方法，可能是作者手误写错了:)还有一点疑问，在CachedProviderHandler这个例子中，似乎cached变量定义为Map<String, Object>就足够了

0 赞4 月前回复赞举报

[https://pic1.zhimg.com/1a35d7494dd69e955a42e8b4039ed370_s.jpg](http://www.zhihu.com/people/intopass)

[Intopass](http://www.zhihu.com/people/intopass)（作者） 回复 [刘铎](http://www.zhihu.com/people/liu-duo)查看对话

代码是我手打的，确实第一点你说的对，我写错了。不过第二点应该没错，毕竟要同时未多个方法提供缓存，那么就有可能重名。

0 赞4 月前回复赞举报

[https://pic4.zhimg.com/7994af253_s.jpg](http://www.zhihu.com/people/liu-duo)

[刘铎](http://www.zhihu.com/people/liu-duo) 回复 [Intopass](http://www.zhihu.com/people/intopass)（作者）查看对话

关于第二点，我认为是由于CachedProviderHandler类只有一个委托类对象（即Object target），所以在新建CachedProviderHandler类对象的时候，就决定了只能传过去指定类型的Provider了（即要么是FontProvider，要么是ImageProvider...），那么用来缓存的cached变量也就只可能缓存指定类型的实体了（即要么是Font、要么是Image...），所以cached变量定义为Map<Method, Map<String, Object>>似乎多余了:)当然，如果cached变量是static的，即用来全局缓存的，那这样定义就是没问题的。不知道我表达清楚了没:)

0 赞4 月前回复赞举报

[https://pic1.zhimg.com/1a35d7494dd69e955a42e8b4039ed370_s.jpg](http://www.zhihu.com/people/intopass)

[Intopass](http://www.zhihu.com/people/intopass)（作者） 回复 [刘铎](http://www.zhihu.com/people/liu-duo)查看对话

恩，你说的对。

0 赞4 月前