***不同点:***

JDK动态代理只能通过接口进行代理（因此你的目标类需要实现一个接口，然后该接口也由代理类实现）。

CGLIB（和javassist）可以通过子类化创建代理。在这种情况下，代理成为目标类的子类。不需要接口。

所以Java动态代理可以代理：public class Foo implements iFoo其中CGLIB可以代理：public class Foo

***功能差异:***

JDK 代理允许在子类化时实现任何接口集Object。任何接口方法，加上Object::hashCode，Object::equals然后Object::toString转发到[InvocationHandler](http://docs.oracle.com/javase/8/docs/api/java/lang/reflect/InvocationHandler.html).此外，还java.lang.reflect.Proxy实现了标准库接口。

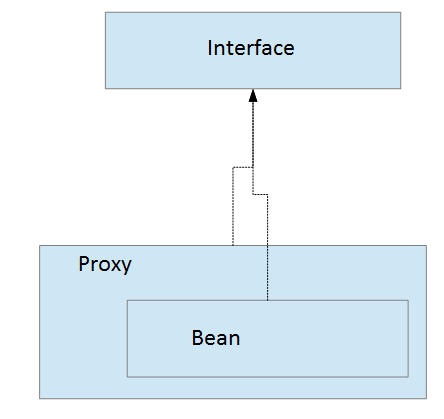
cglib 允许您在子类化任何非最终类的同时实现任何接口集。此外，方法可以选择性地重写，即并非所有非抽象方法都需要被拦截。此外，存在实现方法的不同方式。它还提供了一个InvocationHandler类（在不同的包中），但它也允许通过使用更高级的拦截器来调用超级方法，例如MethodInterceptor.此外，cglib 可以通过专门的拦截

***图文详解:***

**动态代理：使用JDK Reflection API**在运行时动态实现接口。

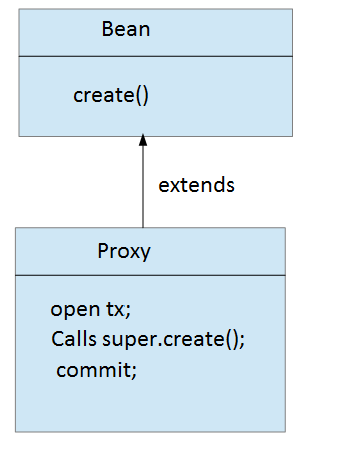
**示例：** Spring 使用动态代理进行事务，如下所示：

生成的代理位于 bean 之上。它为 bean 添加了跨国行为。这里代理使用 JDK Reflection API 在运行时动态生成。



当应用程序停止时，代理将被销毁，文件系统上将只有接口和bean。

在上面的例子中我们有接口。但在大多数接口的实现中并不是最好的。所以bean没有实现接口，在这种情况下我们使用继承,**CGLIB代理:**

[](https://i.sstatic.net/fsOVh.png)

为了生成此类代理，Spring 使用名为**CGLib**的第三方库。

CGLib（**代码**生成**库）**建立在[ASM](https://en.wikipedia.org/wiki/ObjectWeb_ASM)之上，主要用于生成代理扩展bean并在代理方法中添加bean行为。