BCI - The Power of Bytecode Instrumentation in Java

BCI(Bytecode Instrumenation)

- [The Power of Bytecode Instrumentation in Java - Part 1](#)
- [The Power of Bytecode Instrumentation in Java - Part 2](#)
- [The Power of Bytecode Instrumentation in Java - Part 3](#)
- [The Power of Bytecode Instrumentation in Java - Part 4](#)
- [The Power of Bytecode Instrumentation in Java - Part 5](#)
- [The Power of Bytecode Instrumentation in Java - Part 6](#)

: [http://ukja.tistory.com(](#)   ,   )

The Power of Byte Code Instrumentation in Java - Part1

Byte Code Instrumentation   ?

Java   ?

                       BCI,   Byte Code Instrumentation   . (Byte Code Insertion) Byte Code Instrumenation   Java   Byte Code   ,
                           .

                          Java   BCI   , BCI
                                                                                                              .
Bytecode   BCI
                   .

                           [AOP(Aspected Oriented Programming) In Java]   , AOP
                                 BCI   . AOP   ,
        Aspect   Business Logic   Weaving   BCI,   Java
                                              .

                       AOP   BCI
                .

             Jennifer, Performizer   WAS   ,
                                                                              BCI   .
                          BCI   .

Java Bytecode

Java   Bytecode   (   )   .

                                OS/   Java Bytecode   JVM(Java Virtual Machine)

JVM   OS/
Bytecode   . Java   OS
Bytecode   .

                Java Bytecode   .

```
public getValue()I
L0 (0)
LINENUMBER 28 L0
SIPUSH 1000
ISTORE 1
L1 (3)
LINENUMBER 29 L1
ILOAD 1
IRETURN
L2 (6)
LOCALVARIABLE this Lflowlite/io/ASMTest; L0 L2 0
LOCALVARIABLE value I L1 L2 1
MAXSTACK = 1
MAXLOCALS = 2
```

                Bytecode   Java Source   .

```
public int getValue() {
    int value = 1000;
    return value;
}
```

     Sun   JVM   Bytecode   ,
      Bytecode   .(   )

http://java.sun.com/docs/books/jvms/   Java Virtual Machine   .
            Java Class File   Class File   Bytecode   .
                                                              . ^^)

     Bytecode   ,   .
  Bytecode
                                              .

BCI

                              Bytecode   .

  • ASM : Object Web   http://asm.objectweb.org/
  • BCEL : Apache   http://jakarta.apache.org/bcel/
  • SERP : Sourceforge   http://serp.sourceforge.net/

- Javassist : JBoss   http://www.csg.is.titech.ac.jp/~chiba/javassist/(
  …)

ASM   BCI
.

