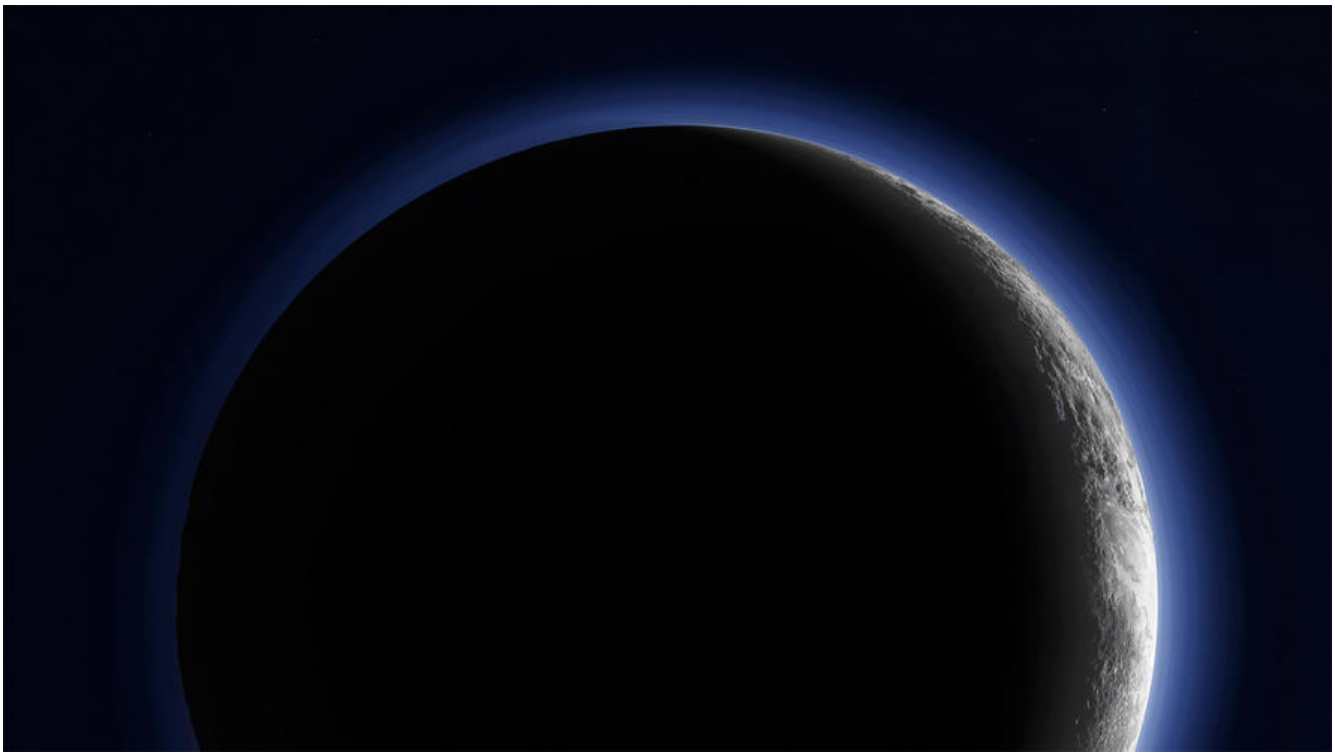
自定义类加载器例子（对 class 文件进行解密操作）

作者：gitsilence
原文链接：https://ld246.com/article/1613490773232
来源网站：链滴



摘抄自 https://xz.aliyun.com/t/9002#toc-16

案例

- 首先创建一个需要加密的 测试类

```
package cn.lacknb.test.classloader;

/**
 * @Desc: cypher 密码
 *  自定义 类加载器: 将此 的 class文件进行加密，然后使用自定义的
 *      类加载器 进行加载。
 * @Author: gitsilence
 * @Date: 2021/2/16
 */
public class CypherTest {

    public static void main(String[] args) {
        System.out.println("This experiment test is successful");
    }

}
```

- 通过上面的类生成class文件。
- 创建一个用来加密 class 文件的类，其实就是 进行按位 异或操作。

```
package cn.lacknb.test.classloader;

import java.io.*;
```

```
/**
 * @Desc: 加密 类 ， 使用逐位 和 1111 1111(0xff) 进行异或。
 * @Author: gitsilence
 * @Date: 2021/2/16
 */
public class Encryption {

    private  static final String CLASS_PATH = "/media/gitsilence/nie475/javaProject/Java_Threa
/01-Threadhello/target/classes/cn/lacknb/test/classloader/CypherTest.class";

    public static void main(String[] args) {
        encode(new File(CLASS_PATH), new File("/media/gitsilence/nie475/javaProject/Java_Thre
d/01-Threadhello/src/main/java/cn/lacknb/test/classloader/CypherTest.class"));
    }

    public static void encode (File src, File dest) {
        FileInputStream fis = null;
        FileOutputStream fos = null;
        try {
            fis = new FileInputStream(src);
            fos = new FileOutputStream(dest);
            int temp = -1;
            // 读取 一个 字节
            while ((temp = fis.read()) != -1) {
                // 0xff 为16进制、 二进制为：1111 1111
                // 按位  进行 异或 运算， 再进行有一次异或 恢复。
                fos.write(temp ^ 0xff);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                    fos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        System.out.println("current class is encoded successfully!");
    }

}
```

