JVM如何实现invokedynamic

Java7中引入了更加底层、更加灵活的方法抽象：方法句柄（MethodHandle）

## 方法句柄

是一个强类型的，能够被直接执行的引用。该引用可以指向常规的静态方法或者实例方法，也可以指向构造器或者字段。当指向字段的时候，方法句柄实则指向包含字段访问字节码的虚构方法，语义上等价于目标字段的getter或者setter方法。

注意：它并不会直接指向目标字段所在的类的getter/setter，毕竟你无法保证已有的getter/setter方法就是在访问目标字段。

方法句柄（MethodType）是由**所指向方法的参数类型以及返回类型组成的**。它是来确认**方法句柄是否适配的唯一**关键。当使用方法句柄时候，我们实际并不关心方法句柄所指的方法的类名或者方法名。

## MethodHandles.Lookup类来完成

**public class Foo** {  
 **public static void** bar(**Object** o){}  
 **public static MethodHandles**.**Lookup** lookup(){  
 **return MethodHandles**.lookup();  
 }  
  
 **public static void** main(**String**[] args) **throws Exception** {  
 // 获取方法句柄的不同方式  
 **MethodHandles**.**Lookup** l = **Foo**.lookup();//具备Foo的访问权限  
 **Method** m = **Foo**.**class**.getDeclaredMethod("bar", **Object**.**class**);  
 MethodHandle mh0 = l.unreflect(m);  
  
 **MethodType** t = **MethodType**.methodType(**void**.**class**, **Object**.**class**);  
 MethodHandle mh1 = l.findStatic(**Foo**.**class**, "bar", t);  
 }  
}

方法句柄同样也有权限问题。但是与反射API不同，其权限检查是在句柄创建的阶段完成的。在实际调用过程中，java虚机并不会检查方法句柄的权限。如果句柄被多次调用的话，那么和反射调用相比，它将省下重复权限检查的开销。

访问权限不取决于方法句柄的创建位置，而是lookup对象的创建位置。

## 方法句柄操作

权限调用分为两种：

### InvokeExact

必须要有严格匹配的参数类型，若一个句柄接受Object类型的参数，那么传入String类型的时候，都会报错。正确的方式是将String转化为Object类型。

public final native @PolymorphicSignature Object invokeExact(Object... args) throws Throwable;

### 注解@PolymorphicSignature

Java会根据编译器传参类型声明来生成方法描述符，而不是目标方法声明的描述符

public void test(MethodHandle mh, String s) throws Throwable {

mh.invokeExact(s);

mh.invokeExact((Object) s);

}

// 对应的 Java 字节码

public void test(MethodHandle, String) throws java.lang.Throwable;

Code:

0: aload\_1

1: aload\_2

2: invokevirtual MethodHandle.invokeExact:(Ljava/lang/String;)V

5: aload\_1

6: aload\_2

7: invokevirtual MethodHandle.invokeExact:(Ljava/lang/Object;)V

10: return

### Invoke

自动适配参数类型，invoke会调用MethodHandle.asType方法，生成一个适配器方法句柄，对传入的参数进行适配，再调用原方法句柄。调用原方法句柄的返回值同样也会先进行适配，然后再返回给调用者。

## 方法句柄的实现

### DirectMethodHandle

java.lang.Exception

at cn.lxj.jvm40courses.course8\_1.FooV2.bar(FooV2.java:14)

at cn.lxj.jvm40courses.course8\_1.FooV2.main(FooV2.java:21)

### -XX:+ShowHiddenFrames

打印虚机隐藏了的栈信息

java.lang.Exception

at cn.lxj.jvm40courses.course8\_1.FooV2.bar(FooV2.java:14)

at java.lang.invoke.LambdaForm$DMH/1581781576.invokeStatic\_L\_V(LambdaForm$DMH:1000010)

at java.lang.invoke.LambdaForm$MH/681842940.invokeExact\_MT(LambdaForm$MH:1000016)

at cn.lxj.jvm40courses.course8\_1.FooV2.main(FooV2.java:21)