June 2nd - 4th 2015
Copenhagen, Denmark

YOUDEBUG

Scriptable Java Debugger

Wolfgang Schell
twitter.com/jetztgradnet

YOUDEBUG

Scriptable Java Debugger

Developed by Kohsuke Kawaguchi

(creator of Hudson/Jenkins)

DEBUGGING

DEBUGGING

using an **IDE**?

DEBUGGING

using **jdb**?

DEBUGGING

using **gdb**?

DEBUGGING

using **logging**?

DEBUGGING

using **println**?

DEBUGGING

try **YouDebug!**

WHAT'S INSIDE

JDI*

+ YouDebug

+ 

+ your debug script

* Java Debug Interface

FEATURES

define **breakpoints**

evaluate arbitrary **expressions**

List **threads**, inspect **stack** frames

access/modify local **variables**

obtain **stack trace**

obtain **heap dump**

call **methods**

BREAKPOINTS

Break when reaching a specific **line**

Break when an **exception** is thrown

Break when a **field** is referenced or updated

Break when a **class** is loaded/unloaded

Break when a **thread** is created/destroyed

Break when a **method** is entered/exited

Break when a **monitor** is waited/contended

USE CASES

USE CASES

- Iterative bug investigation
- Monitoring & information gathering
- Lock breaking (drop to previous frame)
- Monkey patching
- Waiting for the bug to appear (detach/reattach)
- Limited access to target system

DEBUGGING

IDE

source code
documentation
network access

Application

data
users
configuration

PRODUCTION?

PRODUCTION

~~IDE~~

~~source code~~ source confidential

~~documentation~~ docs confidential

~~network access~~ firewall, NAT, proxies

Application

~~data~~ compliance

~~users~~ privacy

~~configuration~~ security

SOLUTION

Bring your debug script to the application

PRODUCTION

YouDebug

debug script

Application

data

users

configuration

BASIC RECIPE

1. Run application in debug mode

2. Create debug script

3. Run debug script
(optionally capture output)

BASIC RECIPE

1.Run application in debug mode

```
> java -agentlib:jdwp=transport=
dt_socket,server=y,suspend=n,
address=5005 -jar myapp.jar
```

(this also works with your favorite container or service wrapper)

BASIC RECIPE

2. Create debug script

```
vm.methodEntryBreakpoint(
    “com.my.EntryPoint","main") {
                method ->
  println “hey, we are in main"
}
```

BASIC RECIPE

3.Run debug script using

```
> java -jar youdebug.jar -socket
5005 yourscript.ydb
```

(additional jars required: groovy-all.jar, args4j.jar, tools.jar)

SOME EXAMPLES

EXAMPLES (1)
Line breakpoint

```
vm.breakpoint("net.jetztgrad.buggyweb.MyServlet",35) {
  println "at buggy position in MyServlet, line 35"
}
```

EXAMPLES (II)
access/modify value

```
vm.breakpoint("net.jetztgrad.buggyweb.MyServlet",35) {
  println "age=" + age
  String ageParam = request.getParameter("age")
  println "setting age to $ageParam"
  age = ageParam
}
```

EXAMPLES (III)

get thread dump and VM info

```
vm.suspend()
println vm.virtualMachine.name()
println vm.virtualMachine.description()
vm.threads*.dumpThread()
vm.resume()
vm.close()
```

EVENTS

- breakpoint(location)
- exceptionBreakpoint(type)
- methodEntryBreakpoint(type, method)
- methodExitBreakpoint(type, method)
- accessWatchpoint(field)
- modificationWatchpoint(field)
- monitorWait(type) (*)
- monitorWaited(type) (*)
- monitorContendedEnter(type)(*)
- monitorContendedEntered(type) (*)

(*) pending contribution

LOOKUP & ACTIONS

- ref(type)/_(type)
- _new(type)
- forEachClass(type)
- loadClass(type)
- suspend()
- resume()
- dumpHeap(String)
- dumpAllThreads()
- withFrozenWorld(Closure)
- exit(status)
- instanceCounts(types[])

