**JAVA嵌入运行Groovy脚本**

 最近设计一个数据统计系统,系统中上百种数据统计维度,而且这些数据统计的指标可能随时会调整.如果基于java编码的方式逐个实现数据统计的API设计,工作量大而且维护起来成本较高;最终确定为将"数据统计"的计算部分单独分离成脚本文件(javascript,或者Groovy),非常便捷了实现了"数据统计Task" 与 "数据统计规则(计算)"解耦,且可以动态的加载和运行的能力.顺便对JAVA嵌入运行Groovy脚本做个备忘.

    Java中运行Groovy,有三种比较常用的类支持:GroovyShell,GroovyClassLoader以及Java-Script引擎(JSR-223).

    1) GroovyShell: 通常用来运行"script片段"或者一些零散的表达式(Expression)

    2) GroovyClassLoader: 如果脚本是一个完整的文件,特别是有API类型的时候,比如有类似于JAVA的接口,面向对象设计时,通常使用GroovyClassLoader.

    3) ScriptEngine: JSR-223应该是推荐的一种使用策略.规范化,而且简便.

一.GroovyShell代码样例

*1) 简单的表达式执行,方法调用*

1. */\*\**
2. *\* 简答脚本执行*
3. *\* @throws Exception*
4. *\*/*
5. public static void evalScriptText() throws Exception{
6. *//groovy.lang.Binding*
7. Binding binding = new Binding();
8. GroovyShell shell = new GroovyShell(binding);
10. binding.setVariable("name", "zhangsan");
11. shell.evaluate("println 'Hello World! I am ' + name;");
12. *//在script中,声明变量,不能使用def,否则scrope不一致.*
13. shell.evaluate("date = new Date();");
14. Date date = (Date)binding.getVariable("date");
15. System.out.println("Date:" + date.getTime());
16. *//以返回值的方式,获取script内部变量值,或者执行结果*
17. *//一个shell实例中,所有变量值,将会在此"session"中传递下去."date"可以在此后的script中获取*
18. Long time = (Long)shell.evaluate("def time = date.getTime(); return time;");
19. System.out.println("Time:" + time);
20. binding.setVariable("list", new String[]{"A","B","C"});
21. *//invoke method*
22. String joinString = (String)shell.evaluate("def call(){return list.join(' - ')};call();");
23. System.out.println("Array join:" + joinString);
24. shell = null;
25. binding = null;
26. }

2)  伪main方法执行.

1. */\*\**
2. *\* 当groovy脚本,为完整类结构时,可以通过执行main方法并传递参数的方式,启动脚本.*
3. *\*/*
4. public static void evalScriptAsMainMethod(){
5. String[] args = new String[]{"Zhangsan","10"};*//main(String[] args)*
6. Binding binding = new Binding(args);
7. GroovyShell shell = new GroovyShell(binding);
8. shell.evaluate("static void main(String[] args){ if(args.length != 2) return;println('Hello,I am ' + args[0] + ',age ' + args[1])}");
9. shell = null;
10. binding = null;
11. }

 3)  通过Shell运行具有类结构的Groovy脚本

1. */\*\**
2. *\* 运行完整脚本*
3. *\* @throws Exception*
4. *\*/*
5. public static void evalScriptTextFull() throws Exception{
6. StringBuffer buffer = new StringBuffer();
7. *//define API*
8. buffer.append("class User{")
9. .append("String name;Integer age;")
10. *//.append("User(String name,Integer age){this.name = name;this.age = age};")*
11. .append("String sayHello(){return 'Hello,I am ' + name + ',age ' + age;}}\n");
12. *//Usage*
13. buffer.append("def user = new User(name:'zhangsan',age:1);")
14. .append("user.sayHello();");
15. *//groovy.lang.Binding*
16. Binding binding = new Binding();
17. GroovyShell shell = new GroovyShell(binding);
18. String message = (String)shell.evaluate(buffer.toString());
19. System.out.println(message);
20. *//重写main方法,默认执行*
21. String mainMethod = "static void main(String[] args){def user = new User(name:'lisi',age:12);print(user.sayHello());}";
22. shell.evaluate(mainMethod);
23. shell = null;
24. }

