每天一点

说说javap命令

javap定义

javap是 Java class文件分解器，可以反编译（即对javac编译的文件进行反编译），也可以查看java编译器生成的字节码。用于分解class文件。

测试类

```
public class JavapTest {

    private static final int _P_1 = 1;
    public static final int _P_2 = 2;

    public static void main(String[] args) {
        int m = 0, n = 0;
        for (int i = 0; i < 10; i++) {
            m = m++;
            n = ++n;
        }
        System.out.println("m = " + m);
        System.out.println("n = " + n);
    }
}
```

javap命令参数

```
C:\>javap -help
用法: javap <options> <classes>
其中, 可能的选项包括:
  -help  --help  -?        输出此用法消息
  -version                 版本信息
  -v  -verbose             输出附加信息
  -l                       输出行号和本地变量表
  -public                  仅显示公共类和成员
  -protected               显示受保护的/公共类和成员
  -package                 显示程序包/受保护的/公共类
                           和成员 (默认)
  -p  -private             显示所有类和成员
  -c                       对代码进行反汇编
  -s                       输出内部类型签名
  -sysinfo                 显示正在处理的类的
                           系统信息 (路径, 大小, 日期, MD5 散列)
  -constants               显示静态最终常量
  -classpath <path>        指定查找用户类文件的位置
  -bootclasspath <path>    覆盖引导类文件的位置
```

javap -version

```
C:\>javap -version
1.7.0_71
```

显示java版本

javap -p

```
C:\Users\user\Desktop>javap -p JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  private static final int _P_1;
  public static final int _P_2;
  public com.method.handler.JavapTest();
  public static void main(java.lang.String[]);
  private void say();
}
```

昵称：每天一点
园龄：1年11个月
粉丝：2
关注：1
+加关注

< 2018年4月 >

日	一	二	三	四	五	六
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

centos7 docker(1)
dhtmlxgrid 使用(1)
docker 命令(1)
JAVA(1)
spring AbstractRoutingDataSource(1)
spring boot docker(1)
spring boot java -jar windows乱码(1)
springmvc @RequestBody JSON.stringify()(1)
动态(1)
添加属性(1)
更多

随笔分类

每天一点(24)

随笔档案

2017年7月 (1)
2017年5月 (1)
2017年4月 (3)
2016年12月 (2)
2016年5月 (18)
2016年4月 (4)

最新评论

1. Re:JAVA"动态"为类添加属性
@no_hehe最近用空余时间写了个spring boot starter，实现可以动态配置添加属性功能，用的就是这个东东，可以瞄瞄~~~...
--每天一点
2. Re:JAVA"动态"为类添加属性
@每天一点好的 谢谢楼主 邮件看到了...
--no_hehe
3. Re:JAVA"动态"为类添加属性

显示类所有可访问修饰符范围》private的成员

javap -public

```
C:\Users\user\Desktop>javap -public JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  public static final int _P_2;
  public com.method.handler.JavapTest();
  public static void main(java.lang.String[]);
}
```

显示类的public成员

javap -protected

```
C:\Users\user\Desktop>javap -protected JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  public static final int _P_2;
  public com.method.handler.JavapTest();
  public static void main(java.lang.String[]);
}
```

显示类所有可访问修饰符范围》protected的成员

javap -l

```
C:\Users\user\Desktop>javap -p -l JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  private static final int _P_1;

  public static final int _P_2;

  public com.method.handler.JavapTest();
    LineNumberTable:
      line 6: 0

  public static void main(java.lang.String[]);
    LineNumberTable:
      line 12: 0
      line 13: 4
      line 14: 12
      line 15: 17
      line 13: 22
      line 17: 28
      line 18: 53
      line 19: 78

  private void say();
    LineNumberTable:
      line 23: 0
      line 24: 8
}
```

输出行号和本地变量表？看的不是很明白

javap -package

```
C:\Users\user\Desktop>javap -package JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  public static final int _P_2;
  public com.method.handler.JavapTest();
  public static void main(java.lang.String[]);
}
```

与javap -public作用类似

javap -v/-p -v

```
C:\Users\user\Desktop>javap -p -v JavapTest.class
Classfile /C:/Users/frinder_liu/Desktop/JavapTest.class
  Last modified 2016-4-27; size 911 bytes
  MD5 checksum e903be7495f5c462d6459a792e063628
  Compiled from "JavapTest.java"
public class com.method.handler.JavapTest
```

@no_hehe额，"->"是java7开始有的，目前用的是java8，支持的更好。其实就是所谓的：java lambda 表达式呗！代码：...