The Power of Byte Code Instrumenation in Java - Part2

Byte Code Instrumentation(BCI)   BCI
    ASM

ASM

ASM   http://asm.objectweb.org/   (Eclipse Plugin - AMS
Bytecode Outline Plugin)   . Eclipse Plugin   ASM
                        .   Eclipse ASM Bytecode Outline Plugin
              Bytecode   .
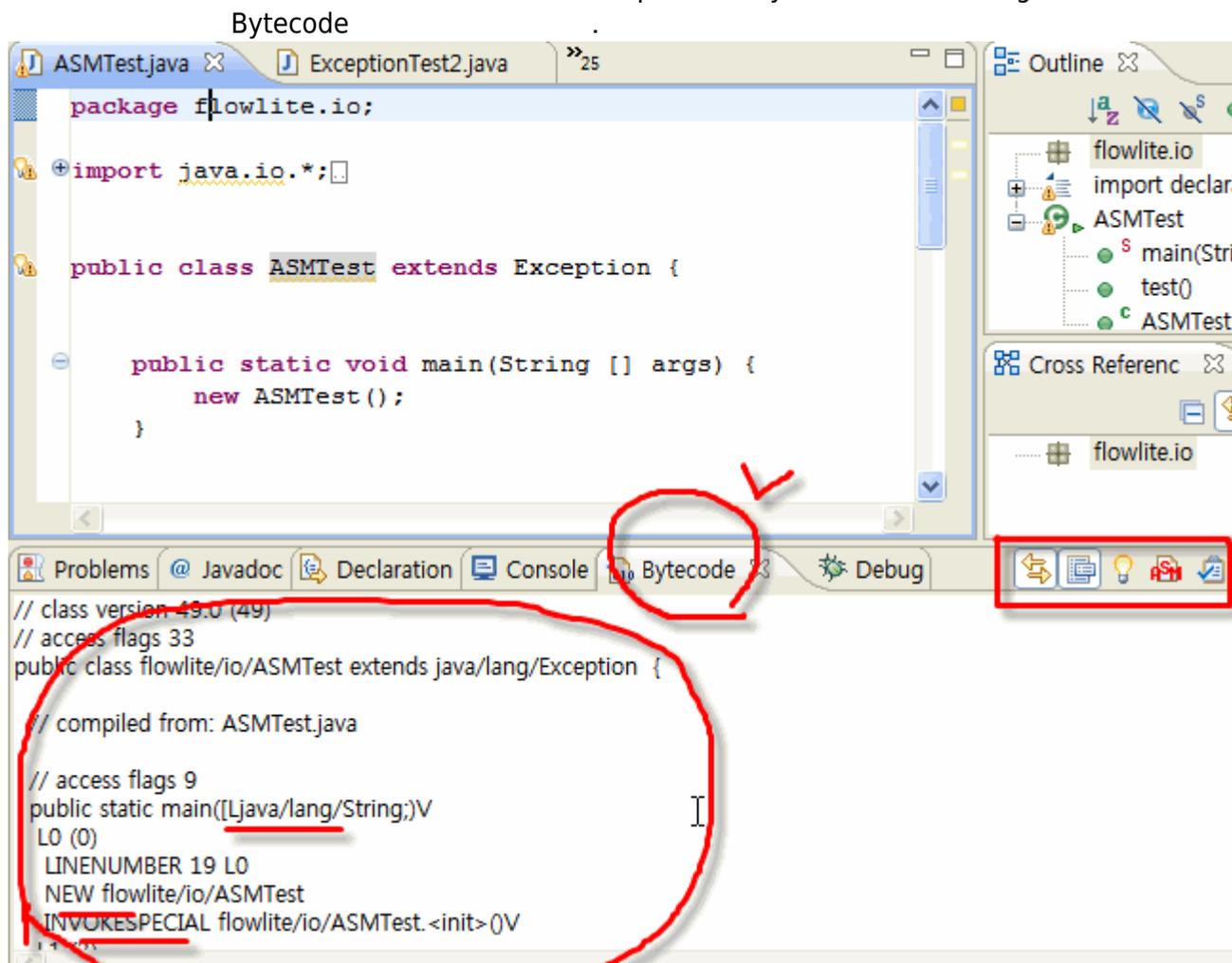
Bytecode   .
          Bytecode   . ASM Bytecode Outline Plugin   ASM
                              [ASM]
Bytecode   Java Code   .

```
package asm.flowlite.io;
import java.util.*;
import org.objectweb.asm.*;
import org.objectweb.asm.attrs.*;
public class ASMTestDump implements Opcodes {

public static byte[] dump () throws Exception {

ClassWriter cw = new ClassWriter(0);
FieldVisitor fv;
MethodVisitor mv;
AnnotationVisitor av0;

cw.visit(V1_5, ACC_PUBLIC + ACC_SUPER, "flowlite/io/ASMTest", null, "java/lang/Exception", null);

cw.visitSource("ASMTest.java", null);

{
mv = cw.visitMethod(ACC_PUBLIC + ACC_STATIC, "main", "([Ljava/lang/String;)V", null, null);
mv.visitCode();
Label l0 = new Label();
mv.visitLabel(l0);
mv.visitLineNumber(19, l0);
mv.visitTypeInsn(NEW, "flowlite/io/ASMTest");
mv.visitMethodInsn(INVOKESPECIAL, "flowlite/io/ASMTest", "<init>", "()V");
Label l1 = new Label();
mv.visitLabel(l1);
```

ASM   (ASMified Code)   ASM   API   Bytecode
            .   ASM   Copy/Paste/Edit   Bytecode
                       .   ?   .   Java
                                    .
                 .   Java Monitor/Profiler/Analyzer
             BCI   .   . ASM
                        http://asm.objectweb.org/   Java Doc   .
                         . ASM   Visitor Pattern
                   .

The Power of Byte Code Instrumentation in Java - Part 3

One Simple but Powerful Example of BCI

ASM   Byte Code Instrumentation   .

                       ,   .

                                         .

                  .

"   WAS Application   Exception
      .   Exception   Catch   …. "

("Aspected Oriented Programming in Java"   )

.

```
public static void main(String[] args) {

    try {
        ExceptionTest2.doException(true);
    } catch(Exception ex) {

        // Do something about exceptions...
        // ExceptionManager.recordException(ex);
    }
```

, Application   try catch   Exception   Catch
                       .   .

- 　　　　　　　　　　　　　　　　　?
- 　　　　　　　　　　　Java Core Library(java.*, javax.*, …)   3rd Party Library
     　?
- Exception   ?
- 　　　　　　　　　　　　　　　　　　　　　　?

Business Logic   Exception
                .

                         AOP   …

                                         ….

                .

- "java.lang.Excetpion"   Exception   ?"
- "   ,   Exception   java.lang.Exception
               "
- "   Java.lang.Exception   Exception   "

!!

     java.lang.Exception   .

          BCI,   Byte Code Instrumentation   .

     java.lang.Exception   Java Decompiler
       .

Decompiler   Exception   .

```
package java.lang;

// Referenced classes of package java.lang:
// Throwable, String

public class Exception extends Throwable {

    public Exception() {
```

```
    }

    public Exception(String s) {
        super(s);
    }

    public Exception(String s, Throwable throwable) {
        super(s, throwable);
    }

    public Exception(Throwable throwable) {
        super(throwable);
    }

    static final long serialVersionUID = 0xd0fd1f3e1a3b1cc4L;
}
```

ASM Bytecode Outline Plugin   Byte Code   .

```
/ class version 49.0 (49)
// access flags 33
public class Exception extends Throwable {


// access flags 24
final static long serialVersionUID = -3387516993124229948

// access flags 1
public () : void
ALOAD 0
INVOKESPECIAL Throwable.() : void
RETURN
MAXSTACK = 1
MAXLOCALS = 1

// access flags 1
public (String) : void
ALOAD 0
ALOAD 1
INVOKESPECIAL Throwable.(String) : void
RETURN
MAXSTACK = 2
MAXLOCALS = 2

// access flags 1
public (String,Throwable) : void
ALOAD 0
ALOAD 1
```

```
ALOAD 2
INVOKESPECIAL Throwable.(String,Throwable) : void
RETURN
MAXSTACK = 3
MAXLOCALS = 3

// access flags 1
public (Throwable) : void
ALOAD 0
ALOAD 1
INVOKESPECIAL Throwable.(Throwable) : void
RETURN
MAXSTACK = 2
MAXLOCALS = 2
}
```