4)  方法执行和分部调用

1. */\*\**
2. *\* 以面向"过程"的方式运行脚本*
3. *\* @throws Exception*
4. *\*/*
5. public static void evalScript() throws Exception{
6. Binding binding = new Binding();
7. GroovyShell shell = new GroovyShell(binding);
8. *//直接方法调用*
9. *//shell.parse(new File(//))*
10. Script script = shell.parse("def join(String[] list) {return list.join('--');}");
11. String joinString = (String)script.invokeMethod("join", new String[]{"A1","B2","C3"});
12. System.out.println(joinString);
13. *////脚本可以为任何格式,可以为main方法,也可以为普通方法*
14. *//1) def call(){...};call();*
15. *//2) call(){...};*
16. script = shell.parse("static void main(String[] args){i = i \* 2;}");
17. script.setProperty("i", new Integer(10));
18. script.run();*//运行,*
19. System.out.println(script.getProperty("i"));
20. *//the same as*
21. System.out.println(script.getBinding().getVariable("i"));
22. script = null;
23. shell = null;
24. }

二. GroovyClassLoader代码示例

 1) 解析groovy文件

1. */\*\**
2. *\* from source file of \*.groovy*
3. *\*/*
4. public static void parse() throws Exception{
5. GroovyClassLoader classLoader = new GroovyClassLoader(Thread.currentThread().getContextClassLoader());
6. File sourceFile = new File("D:\\TestGroovy.groovy");
7. Class testGroovyClass = classLoader.parseClass(new GroovyCodeSource(sourceFile));
8. GroovyObject instance = (GroovyObject)testGroovyClass.newInstance();*//proxy*
9. Long time = (Long)instance.invokeMethod("getTime", new Date());
10. System.out.println(time);
11. Date date = (Date)instance.invokeMethod("getDate", time);
12. System.out.println(date.getTime());
13. *//here*
14. instance = null;
15. testGroovyClass = null;
16. }

2) 如何加载已经编译的groovy文件

1. public static void load() throws Exception {
2. GroovyClassLoader classLoader = new GroovyClassLoader(Thread.currentThread().getContextClassLoader());
3. BufferedInputStream bis = new BufferedInputStream(new FileInputStream("D:\\TestGroovy.class"));
4. ByteArrayOutputStream bos = new ByteArrayOutputStream();
5. for(;;){
6. int i = bis.read();
7. if( i == -1){
8. break;
9. }
10. bos.write(i);
11. }
12. Class testGroovyClass = classLoader.defineClass(null, bos.toByteArray());
13. *//instance of proxy-class*
14. *//if interface API is in the classpath,you can do such as:*
15. *//MyObject instance = (MyObject)testGroovyClass.newInstance()*
16. GroovyObject instance = (GroovyObject)testGroovyClass.newInstance();
17. Long time = (Long)instance.invokeMethod("getTime", new Date());
18. System.out.println(time);
19. Date date = (Date)instance.invokeMethod("getDate", time);
20. System.out.println(date.getTime());
22. *//here*
23. instance = null;
24. testGroovyClass = null;
25. }

三. ScriptEngine

1) pom.xml依赖

1. <dependency>
2. <groupId>org.codehaus.groovy</groupId>
3. <artifactId>groovy</artifactId>
4. <version>2.1.6</version>
5. </dependency>
6. <dependency>
7. <groupId>org.codehaus.groovy</groupId>
8. <artifactId>groovy-jsr223</artifactId>
9. <version>2.1.6</version>

</dependency>

2) 代码样例

1. public static void evalScript() throws Exception{
2. ScriptEngineManager factory = new ScriptEngineManager();
3. *//每次生成一个engine实例*
4. ScriptEngine engine = factory.getEngineByName("groovy");
5. System.out.println(engine.toString());
6. assert engine != null;
7. *//javax.script.Bindings*
8. Bindings binding = engine.createBindings();
9. binding.put("date", new Date());
10. *//如果script文本来自文件,请首先获取文件内容*
11. engine.eval("def getTime(){return date.getTime();}",binding);
12. engine.eval("def sayHello(name,age){return 'Hello,I am ' + name + ',age' + age;}");
13. Long time = (Long)((Invocable)engine).invokeFunction("getTime", null);
14. System.out.println(time);
15. String message = (String)((Invocable)engine).invokeFunction("sayHello", "zhangsan",new Integer(12));
16. System.out.println(message);
17. }

3) 实际案例，Java调用groovy文件里面的方法，并传递参数

1. static void simpleTest() throws IOException, InstantiationException, IllegalAccessException, ResourceException, ScriptException{
2. String[] roots = new String[] { "src/main/groovy/com/mobile263/billing/groovy/" };
3. *//通过指定的roots来初始化GroovyScriptEngine*
4. GroovyScriptEngine gse = new GroovyScriptEngine(roots);
5. GroovyObject groovyObject = (GroovyObject) gse.loadScriptByName("TestScript.groovy").newInstance();
6. String result = (String) groovyObject.invokeMethod("output", "hello");
7. System.out.println(result);
8. }
9. class TestScript {
10. static String output(def str){
11. println str;
12. return "hello"+str;
13. }
14. }