--每天一点

4. Re:JAVA“动态”为类添加属性

楼主，为什么我运行不了你写的程序，“->"是java的什么运算符？能否给我发一段完全代码 谢谢！

--no_hehe

```
  SourceFile: "JavapTest.java"
  minor version: 0
  major version: 51
  flags: ACC_PUBLIC, ACC_SUPER
Constant pool:
   #1 = Methodref          #13.#30        //  java/lang/Object."<init>":()V
   #2 = Fieldref           #31.#32        //  java/lang/System.out:Ljava/io/PrintStream;
   #3 = Class              #33            //  java/lang/StringBuilder
   #4 = Methodref          #3.#30         //  java/lang/StringBuilder."<init>":()V
   #5 = String             #34            //  m =
   #6 = Methodref          #3.#35         //  java/lang/StringBuilder.append:
(Ljava/lang/String;)Ljava/lang/StringBuilde
r;
   #7 = Methodref          #3.#36         //  java/lang/StringBuilder.append:
(I)Ljava/lang/StringBuilder;
   #8 = Methodref          #3.#37         //  java/lang/StringBuilder.toString:()Ljava/lang/String;
   #9 = Methodref          #38.#39        //  java/io/PrintStream.println:(Ljava/lang/String;)V
  #10 = String             #40            //  n =
  #11 = String             #41            //  hello world...
  #12 = Class              #42            //  com/method/handler/JavapTest
  #13 = Class              #43            //  java/lang/Object
  #14 = Utf8               _P_1
  #15 = Utf8               I
  #16 = Utf8               ConstantValue
  #17 = Integer            1
  #18 = Utf8               _P_2
  #19 = Integer            2
  #20 = Utf8               <init>
  #21 = Utf8               ()V
  #22 = Utf8               Code
  #23 = Utf8               LineNumberTable
  #24 = Utf8               main
  #25 = Utf8               ([Ljava/lang/String;)V
  #26 = Utf8               StackMapTable
  #27 = Utf8               say
  #28 = Utf8               SourceFile
  #29 = Utf8               JavapTest.java
  #30 = NameAndType        #20:#21        //  "<init>":()V
  #31 = Class              #44            //  java/lang/System
  #32 = NameAndType        #45:#46        //  out:Ljava/io/PrintStream;
  #33 = Utf8               java/lang/StringBuilder
  #34 = Utf8               m =
  #35 = NameAndType        #47:#48        //  append:(Ljava/lang/String;)Ljava/lang/StringBuilder;
  #36 = NameAndType        #47:#49        //  append:(I)Ljava/lang/StringBuilder;
  #37 = NameAndType        #50:#51        //  toString:()Ljava/lang/String;
  #38 = Class              #52            //  java/io/PrintStream
  #39 = NameAndType        #53:#54        //  println:(Ljava/lang/String;)V
  #40 = Utf8               n =
  #41 = Utf8               hello world...
  #42 = Utf8               com/method/handler/JavapTest
  #43 = Utf8               java/lang/Object
  #44 = Utf8               java/lang/System
  #45 = Utf8               out
  #46 = Utf8               Ljava/io/PrintStream;
  #47 = Utf8               append
  #48 = Utf8               (Ljava/lang/String;)Ljava/lang/StringBuilder;
  #49 = Utf8               (I)Ljava/lang/StringBuilder;
  #50 = Utf8               toString
  #51 = Utf8               ()Ljava/lang/String;
  #52 = Utf8               java/io/PrintStream
  #53 = Utf8               println
  #54 = Utf8               (Ljava/lang/String;)V
{
  private static final int _P_1;
    flags: ACC_PRIVATE, ACC_STATIC, ACC_FINAL
    ConstantValue: int 1

  public static final int _P_2;
    flags: ACC_PUBLIC, ACC_STATIC, ACC_FINAL
    ConstantValue: int 2

  public com.method.handler.JavapTest();
    flags: ACC_PUBLIC
    Code:
      stack=1, locals=1, args_size=1
         0: aload_0
         1: invokespecial #1                  // Method java/lang/Object."<init>":()V
         4: return
      LineNumberTable:
        line 6: 0
```

```
public static void main(java.lang.String[]);
  flags: ACC_PUBLIC, ACC_STATIC
  Code:
    stack=3, locals=4, args_size=1
       0: iconst_0
       1: istore_1
       2: iconst_0
       3: istore_2
       4: iconst_0
       5: istore_3
       6: iload_3
       7: bipush        10
       9: if_icmpge     28
      12: iload_1
      13: iinc          1, 1
      16: istore_1
      17: iinc          2, 1
      20: iload_2
      21: istore_2
      22: iinc          3, 1
      25: goto          6
      28: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
      31: new           #3                  // class java/lang/StringBuilder
      34: dup
      35: invokespecial #4                  // Method java/lang/StringBuilder."<init>":()V
      38: ldc           #5                  // String m =
      40: invokevirtual #6                  // Method java/lang/StringBuilder.append:
(Ljava/lang/String;)Ljava/lang/St
ringBuilder;
       43: iload_1
       44: invokevirtual #7                 // Method java/lang/StringBuilder.append:
(I)Ljava/lang/StringBuilder;
       47: invokevirtual #8                 // Method java/lang/StringBuilder.toString:
()Ljava/lang/String;
       50: invokevirtual #9                 // Method java/io/PrintStream.println:
(Ljava/lang/String;)V
       53: getstatic     #2                 // Field java/lang/System.out:Ljava/io/PrintStream;
       56: new           #3                 // class java/lang/StringBuilder
       59: dup
       60: invokespecial #4                 // Method java/lang/StringBuilder."<init>":()V
       63: ldc           #10                // String n =
       65: invokevirtual #6                 // Method java/lang/StringBuilder.append:
(Ljava/lang/String;)Ljava/lang/St
ringBuilder;
       68: iload_2
       69: invokevirtual #7                 // Method java/lang/StringBuilder.append:
(I)Ljava/lang/StringBuilder;
       72: invokevirtual #8                 // Method java/lang/StringBuilder.toString:
()Ljava/lang/String;
       75: invokevirtual #9                 // Method java/io/PrintStream.println:
(Ljava/lang/String;)V
       78: return
     LineNumberTable:
       line 12: 0
       line 13: 4
       line 14: 12
       line 15: 17
       line 13: 22
       line 17: 28
       line 18: 53
       line 19: 78
     StackMapTable: number_of_entries = 2
          frame_type = 254 /* append */
            offset_delta = 6
       locals = [ int, int, int ]
          frame_type = 250 /* chop */
         offset_delta = 21


  private void say();
    flags: ACC_PRIVATE
    Code:
      stack=2, locals=1, args_size=1
         0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
         3: ldc           #11                 // String hello world...
         5: invokevirtual #9                  // Method java/io/PrintStream.println:
(Ljava/lang/String;)V
         8: return
      LineNumberTable:
        line 23: 0
        line 24: 8
```

```
}
```

