链滴

自定义类加载器加载类遇到的问题

作者：leopoldwu

原文链接：https://ld246.com/article/1497183580008

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文件为字节数组，然后通过ClassLoader中的defineClass(String name,byte[],int off,int len)把字节数组中的内容转换成Java类，返回是Class类的实例，代码如下：

```
1. package priv.leopold.tool;
2. import java.io.DataInputStream;
3. import java.io.File;
4. import java.io.FileInputStream;
5. import java.util.ArrayList;
6. import java.util.HashMap;
7. import java.util.List;
8. import java.util.Map;
9. public class MyClassLoader extends ClassLoader {
10.  public Class load(File classFile) throws ClassNotFoundException {
11.  byte[] classData = getClassData(classFile);
12.  if (classData == null) {
13.  throw new ClassNotFoundException();
14.  } else {
15.  return defineClass(null, classData, 0, classData.length);
16.  }
17.  }
18.  private byte[] getClassData(File classFile) {
19.  try {
20.  InputStream ins = new FileInputStream(classFile);
21.  ByteArrayOutputStream baos = new ByteArrayOutputStream();
22.  int bufferSize = 4096;
23.  byte[] buffer = new byte[bufferSize];
24.  int bytesNumRead = 0;
25.  while ((bytesNumRead = ins.read(buffer)) != -1) {
26. baos.write(buffer, 0, bytesNumRead);
27.  }
28. ins.close();
29.  return baos.toByteArray();
30.  } catch (IOException e) {
31. e.printStackTrace();
32.  }
```

```
33.    return  null;
34.    }
35. }
```

二、测试自定义类加载器

先创建两个MyClassLoader实例：

```
1. MyClassLoader loader1 =  new  MyClassLoader();
2. 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：

```
3. Class class1 = loader1.load(classFile);  //第一个类加载器加载类
4. Object obj1 = class1.newInstance();
5. Object obj2 = class1.newInstance();
6.
7. Class class2 = loader2.load(classFile);  //第二个类加载器加载类
8. Object obj3 = class2.newInstance();
```

接下来执行类中的方法：

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

其中测试的类为Sample，代码如下：

```
12. public  class  Sample  {
13.    private  Sample instance;
14.    public  Sample getInstance()  {
15.    return instance;
16.    }
17.    public  void setInstance(Object instance)  {
18.    this.instance =  (Sample) instance;
19.    }
20. }
```

执行上面的代码后会抛出异常：

```
1. Exception in thread "main" java.lang.reflect.InvocationTargetException
2.  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native  Method)
3.  at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown  Source)
4.  at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown  Source)
5.  at java.lang.reflect.Method.invoke(Unknown  Source)
```

6. at test.Test.main(Test.java:28)
7. Caused by: java.lang.ClassCastException: bean.Sample cannot be cast to bean.Sample
8. at bean.Sample.setInstance(Sample.java:12)
9. ... 5 more

引起上面异常的是这句代码：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载器来说，一个Java类只能被载入一次，也就是说在Java虚拟机中，类的完整标是（classLoader，package，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有重写ClassLoader中的getSystemClassLoader方法，那么用的都是AppClassoader的同一个实例进行类的加载的，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派模式的类加载器加载的classpath下的和ext下的所有类在方法区都是同一个类，堆中的Class实例也同一个。