Exception   ?

```
public class Exception extends Throwable {

    public Exception() {
        ExceptionCallBack.exceptionOccurred(this);
    }

    public Exception(String s) {
        super(s);
        ExceptionCallBack.exceptionOccurred(this);
    }

    public Exception(String s, Throwable throwable) {
        super(s, throwable);
        ExceptionCallBack.exceptionOccurred(this);
    }

    public Exception(Throwable throwable) {
        super(throwable);
        ExceptionCallBack.exceptionOccurred(this);
    }

    static final long serialVersionUID = 0xd0fd1f3e1a3b1cc4L;
}
```

             Exception   ExceptionCallBack.exceptionOccurred(this)
             .   Java Core Library   Exception
        .

Part4   ASM   java.lang.Exception
    .

. "java.lang.Exception"
java.lang.Exception   ?"

.   .

JVM   Version   Vendor   .

.

.   Java Core Library   (?)

.

The Power of Byte Code Instrumentation in Java - Part 4

One Simple but Powerful Example of BCI

Part3   java.lang.Exception
Exception   . Exception
.

```
public Exception(String s)
{
    super(s);
    ExceptionCallBack.exceptionOccurred(this);
}
```

ASM   Library   .

ASM
http://asm.objectweb.org   .

java.lang.Exception   ASM Library   (Read),   (   )
Write   .

```
package flowlite.exception2;

import java.io.FileOutputStream;

import org.objectweb.asm.*;
import org.objectweb.asm.commons.*;

public class ExceptionTransformer implements Opcodes {

    // Convert class
```

```java
    public void transform(String newClassName) throws Exception {

        System.out.println("Starting transformation of
java.lang.Exception...");

        // Reader
        ClassReader reader = new ClassReader("java.lang.Exception");
        // Writer

        ClassWriter writer = new ClassWriter(ClassWriter.COMPUTE_MAXS);

        // Class Adapter
        ClassAdapter adapter = new ExceptionClassAdapter(writer);

        reader.accept(adapter, ClassReader.SKIP_FRAMES);

        byte[] b = writer.toByteArray();
        FileOutputStream fos = new FileOutputStream(newClassName +
".class");
        fos.write(b);
        fos.flush();

    }

    public static void main(String[] args) {
        try {
            String newClassName = "Exception";
            if (args.length >= 1)
                newClassName = args[0];
            ExceptionTransformer fit = new ExceptionTransformer();
            fit.transform(newClassName);
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }
}

class ExceptionClassAdapter extends ClassAdapter implements Opcodes {

    public ExceptionClassAdapter(ClassVisitor visitor) {
        super(visitor);

    }

}
```

- ClassReader
- ClassAdapter

- ClassWriter   .
-   ClassLoader   .

                         Exception.class   .

.project	1KB	PROJECT 파일
.classpath	1KB	CLASSPATH
Exception	1KB	CLASS File
rt	33,142KB	알집 jar 파일
aspectjrt	113KB	알집 jar 파일
asm-commons-3.0	25KB	알집 jar 파일
asm-3.0	42KB	알집 jar 파일

-Xbootclasspath

                              JVM   rt.jar   Exception
Exception   . rt.jar   ,
(   )   .

       -Xbootclasspath   . "java -X"
             .

```
C:\Documents and Settings\mulder>java -X
    -Xmixed           mixed mode execution (default)
    -Xint             interpreted mode execution only
    -Xbootclasspath:<directories and zip/jar files separated by ;>
                      set search path for bootstrap classes and resources
    -Xbootclasspath/a:<directories and zip/jar files separated by ;>
                      append to end of bootstrap class path
    -Xbootclasspath/p:<directories and zip/jar files separated by ;>
```

  , -Xbootclasspath/p:<   Exception Class   >   JVM   rt.jar
    (prepend)   Exception Class   .   rt.jar
                        .

                    Test Class

```
public class ExceptionTest2 {

    public static void doException(boolean bException) throws
RuntimeException {
        if (bException)
            throw new RuntimeException("test");

    }

    public static void main(String[] args) {

        try {
            ExceptionTest2.doException(true); // <--        Exception
```

```
        } catch (Exception ex) {
        }

        try {
            ExceptionTest2.doException(false);
        } catch (Exception ex) {
        }
    }
}
```

-Xbootclasspath . (./converted_classes/java/lang
Exception Class …)

```
java -Xbootclasspath/p:./converted_classes ExcetpionTest2
```

java.lang.Exception

java.lang.Exception .

```
ExceptionCallBack.exceptionOccurred(this);
```

. ASM Bytecode
Outline Plugin .

. …

- mv.visitVarInsn(ALOAD, 0) : 0   (this) .
- mv.visitMethodInsns(INVOKESTATIC, [className], [methodName], [methodDescription]) :
    .
    - INVOKESTATIC   Static Method   . methodDescription

.

- (Ljava/lang/Exception;)   java.lang.Exception

.