命令说明是：输出附加信息
class文件的路径、最后修改时间、文件大小等
类的全路径、源（java）文件等
常量池
常量定义、值
构造方法
程序调用及执行逻辑（这个涉及的内容就比较多了）

总之，javap -v命令是很强大的一个命令！

javap -c

```
C:\Users\user\Desktop>javap -c JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  public static final int _P_2;

  public com.method.handler.JavapTest();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: iconst_0
       1: istore_1
       2: iconst_0
       3: istore_2
       4: iconst_0
       5: istore_3
       6: iload_3
       7: bipush        10
       9: if_icmpge     28
      12: iload_1
      13: iinc          1, 1
      16: istore_1
      17: iinc          2, 1
      20: iload_2
      21: istore_2
      22: iinc          3, 1
      25: goto          6
      28: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
      31: new           #3                  // class java/lang/StringBuilder
      34: dup
      35: invokespecial #4                  // Method java/lang/StringBuilder."<init>":()V
      38: ldc           #5                  // String m =
      40: invokevirtual #6                  // Method java/lang/StringBuilder.append:
(Ljava/lang/String;)Ljava/lang/Stri
ngBuilder;
      43: iload_1
      44: invokevirtual #7                  // Method java/lang/StringBuilder.append:
(I)Ljava/lang/StringBuilder;
      47: invokevirtual #8                  // Method java/lang/StringBuilder.toString:
()Ljava/lang/String;
      50: invokevirtual #9                  // Method java/io/PrintStream.println:
(Ljava/lang/String;)V
      53: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
      56: new           #3                  // class java/lang/StringBuilder
      59: dup
      60: invokespecial #4                  // Method java/lang/StringBuilder."<init>":()V
      63: ldc           #10                 // String n =
      65: invokevirtual #6                  // Method java/lang/StringBuilder.append:
(Ljava/lang/String;)Ljava/lang/Stri
ngBuilder;
      68: iload_2
      69: invokevirtual #7                  // Method java/lang/StringBuilder.append:
(I)Ljava/lang/StringBuilder;
      72: invokevirtual #8                  // Method java/lang/StringBuilder.toString:
()Ljava/lang/String;
      75: invokevirtual #9                  // Method java/io/PrintStream.println:
(Ljava/lang/String;)V
      78: return
}
```