- 执行 对class文件的加密后，自定义 一个类加载器，在加载前进行 解密操作。

```
package cn.lacknb.test.classloader;

import java.io.*;
```

```java
/**
 * @Desc: 自定义 类加载器，解密class
 *      将Decryption 继承 ClassLoader类，之后覆盖findClass方法，
 *  并且 在findClass方法中调用defineClass()方法使用，最后加载我们自定义
 *  getClassData 方法进行 解密操作。
 * @Author: gitsilence
 * @Date: 2021/2/16
 */
public class Decryption extends ClassLoader {

    private String rootDir;

    public Decryption (String rootDir) {
        this.rootDir = rootDir;
    }

    /**
     * 重写覆盖 findClass
     * @param name
     * @return
     * @throws ClassNotFoundException
     */
    @Override
    protected Class<?> findClass(String name) throws ClassNotFoundException {
        // 先从 已加载的类中 查找 是否已经加载。
        Class<?> clzz = findLoadedClass(name);
        if (clzz != null) {
            return clzz;
        } else {
            ClassLoader parent = this.getParent();
            try {
                clzz = parent.loadClass(name);
            } catch (ClassNotFoundException e) {
                System.out.println("父类无法加载你的class，捕获异常，继续运行。");
            }
            if (clzz != null) {
                System.out.println("父类加载成功");
                return clzz;
            } else {
                // 读取文件 转化为 字节数组
                byte[] classData = getClassData(name);
                if (classData == null) {
                    throw new ClassNotFoundException("not found class : " + name);
                } else  {
                    // 调用 defineClass方法
                    clzz = defineClass(name, classData, 0, classData.length);
                    return clzz;
                }
            }
        }
    }

    /**
     * 解密字节码 文件的内容
```

```
     * @param className
     * @return
     */
    public byte[] getClassData (String className) {
        String path = rootDir + "/" + className.replace(".", "/") + ".class";
        // 将流中的数据 转换为字节数组
        InputStream is = null;
        ByteArrayOutputStream baso = new ByteArrayOutputStream();
        try {
            is = new FileInputStream(path);
            byte[] buffer = new byte[1024];
            int temp = -1;
            while ((temp = is.read()) != -1) {
                // 与 二进制 1111 1111 进行 异或运算。
                baso.write(temp ^ 0xff);
            }
            return baso.toByteArray();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (is != null) {
                try {
                    is.close();
                    baso.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return null;
    }

}
```

- 测试

```
package cn.lacknb.test.classloader;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

/**
 * @Desc: 测试 加载 加密 后的class文件。
 * @Author: gitsilence
 * @Date: 2021/2/16
 */
public class ClassLoaderDemo {

    public static void main(String[] args) {
        // 解密加载器。
        Decryption decryption = new Decryption("/media/gitsilence/nie475/javaProject/Java_Thr
```

```
ad/01-Threadhello/src/main/java/");
        try {
            // 加载已经解密的 class文件
            Class<?> clzz = decryption.findClass("cn.lacknb.test.classloader.CypherTest");
            System.out.println("class is : " + clzz);
            // 通过反射 获取main方法
            Method main = clzz.getMethod("main", String[].class);
            // JDK1.4以后 把整个数组 作为一个参数
            // 按照1.4的语法，是数组中的每个元素对应一个参数
            // 1.5 必须兼容1.4，因此字符串数组会被拆分为 若干个单独的参数
            // 因此在给main方法传递参数时，不能使用代码 main.invoke(null, new String[]{"1", "2"});
            // javac 会按照1.4的语法进行编译，会出现参数数量不匹配的问题。
            main.invoke(null, (Object)new String[]{"1", "2"});
            // 解决办法：对我们要传的字符数组进行一次包装，把它包装到一个Object数组
            // 中，这时拆开后，得到唯一参数 就是字符串数组。
        } catch (ClassNotFoundException | NoSuchMethodException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }
    }

}
```

最后打印结果。

父类加载成功

class is : class cn.lacknb.test.classloader.CypherTest

This experiment test is successful