- V   void   .

java.lang.Exception

.   .

```java
package flowlite.exception2;

import java.io.FileOutputStream;

import org.objectweb.asm.*;
import org.objectweb.asm.commons.*;

public class ExceptionTransformer implements Opcodes {

    // Convert class
    public void transform(String newClassName) throws Exception {

        System.out.println("Starting transformation of
java.lang.Exception...");

        ClassReader reader = new ClassReader("java.lang.Exception");
        ClassWriter writer = new ClassWriter(ClassWriter.COMPUTE_MAXS);
        ClassAdapter adapter = new ExceptionClassAdapter(writer);

        reader.accept(adapter, ClassReader.SKIP_FRAMES);

        byte[] b = writer.toByteArray();
        FileOutputStream fos = new FileOutputStream(newClassName +
".class");
        fos.write(b);
        fos.flush();

    }

    public static void main(String[] args) {
        try {
            String newClassName = "Exception";
            if (args.length >= 1)
                newClassName = args[0];
            ExceptionTransformer fit = new ExceptionTransformer();
            fit.transform(newClassName);
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }
```

```
}

class ExceptionClassAdapter extends ClassAdapter implements Opcodes {

    public ExceptionClassAdapter(ClassVisitor visitor) {
        super(visitor);

    }

    public MethodVisitor visitMethod(int access, String name, String desc,
String sig, String[] exes) {
        MethodVisitor mv = super.visitMethod(access, name, desc, sig, exes);

        if (name.equals("")) { // Constructor
            System.out.println("Redefine Constructor...");
            ExceptionConstructorAdviceAdapter ecaa = new
ExceptionConstructorAdviceAdapter(access, name, desc, mv);
            return ecaa;
        }

        return mv;

    }
}

class ExceptionConstructorAdviceAdapter extends AdviceAdapter {

    public ExceptionConstructorAdviceAdapter(int access, String name, String
desc, MethodVisitor mv) {
        super(mv, access, name, desc);
    }

    protected void onMethodEnter() {

    }

    protected void onMethodExit(int opcode) {
        if (opcode == RETURN) {
            mv.visitVarInsn(ALOAD, 0);
            mv.visitMethodInsn(INVOKESTATIC,
"flowlite/exception2/ExceptionCallBack", "exceptionOccurred",
"(Ljava/lang/Exception;)V");
            mv.visitEnd();
        }
    }

}
```

.

- ClassReader   rt.jar   java.lang.Exception   .

- ExceptionClassAdapter(extends ClassAdapter)

  - ExceptionClassAdpater   visitMethod   Exception

  - ClassAdapter.visitMethod
            visitMethod

  - (Constructor)   .
- ExceptionClassAdapter.visitMethod   ()
  ExceptionConstructorAdviceAdapter(extends AdviceAdapter)

  - AdviceAdpater   onMethodEnter, onMethodExit(opcode)   .
  - onMethodEnter   , onMethodExit

  -    , ExceptionConstructorAdviceAdapter
       /   .

              ClassAdapter   AdviceAdpater

            onMethodExit
. ExceptionConstructorAdviceAdapter.onMethodExit   .

```
protected void onMethodExit(int opcode) {
    if(opcode == RETURN) {
            mv.visitVarInsn(ALOAD, 0);
            mv.visitMethodInsn(INVOKESTATIC,
"flowlite/exception2/ExceptionCallBack", "exceptionOccurred",
"(Ljava/lang/Exception;)V");
            mv.visitEnd();
    }
}
```

                               (opcode == RETURN)
"ExceptionCallBack.exceptionOccurred(this)"   .

ExceptionTransformer   Exception Class
                    .

```
// class version 49.0 (49)
// access flags 33
public class Exception extends Throwable {

// access flags 24
final static long serialVersionUID = -3387516993124229948

// access flags 1
public () : void
ALOAD 0
```

```
INVOKESPECIAL Throwable.() : void
ALOAD 0
INVOKESTATIC ExceptionCallBack.exceptionOccurred(Exception) : void
RETURN
MAXSTACK = 1
MAXLOCALS = 1

// access flags 1
public (String) : void
ALOAD 0
ALOAD 1
INVOKESPECIAL Throwable.(String) : void
ALOAD 0
INVOKESTATIC ExceptionCallBack.exceptionOccurred(Exception) : void
RETURN
MAXSTACK = 2
MAXLOCALS = 2

// access flags 1
public (String,Throwable) : void
ALOAD 0
ALOAD 1
ALOAD 2
INVOKESPECIAL Throwable.(String,Throwable) : void
ALOAD 0
INVOKESTATIC ExceptionCallBack.exceptionOccurred(Exception) : void
RETURN
MAXSTACK = 3
MAXLOCALS = 3

// access flags 1
public (Throwable) : void
ALOAD 0
ALOAD 1
INVOKESPECIAL Throwable.(Throwable) : void
ALOAD 0
INVOKESTATIC ExceptionCallBack.exceptionOccurred(Exception) : void
RETURN
MAXSTACK = 2
MAXLOCALS = 2
}
```

ExceptionCallBack.exceptionOccurred   .

```
package flowlite.exception2;

public class ExceptionCallBack {

    public static void exceptionOccurred(Exception ex) {
```

```
        System.out.println("Oops~ " + ex + " has occurred...");
        ex.printStackTrace();
    }
}
```

-Xbootclasspath   Exception
ExceptionTest2   . (Exception.class   converted_classes/java/lang
                    !!!)

```
java -Xbootclasspath/p:./converted_classes ExcetpionTest2
```

Exception   ExceptionCallBack.exceptionOccurred
                    .