其实，javap -c 输出内部javap -v中已经有了，我们详细介绍下javap -c命令的输出内容
0: iconst_0 前面0:表示执行的顺序，iconst_0把值 0 放入栈顶，_0中的0代表压栈的值，如：iconst_5，即把5压入栈顶
1: istore_1 将栈顶的值放入变量1 中，_1代表变量的序列，本例中为：m，如：istore_2即为变量n赋值

```
6: iload_3 将变量3 即 i 的的值放入栈顶，与iconst不同的是，iload操作的值是已经定义好存在的值，iconst是定义时的压栈操作
13: iinc          1, 1  将变量1 的值加1
```

文章开头的demo中最终的结果是什么呢？

```
m = 0
n = 10
```

为什么呢？
我们来关注下这几行：

```
12: iload_1             -- 将变量1 值放入栈项
13: iinc          1, 1  -- 将变量1 增加1 变为2，栈顶值仍然是1。其中第一个1 表示变量，第二个1 表示增量
16: istore_1            -- 将栈顶值赋值给变量1，变量1 仍然为1
17: iinc          2, 1  -- 将变量2 增加1 变为2，栈顶值仍然是1。其中第一个2 表示变量，第二个1 表示增量
20: iload_2             -- 将变量2 值放入栈项
21: istore_2            -- 将栈顶值赋值给变量2，变量2 值为2
```

记住一个点：

```
int m = 0, n = 0;
m = m++;  -- 会先将m值（即0）赋值给m后再++
n = ++n;  -- n先++后再把值赋值给n
此时，m = 0, n = 1;
```

这个地方有点绕，需要多看几次，理顺逻辑与关系！

这里可以参考下http://blog.csdn.net/junsure2012/article/details/7099222，讲的很详细也易懂！

javap -s/-p -s

```
C:\Users\user\Desktop>javap -p -s JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  private static final int _P_1;
    Signature: I
  public static final int _P_2;
    Signature: I
  public com.method.handler.JavapTest();
    Signature: ()V

  public static void main(java.lang.String[]);
    Signature: ([Ljava/lang/String;)V

  private void say();
    Signature: ()V
}
```

输出内部类型签名

javap -sysinfo/-p -sysinfo

```
C:\Users\user\Desktop>javap -sysinfo JavapTest.class
Classfile /C:/Users/user/Desktop/JavapTest.class
  Last modified 2016-4-27; size 911 bytes
  MD5 checksum 3f6dfcf7121785760b234224c5d135fd
  Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  public static final int _P_2;
  public com.method.handler.JavapTest();
  public static void main(java.lang.String[]);
}
```

显示正在处理的类的
系统信息 (路径, 大小, 日期, MD5 散列)

javap -constants/-p -constants

```
C:\Users\user\Desktop>javap -constants JavapTest.class
Compiled from "JavapTest.java"
public class com.method.handler.JavapTest {
  public static final int _P_2 = 2;
  public com.method.handler.JavapTest();
  public static void main(java.lang.String[]);
}
```

显示静态最终常量

javap -classpath/-bootclasspath

先跳过

分类: 每天一点

每天一点
关注 - 1
粉丝 - 2
+加关注

« 上一篇：java中的类加载器
» 下一篇：说说JVM中的操作码

posted @ 2016-04-27 20:02 每天一点 阅读(4859) 评论(0) 编辑 收藏