链滴

自定义类加载器加载类的问题

作者：leopoldwu

原文链接：https://ld246.com/article/1497673963301

来源网站：链滴

一、创建自定义类加载器

  自定义了一个类加载器MyClassLoader，其中实现是通过读取class文件为字数组，然后通过ClassLoader中的defineClass(String name,byte[],int off,int len)把字节数组中的内转换成Java类，返回的是Class类的实例，代码如下：

```
package priv.leopold.tool;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class MyClassLoader extends ClassLoader {
    public Class<?> load(File classFile) throws ClassNotFoundException {
        byte[] classData = getClassData(classFile);
        if (classData == null) {
            throw new ClassNotFoundException();
        } else {
            return defineClass(null, classData, 0, classData.length);
        }
    }
    private byte[] getClassData(File classFile) {
        try {
            InputStream ins = new FileInputStream(classFile);
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            int bufferSize = 4096;
            byte[] buffer = new byte[bufferSize];
            int bytesNumRead = 0;
            while ((bytesNumRead = ins.read(buffer)) != -1) {
                baos.write(buffer, 0, bytesNumRead);
            }
            ins.close();
            return baos.toByteArray();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

二、测试自定义类加载器

先创建两个MyClassLoader实例：

```
    MyClassLoader  loader1 = new MyClassLoader();
    MyClassLoader  loader2 = new MyClassLoader();
```

然后通过这两个类加载器的实例加载同一个类，并通过Class对象创建实例：

```
    Class<?> class1 = loader1.load(classFile);       //第一个类加载器加载类
    Object obj1 = class1.newInstance();
```

```
Object obj2 = class1.newInstance();

Class<?> class2 = loader2.load(classFile);       //第二个类加载器加载类
Object obj3 = class2.newInstance();
```

接下来执行类中的方法：

```
Method method = class1.getMethod("setInstance", Object.class);
method.invoke(obj1, obj2);
method.invoke(obj1, obj3);
```

其中测试的类为Sample，代码如下：

```
public class Sample {
    private Sample instance;
    public Sample getInstance() {
        return instance;
    }
    public void setInstance(Object instance) {
        this.instance = (Sample) instance;
    }
}
```

执行上面的代码后会抛出异常：

```
Exception in thread "main" java.lang.reflect.InvocationTargetException
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
at java.lang.reflect.Method.invoke(Unknown Source)
at test.Test.main(Test.java:28)
Caused by: java.lang.ClassCastException: bean.Sample cannot be cast to bean.Sample
at bean.Sample.setInstance(Sample.java:12)
... 5 more
```

    引起上面异常的是这
代码：this.instance = (Sample) instance;

虽然是用同一个类文件（这里用的是Sample.class文件）加载出来的Class对象，但通过上面的这句码抛出的异常可以发现，它们创建的实例属于不同类型（即使这里都叫做bean.Sample）。

三、分析原因

    对于某个特定的类加
器来说，一个Java类只能被载入一次，也就是说在Java虚拟机中，类的完整标识是（classLoader，pa
kage，className）。

另外一个类可以被不同的类加载器加载，但是，同一个类，由不同的类加载器实例加载的话，会在方区产生两个不同的类，彼此不可见，并且在堆中生成不同Class实例。

    如果用的类加载器没
重写ClassLoader中的getSystemClassLoader方法，那么用的都是AppClassLoader的同一个实例进类的加载的，所以加载同一个类在内存中只有一个类，即所有通过正常双亲委派模式的类加载器加载classpath下的和ext下的所有类在方法区都是同一个类，堆中的Class实例也是同一个。