                    ?

                                                        Exception   .

                    ASM   BCI   ,
                    .

The Power of Byte Code Instrumentation in Java - Part 5

Part4   Byte Code Instrumentation(BCI)   ,
              BCI   .

                    .

> "   Java   Socket
>   . ,   Socket
> I/O   ."

BCI   OS   Network Monitoring Tool   API
      Java Application   Port   .   Tool   API
                                                        .

BCI 　　　　　　　　　　　　　　　　　　　　　　　．　　　　　　　　　　　　　　　？

java.net..Socket 　　　　　　　　　？

Socket Class 　　　　　　　　　　 API 　　　．

```
public InputStream getInputStream()throws IOException
public OutputStream getOutputStream()throws IOException
```

　　　　　　　 Socket 　　　　　　　　　　 InputStream　 OutputStream 　　　　．
　　　　　　　 InputStream 　　　　　　　　． InputStream Class 　　　　　　　　 API
　　　　．

```
public int read(byte[]b,intoff,intlen)throws IOException
```

　　　　　　 InputStream 　　　　　　　　 (b[]) 　
return 　　　　　　．　　　　　　　　　　　　　　　　　　．

BCI 　　　　　InputStream.read 　　　　　　　　　　　　　　　　，
　　　　　　　　．

- 　　 Socket.getInputStream() 　　　　　　 __orig$getInputStream__ 　　　．
- Socket.getInputStream() 　　　　　　　　　　．

```
public InputStream getInputStream() {
  //       getInputStream(     __orig$getInputStream__       )
InputStream       .
    InputStream is = __orig$getInputStream__();
  //   InputStream             InputStream       .
    FlowLiteSocketInputStream fsis = new FlowLiteSocketInputStream(this,
is);

    return fsis;
}
```

　　　　　　 Socket 　　　　　 InputStream 　　　　　　　　　　　　　　． JDK
　 Socket　 InputStream 　　　　　　　　 InputStream, 　 FlowLiteSocketInputStream
InputStream 　　　　　　，　　　　　　　　　　　　　　　　．

```
package java.net;

public class FlowLiteSocketInputStream extends InputStream {

    Socket s = null;
    InputStream is = null;
```

```java
    public FlowLiteSocketInputStream(Socket s, InputStream is) {
        this.s = s;
        this.is = is;
        SocketIOCallBack.createCalled(this);
    }

    public int read() throws IOException {
        int len = is.read();
        SocketIOCallBack.readCalled(this, 4);

        return len;

    }

    public int read(byte[] b) throws IOException {
        int len = is.read(b);
        SocketIOCallBack.readCalled(this, len);
        return len;
    }

    public int read(byte[] b, int off, int len) throws IOException {
        int len2 = is.read(b, off, len);
        SocketIOCallBack.readCalled(this, len2);
        return len2;
    }

    public int available() throws IOException {
        return is.available();
    }

    public void close() throws IOException {
        is.close();
    }

    public void mark(int readlimit) {
        is.mark(readlimit);
    }

    public boolean markSupported() {
        return is.markSupported();
    }

    public Socket getSocket() {
        return this.s;
    }
}
```

SocketIOCallBack   FlowLiteInputStream
                     .

ASM　　　　　java.net.Socket

ASM　　　　　　　　　　　　　　　　　　　　　　　java.net.Socket
　　　　　　　　　　　　.　　　　　　　　　.

```
package flowlite.net;

import java.io.*;
import org.objectweb.asm.*;
import org.objectweb.asm.commons.*;

/**
 * Net io class transformer. Execute byte code transformation to convert
 * NetInputStream class You must convert native java.io.Socket class using
this
 * class
 *
 * The technique is the most powerful. So keep in mind~~
 *
 * Must use ASM 3.0 library
 *
 * @history 2007/07/17Dongwook ChoInitial Coding
 *
 */
public class SocketTransformer implements Opcodes {

    public SocketTransformer() {

    }

    // Convert class
    public void transform(String newClassName) throws Exception {
        System.out.println("Starting transformation of java.net.Socket...");
        // Prepared reader, writer, adapter

        ClassReader reader = new ClassReader("java.net.Socket");
        ClassWriter writer = new ClassWriter(ClassWriter.COMPUTE_MAXS +
ClassWriter.COMPUTE_FRAMES);
        ClassAdapter adapter = new SocketClassAdapter(writer);

        reader.accept(adapter, ClassReader.SKIP_DEBUG);

        byte[] b = writer.toByteArray();
        FileOutputStream fos = new FileOutputStream(newClassName +
".class");
        fos.write(b);
        fos.flush();
    }

    public static void main(String[] args) {
```

```
            try {
                String newClassName = "Socket";
                if (args.length >= 1)
                    newClassName = args[0];
                SocketTransformer fit = new SocketTransformer();
                fit.transform(newClassName);
            } catch (Exception ex) {
                ex.printStackTrace();
            }


    }
}


// Socket Class
class SocketClassAdapter extends ClassAdapter implements Opcodes {

    public SocketClassAdapter(ClassVisitor cv) {
        super(cv);
    }

  //                                            .
    public MethodVisitor visitMethod(int access, String name, String
descriptor, String signature,
            String[] exceptions) {

        if (name.equals("getInputStream")) {
            // getInputStream   __orig$getInputStreram__           .

            System.out.println("Rename getInputStream to
__orig$getInputStream__");
            MethodVisitor mv = cv.visitMethod(ACC_PUBLIC,

                "__orig$getInputStream__", "()Ljava/io/InputStream;",
null, new String[] { "java/io/IOException" });
            mv.visitCode();
            mv.visitEnd();

            return mv;
        }

        return super.visitMethod(access, name, descriptor, signature,
exceptions);
    }

  //       getInputStream                      .

    public void visitEnd() {
        MethodVisitor mv = cv.visitMethod(ACC_PUBLIC, "getInputStream",
"()Ljava/io/InputStream;",
```

```
                    null, new String[] { "java/io/IOException" });

        mv.visitCode();
        mv.visitTypeInsn(NEW, "java/net/FlowLiteSocketInputStream");
        mv.visitInsn(DUP);
        mv.visitVarInsn(ALOAD, 0);
        mv.visitVarInsn(ALOAD, 0);

    //                              ,
        mv.visitMethodInsn(INVOKEVIRTUAL, "java/net/Socket",
"__orig$getInputStream__", "()Ljava/io/InputStream;");
        mv.visitMethodInsn(INVOKESPECIAL,
"jav`a/net/FlowLiteSocketInputStream",

                "", "(Ljava/net/Socket;Ljava/io/InputStream;)V");
        mv.visitInsn(ARETURN);
        mv.visitMaxs(4, 1);
        mv.visitEnd();

    }
}
```

