**package** helloGroovy

**import** org.codehaus.groovy.runtime.InvokerHelper

//1.groovy类解析

//GroovyClassLoader解析一段定义方法的文本代码并且可以创建Script调用方法

**class** a{

**static** **void** main(String[] rags){

GroovyClassLoader classLoader=**new** GroovyClassLoader()

Class<?> ruleClazz=classLoader.parseClass(

"def b(){println \"I am a groovy script b\"}\n"+

"void c(){println \"I am a groovy script c \"}\n"+

"private d(){println \"I am a groovy script d \"}\n" +

"def e(int param =1000){ println \"I am a groovy script e param default value is \" +param }\n"+

"String f(){ return \" return from method f()\"}"

)

Script script=InvokerHelper.*createScript*(ruleClazz,**new** Binding())

script.invokeMethod("b",**null**)

script.invokeMethod("c",**null**)

script.invokeMethod("d",**null**)

script.invokeMethod("e",**null**)

println script.invokeMethod("f",**null**)

}

}

//2.groovy表达式解析

**import** org.codehaus.groovy.control.CompilerConfiguration

**class** ab{

**static** **void** main(String[] rags){

CompilerConfiguration cfg=**new** CompilerConfiguration()

cfg.setScriptBaseClass(Script.**class**.getName())

**def** shell=**new** GroovyShell(cfg)

**def** script=shell.parse("print 'value' +value + '为:'\n value>0?'整数':'负数'") //为script对象添加执行代码

**def** binding =**new** Binding()

script.setBinding(binding)

binding.setVariable("value",-1) //输入不同变量，就会有不同效果

println script.run() //调用run方法执行代码

binding.setVariable("value",3)

println script.run()

println Eval.*xyz*(1,2,3,"x+y\*z")//简单的表达式脚本，对GroovyShell的包装

}

}