链滴

JAVA9 JShell 初体验

作者：liudongdong
原文链接：https://ld246.com/article/1506073849959
来源网站：链滴

JAVA 9 JShell 初体验

时隔三年之久，新的JAVASE版本JAVA9终于正式发布了，喜欢尝鲜的同学们就一起来看看JAVA9引了哪些新的特性吧。

A. 概括介绍

1. 模块系统

在java9中引入了module系统，来封装代码和数据，改善java应用程序的依赖和部署问题

2. REPL

属于java语言的交互式解释器，允许你快速验证原型和执行代码片段。目前一些小的，工具化，重复的任务主要使用Bash,Python等动态类型的语言去实现，以后应该会有同学用Java去写脚本(^▿^)

3. 语言变化

更简洁的try-with-resources语法，接口内部允许定义私有方法，匿名内部类支持diamond syntax等

4. 其他更新：Javascript引擎　Nashorn更新，支持了部分ES6特性，新的版本字符串，安全，并发方面的更新。全部新特性参见Java9新特性一览

B. 环境安装(CentOS Linux release 7.2示例)

1. Oracle JDK 9下载地址
2. 以管理员权限执行　rpm -ivh jdk-9_linux-x64_bin.rpm
3. 在终端输入java -version,验证安装

```
[ldd@localhost learn]$ java -version
java version "9"
Java(TM) SE Runtime Environment (build 9+181)
Java HotSpot(TM) 64-Bit Server VM (build 9+181, mixed mode)
```

C. REPL使用

C.1 打开和退出jshell

在终端输入jshell命令，按回车，即进入shell解释器

```
[ldd@localhost learn]$ jshell
|  Welcome to JShell -- Version 9
|  For an introduction type: /help intro
jshell>
```

输入/exit，按回车退出jshell

```
jshell> /exit
|  Goodbye
[ldd@localhost learn]$
```

C.2 Hello World

在终端输入jshell PRINTING,进入jshell解释器

```
[ldd@localhost learn]$ jshell PRINTING
|  Welcome to JShell -- Version 9
|  For an introduction type: /help intro

jshell> println("Hello World")
Hello World

jshell>
```

jshell解释器启动的完整语法为jshell [options] [load-files],其中[options]指定了启动的参数，[load-files]则指定了在启动是要执行的jshell脚本，脚本可以是任意的jshell代码片段，系统默认定了三个常用启动脚本:DEFAULT,JAVASE,PRINTING,在上述脚本中，我们指定了启动jshell时执行PRINTING脚本

DEFAULT脚本：导入一些常用的javase包,比如java.io,java.math,java.util等

JAVASE脚本：导入所有的javase包，此方式加载的包较多

PRINTING脚本：在DEFAULT的基础上，将print,println,printf定义为jshell方法，无需通过System.out.println()的方式使用

C.3 定义变量，方法和类型

可以在jshell中输入任意合法的java代码片段，包括变量定义，方法定义，类型定义，导入类型等等。语句后面的分号可以省略，解释器会自动在行尾添加分号

- 定义和使用变量

```
jshell> int a=3
a ==> 3

jshell> String str="hello jshell"
str ==> "hello jshell"

jshell> System.out.println(str)
hello jshell

jshell>
```

- 定义和调用方法

```
jshell> void printNowTime(){
   ...> Date now=new Date();
   ...> System.out.println(now);
   ...> }
|  created method printNowTime()
```

```
jshell> printNowTime()
Fri Sep 22 02:04:34 PDT 2017

jshell>
```

● 定义类型，实例化对象

由于类型定义的代码片段较长，我们用vim新建person.txt文本文件,定义Person类型如下：

```
public class Person{
	private int id;
	private String name;

	public Person(int id,String name)
	{
		this.id=id;
		this.name=name;
	}

	public String toString()
	{
		return "id:"+this.id+",name:"+this.name;
	}
}
```

用jshell命令，指定person.txt为启动脚本,加载其中定义的Person类型

```
[ldd@localhost learn]$ jshell person.txt
|  Welcome to JShell -- Version 9
|  For an introduction type: /help intro

jshell> /types
|   class Person

jshell> Person p=new Person(1,"zhangsan")
p ==> id:1,name:zhangsan

jshell> System.out.println(p)
id:1,name:zhangsan

jshell>
```

● 当前环境中的变量，方法，类型，导入查看

jshell提供了多个命令来查看当前jshell环境中的数据,/vars查看变量定义,/methods查看方法定义,/typs查看类型定义,/import查看导入的包

```
jshell> int a=3
a ==> 3

jshell> /vars
|   int a = 3

jshell> void sayHello(){
   ...> println("hello");
```

```
   ...> }
|  created method sayHello()

jshell> /methods
|   void print(boolean) #以PRINTING作为启动脚本导入一系列的打印方法
|   ......
|   void println(Object)
|   void printf(java.util.Locale,String,Object...)
|   void printf(String,Object...)
|   void sayHello() #我们定义的方法

jshell> public class Student{
   ...> }
|  created class Student

jshell> /types
|   class Student

jshell> /import
|   import java.io.*
|   import java.math.*
|   import java.net.*
|   import java.nio.file.*
|   import java.util.*
|   import java.util.concurrent.*
|   import java.util.function.*
|   import java.util.prefs.*
|   import java.util.regex.*
|   import java.util.stream.*

jshell>
```

C.4 删除定义的变量，方法和类型

jshell中使用/drop [name[name...]|id[id...]]语法，删除已经定义的变量类型和方法。多个名称之间用空格分隔

```
jshell> /drop Student
|  dropped class Student

jshell>
```

D. 总结

总的来说，jshell为我们快速验证原型，使用java进行简单的url访问，或者写个小工具处理一些重复的任务还是有很多帮助的。至于以后其能否作为脚本语言，发挥更大的用处，就不得而知了。

1234