FlowLiteInputStream, Socket, SocketIOCallBack   Socket
                         .

```
[Socket Info] Thread id = 20, Host name =localhost:10583, Status = 1, Access
time = Fri Aug 10 18:11:06 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Fri Aug 10
18:11:06 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Fri Aug 10
18:11:06 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Fri Aug 10
18:11:06 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Fri Aug 10
18:11:06 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Fri Aug 10
18:11:06 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Fri Aug 10
18:11:06 KST 2007
....

[Net IO], Thread id = 20, Bytes read = 736, Access time = Fri Aug 10
18:11:06 KST 2007
```

                ?   Byte code   Socket   I/O   .

                                                                    . File I/O, Network I/O, JDBC Request,
Servlet Request, EJB Request, Struts Request, Spring Framework Request   Enterprise Java
                                                    .

WAS

,   Byte Code Instrumentation
WAS   Java Application

BCI   ,

Part   Java 5 (JDK 1.5)   BCI   JDK
java.lang.instrument   Byte Code Instrumetation

PS)

BCI   Exception Tracking, Object Creation Tracking, File I/O Tracking, Network I/O
Tracking, JDBC Tracking   ,   .
!!!

The Power of Byte Code Instrumentation in Java- Part 6

JDK   JVM   JVM   (Agent),
                          . Java 5,   JDK 1.5   JVMPI(JVM Profiler Interface)
    . Java 5   JVMTI(JVM Tool Interface)   . JVMPI/JVMTI
C/C++   .

Java 5   Java   .
    ~ ~   .

```
Prompt> java

....

-agentlib:[=]
load native agent library , e.g. -agentlib:hprof
see also, -agentlib:jdwp=help and -agentlib:hprof=help
-agentpath:[=]
load native agent library by full pathname
-javaagent:[=]
load Java programming language agent, see java.lang.instrument
```

agentlib/agentpath   JVMPI/JVMTI   .   javaagent
    Java   .

( )   java.lang.instrument   Java
        .

java.lang.instrument

http://java.sun.com/j2se/1.5.0/docs/api/java/lang/instrument/package-summary.html .

JavaAgent BCI

java.lang.instrument .

