Manipulação de Bytecode Java

Mantendo o espírito Hacker no mundo das linguagens de alto nível

André Luiz Breves de Oliveira
andre.breves@gmail.com

```
class Hello {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

```
class Hello extends java.lang.Object {
Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hello World
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
   8:   return
}
```

```
cafebabe0017890abf9069a0f696a00a9b6f60a96f0a676debac758a78db876e66897f7878a788
32190db5891eba0c70a0b06efb08a0860a8bf0ba0b680030739abca2938092189ab308b8239b
```

Conhecendo as Regras: A Plataforma Java

- Arquivo .class
- Bytecode
- Máquina Virtual
- Linguagem Java
- API

Arquivo .class

Header	Magic Number: **0xcafebabe**
	Version Numbers
Constant Pool	Constant Pool Count
	[*] Constant Content
Class Attributes	Access Flags
	This and Super classes
	[*] Interface
[*] Fields	Field Name, Descriptor, etc
	[*] Field Attributes
[*] Methods	Method Name, Descriptor, etc
	Method max stack and locals
	[*] Method Code Table
	[*] Method Exception Table
	[*] Method Code Attributes
	[*] Method Attributes

Bytecode

- Load e Store
- Aritmética
- Conversão de Tipos
- Criação e Manipulação de Objetos
- Gerenciamento da Pilha de Operação
- Transferência do Controle (branching)
- Invocação e Retorno de Métodos
- Lançar Exceções

JVM - Java Virtual Machine

- Poucos registros, maior uso de pilha
- Carrega, verifica e executa o bytecode

Method Area	Heap	Stack	PC Register
•RCP •Field data •Method data •Code	•Compartilhado	•Um por thread •Stack Frame: •Local Variables •Operand Stack •Reference to RCP	•Um por thread

Quebrando as Regras: Manipulação de Bytecode

- AKA Bytecode Engineering
- Estática: arquivo .class
- Dinâmica: usando ClassLoader
- Limitada: JVM não permite reload de classe

Alteração de arquivo .class

- Atributos da Classe
- Variáveis da Classe
- Assinatura dos métodos
- Atributos dos métodos
- Bytecode dos métodos

Usando ClassLoader

- Extende o ClassLoader abstrato
- Altera ou cria um arquivo .class na memória
- Chama o método final defineClass()

Uso Comum

Extensão da Linguagem Java:

- Programação Orientada a Aspectos
- Reflexão Computacional

Programação Orientada a Aspectos

- Captura dos métodos
- Aplicação dos advices nos join points

Reflexão Computacional

- java.lang.reflection: apenas inspeção
- Manipulação de Bytecode: alteração de classes

Frameworks e Ferramentas

- ASM Java Bytecode Manipulation Framework
- BCEL - Byte Code Engineering Library
- Javassist - Java Programming Assistant

ASM

- Projeto da ObjectWeb
- Leve (33 Kb)
- Melhor performance
- Mais difícil de usar
- Baseado no padrão Visitor
- Dividido em bytecode producers e consumers

Exemplo de Código

```
01  ClassWriter cw = new ClassWriter(computeMax);
02  ClassVisitor cc = new CheckClassAdapter(cw);
03  //TraceClassVisitor prints the transformed class and delegates to cc
04  ClassVisitor tv =  new TraceClassVisitor(cc, new PrintWriter(System.out));
05  //TransformingClassAdapter implements custom class transformations
06  ClassVisitor cv = new TransformingClassAdapter(tv);
07  ClassReader cr = new ClassReader(bytecode);
08  cr.accept(cv, skipDebug);
09
10  //Gets the generated bytecode
11  byte[] newBytecode = cw.toByteArray();
```

BCEL

- Projeto do Jakarta
- Primeiro a ter grande adoção
- Acompanha bytecode verifier JustIce
- Pode usar o padrão Visitor
- Manipulação usando mnemonics

Exemplo de Código

```
01 ClassGen cg = new ClassGen("HelloWorld", "java.lang.Object",
02                            "<generated>", ACC_PUBLIC | ACC_SUPER,
03                            null);
04  ConstantPoolGen cp = cg.getConstantPool(); // cg creates constant pool
05  InstructionList il = new InstructionList();
06  MethodGen  mg = new MethodGen(ACC_STATIC | ACC_PUBLIC, // access flags
07                                Type.VOID,               // return type
08                                new Type[] {             // argument types
09                                  new ArrayType(Type.STRING, 1) },
10                                new String[] { "argv" }, // arg names
11                                "main", "HelloWorld",    // method, class
12                                il, cp);
13  InstructionFactory factory = new InstructionFactory(cg);
14  il.append(factory.createNew(Type.STRINGBUFFER));
15  il.append(InstructionConstants.DUP);
16  il.append(new PUSH(cp, "Hello, "));
17  il.append(factory.createInvoke("java.lang.StringBuffer", "<init>",
18                                 Type.VOID, new Type[] { Type.STRING },
19                                 Constants.INVOKESPECIAL));
20  il.append(new ALOAD(name));
21  mg.setMaxStack();
22  cg.addMethod(mg.getMethod());
```

Javassist

- Projeto do JBoss
- Mais fácil de usar
- Semelhante ao java.lang.reflection
- Gera bytecode de código Java em uma String
- Dispositivos para AOP e Reflection
- Dividido em source level e bytecode level

Exemplo de Código

```
01  //ClassPool controls bytecode modification
02  ClassPool pool = ClassPool.getDefault();
03  //CtClass: CompileTimeClass
04  CtClass cc = pool.get("test.Rectangle");
05  //Alters the Super class
06  cc.setSuperclass(pool.get("test.Point"));
07  //Creats a new method
08  CtMethod m = CtNewMethod.make("public int xmove(int dx) { x += dx; }", cc);
09  cc.addMethod(m);
10  cc.writeFile();
11  byte[] b = cc.toBytecode();
12  Class clazz = cc.toClass();
13
14  //Defining a new class
15  ClassPool pool = ClassPool.getDefault();
16  CtClass cc = pool.makeClass("Point");
```

Referências

- Decompiling Java – Godfrey Nolan
- The Java Virtual Machine Specification
  http://java.sun.com/docs/books/vmspec/
- ASM Java Bytecode Manipulation Framework
  http://asm.objectweb.org/
- BCEL Byte Code Engineering Library
  http://jakarta.apache.org/bcel/
- Javassist Java Programming Assistant
  http://www.csg.is.titech.ac.jp/~chiba/javassist/

Contato

André Luiz Breves de Oliveira
andre.breves@gmail.com

Obrigado!