.
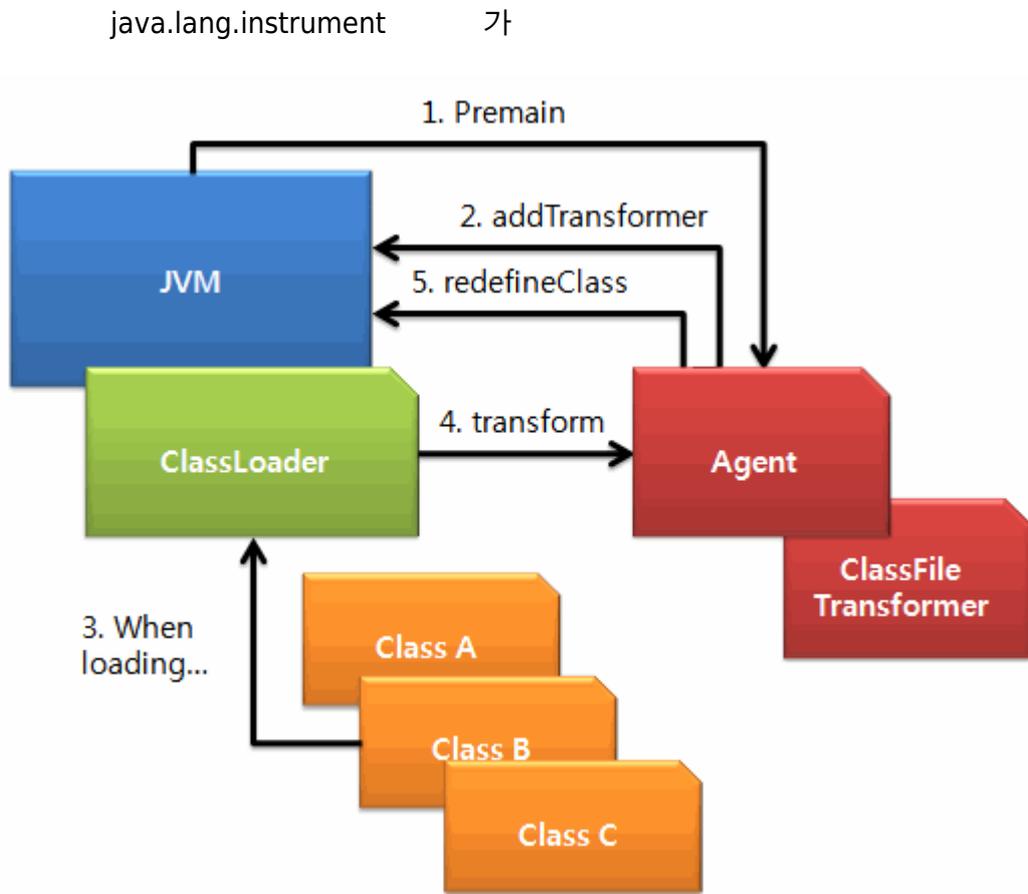
1. Java Agent premain . JVM Java Agent premain
   Agent Application main .
2. Java Agent ClassFileTransformer( ) ,
   Instrumentation.addTransformer JVM .
3. JVM ClassFileTransformer transform
   , .
   ASM .
4. Agent Instrumentation.redefineClasses
   .

JavaAgent

JavaAgent .(ASM
)

```
public class SimpleProfiler implements ClassFileTransformer {

    public SimpleProfiler() {
```

```java
        super();
    }

    public static void premain(String args, Instrumentation inst) {
        try {
            // Redefine preloadedclasses. Especially rt.jar boot class
            ArrayList defs = new ArrayList();
            for( Class c : inst.getAllLoadedClasses()) {
                try {
                    if(c.getName().equals("java.io.File")) { //
  java.io.File             .
                        System.out.println("Redefining class " +
c.getName());

                        ClassReader reader = new ClassReader(c.getName());
                        ClassWriter writer = new
ClassWriter(ClassWriter.COMPUTE_MAXS+ClassWriter.COMPUTE_FRAMES);

                //             . ASM                    .
                // -->                          ,

                        // --> Method Start  Method End               .
                        ClassAdapter adapter = new
SimpleClassAdapter(writer, c.getName);
                        reader.accept(adapter, ClassReader.SKIP_DEBUG);
                        byte[] result = writer.toByteArray();
                        if(result != null) defs.add(new ClassDefinition(c,
result));
                    }
                } catch(Exception ex) {
                    ex.printStackTrace();
                }
            }
            ClassDefinition[] cdef = defs.toArray(new
ClassDefinition[defs.size()]);
            inst.redefineClasses(cdef);
            inst.addTransformer(new SimpleProfiler());
            (new Thread(new MonitoringThread())).start();
        } catch(Exception ex) {
            ex.printStackTrace();
        }
    }

    public byte[] transform(ClassLoader l, String className, Class c,
ProtectionDomain pd, byte[] b)
            throws IllegalClassFormatException {

        if (l != ClassLoader.getSystemClassLoader()) {
            return b;
        }
```

```
        if (className.startsWith("instrument/profiler")) {
            return b;
        }

        System.out.println("Classloader = " + l);
        System.out.println("Class = " + className);

        byte[] result = b;
        ClassReader reader = new ClassReader(b);
        ClassWriter writer = new ClassWriter(ClassWriter.COMPUTE_MAXS +
ClassWriter.COMPUTE_FRAMES);
        ClassAdapter adapter = new SimpleClassAdapter(writer, className); //
          ...
        reader.accept(adapter, ClassReader.SKIP_DEBUG);

        result = writer.toByteArray();

        return result;

    }
}
```

SimpleProfiler   JavaAgent   .

  1. premain   Load   (rt.jar   Class   )   Redefine   .(
)
  2. transform   .

Manifest

JavaAgent   Jar   JVM   . (
profiler.jar   )

    JVM   JavaAgent   Manifest   Jar
        . Mainfest   .

  - Premain-Class: Premain   Agent Class   .   )
    instrument.profiler.SimpleProfiler
  - Boot-Class-Path: Boot classloader   Jar   .   ) profiler.jar;asm-3.0.jar
  - Can-Redefine-Classes: Instrmentation.redefineClasses   Runtime
                            .   . Default   False   ) True

                  Manifest   .

```
Premain-Class: instrument.profiler.SimpleProfiler
Can-Redefine-Classes: True
Boot-Class-Path: profiler.jar;asm-3.0.jar
```

1. instrument.profiler.SimpleProfiler   premain   JavaAgent
2.   (Instrumentation.redefineClasses)
3. profiler.jar(JavaAgent   )   asm-3.0.jar(ASM   )   Boot class path
   .

                                               .   rt.jar   ,
Boot class path   Profiler   Boot class path
   .   ClassLoader   JVM   .

JavaAgent

          JavaAgent Jar   ,   Java   .

```
java -javaagent:profiler.jar [Your Target Applicatin]
```

                                    .   .

```
---------- Call Table ----------------------------
instrument/target/RunThread.do31179[4694]
instrument/target/RunThread.do2791[5730]
instrument/target/RunThread.do1396[3189]
java.io.File.14[0]
java.io.File.getPath8[0]
java.io.File.exists6[0]
instrument/target/RunThread.5[0]
java.io.File.lastModified4[0]
java.io.File.length2[0]
java.io.File.getPrefixLength2[0]
java.io.File.getCanonicalPath2[0]
instrument/target/ProfilerTarget.main1[0]
```

JavaAgent   JVMTI   .   JMX

JavaAgent   JVMTI   Java 5 (JDK1.5)   . JavaAgent   JVMTI
                        .   , Java
                           . (Instrumentation.getObjectSize   JVMTI
GetObjectSize)

                              BCI   BCI
                        .

- Static BCI :
- Load-Time BCI :
- Runt-Time BCI : JVM

Sun   Java 5   JVMTI   JavaAgent
              .   , JVM   JVMTI   C   API
    , BCI   JavaAgent   Java   API
                   .

        JMX   Platform MBean(JVM   MBean)   ,   Java
                                                  .

                      JVMTI   JMX   .

Epilogue

BCI   .
                      ,   Java 5   JVM   BCI   .
      WAS   Java Application   , BCI
          .

                                                          , 3rd party
                                                                            .

PS)

              BCI   Exception, Call Tree, File I/O, Net I/O, JDBC Request
      .   BCI
      .

```
-------------------[Exception Tracking]-------------------
Exception count = 9
[Exception] Exception = java.io.IOException, Message = Something Bad2~~,
Thread id = 11, Caller =
[Sigature]flowlite.server.ExceptionGenerator.doSomething2, Called from =
[Sigature]flowlite.server.ExceptionGenerator.doSomething2, When = Mon Aug 13
11:06:10 KST 2007
flowlite.server.ExceptionGenerator.doSomething2(FlowLiteMBeanServer.java:128
)
flowlite.server.ExceptionGenerator.run(FlowLiteMBeanServer.java:113)
java.lang.Thread.run(Unknown Source)

[Exception] Exception = java.io.IOException, Message = Something Bad2~~,
Thread id = 11, Caller =
[Sigature]flowlite.server.ExceptionGenerator.doSomething2, Called from =
[Sigature]flowlite.server.ExceptionGenerator.run, When = Mon Aug 13 11:06:10
KST 2007
flowlite.server.ExceptionGenerator.doSomething2(FlowLiteMBeanServer.java:128
)
flowlite.server.ExceptionGenerator.run(FlowLiteMBeanServer.java:113)
java.lang.Thread.run(Unknown Source)

...
```

```
-------------------[Active Thread & Call Tree Tracking]-------------------
Thread count = 26
[ActiveThread] Thread id = 62, Thread name = Thread-49, Group name = main
[Call Tree]
[Call][Sigature]flowlite.server.CallTreeGenerator$InnerGenerator.depth1_1,
duration = 31[ms]
[Call][Sigature]java.util.Random.nextLong, duration = 0[ms]
[Call][Sigature]java.lang.Math.abs, duration = 0[ms]
[Call][Sigature]java.lang.Thread.sleep, duration = 0[ms]
[Call][Sigature]flowlite.server.CallTreeGenerator$InnerGenerator.depth2_1,
duration = 15[ms]
[Call][Sigature]java.util.Random.nextLong, duration = 0[ms]
[Call][Sigature]java.lang.Math.abs, duration = 0[ms]
[Call][Sigature]java.lang.Thread.sleep, duration = 15[ms]


...




-------------------[File I/O Tracking]-------------------
File I/O count = 22
[File Info] Thread id = 22, File name = c:test.txt, Status = 2, File mode =
[r+w], read time = Mon Aug 13 11:06:15 KST 2007
[File IO], Thread id = 22, File name =c:test.txt, Bytes read = 1024, Access
time = Mon Aug 13 11:06:15 KST 2007
[File IO], Thread id = 22, File name =c:test.txt, Bytes read = 1024, Access
time = Mon Aug 13 11:06:15 KST 2007
...




Socket Info] Thread id = 20, Host name =localhost:17904, Status = 1, Access
time = Mon Aug 13 11:06:13 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Mon Aug 13
11:06:13 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Mon Aug 13
11:06:13 KST 2007
[Net IO], Thread id = 20, Bytes read = 1024, Access time = Mon Aug 13
11:06:13 KST 2007


...




-------------------[JDBC I/O Tracking]-------------------
Connection count = 1
Statement count = 1
FetchCount count = 100
[Connection] Statement count = 10, DB = Oracle, Oracle Database 10g
Enterprise Edition Release 10.2.0.3.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options, Created = 2007-08-13
```

```
[Statement] Query = SELECT name FROM t_pstmt_test WHERE id = ?, Execution
count = 10, Fetch size = 10, Statement Type = Prepared
[Execution] Column Count = 1, Fetch count = 1, {Parameters} = (1, 4),
[Fetch] Fetched value count = 1
(1, name4),

...
```

From:
<https://atl.kr/dokuwiki/> - **AllThatLinux!**

Permanent link:
https://atl.kr/dokuwiki/doku.php/bci_-_the_power_of_bytecode_instrumentation_in_java

Last update: **2017/06/16 03:00**