DEMO TIME

UNDER THE HOOD

WHAT'S INSIDE

JDI*

YouDebug

Groovy

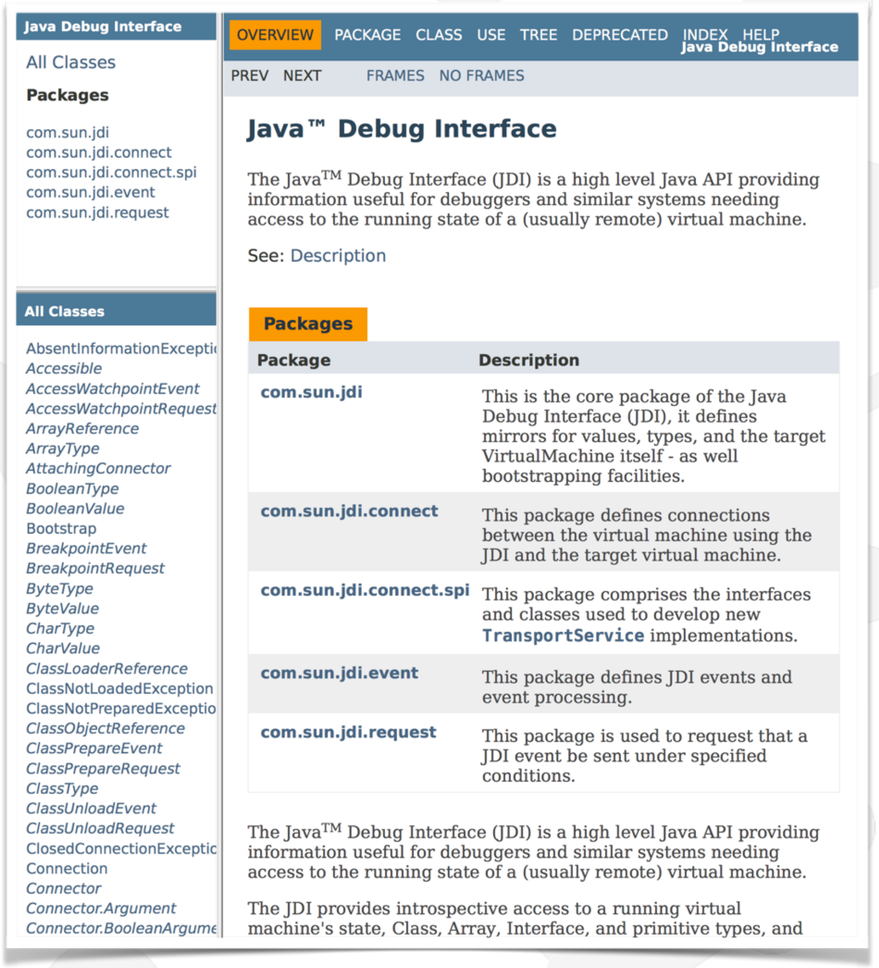
your debug script

* Java Debug Interface

GR Conf EUROPE 2013

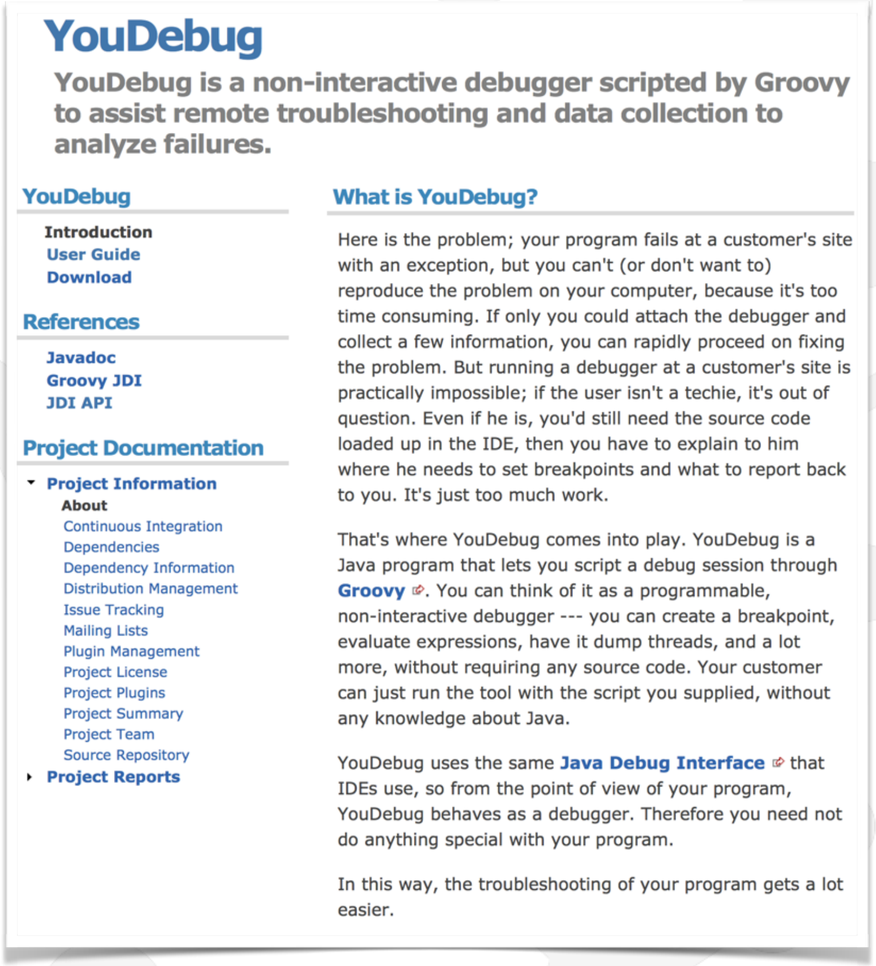
JAVA DEBUG INTERFACE (JDI)

- API to connect to or launch a JVM
- supports local and remote connections
- VirtualMachine object provides access to breakpoints, objects
- Can be used to access fields and invoke methods



YOUDEBUG

- YouDebug is a thin wrapper around JDI
- Some Groovy magic is providing syntactic sugar with convenient access to the target's JVM types, fields, variables and methods.
- everything can be changed dynamically by your debug script!



YouDebug

YouDebug is a non-interactive debugger scripted by Groovy to assist remote troubleshooting and data collection to analyze failures.

YouDebug

- **Introduction**
- **User Guide**
- **Download**

References

- **Javadoc**
- **Groovy JDI**
- **JDI API**

Project Documentation

- **Project Information**
  - **About**
  - Continuous Integration
  - Dependencies
  - Dependency Information
  - Distribution Management
  - Issue Tracking
  - Mailing Lists
  - Plugin Management
  - Project License
  - Project Plugins
  - Project Summary
  - Project Team
  - Source Repository
- **Project Reports**

What is YouDebug?

Here is the problem; your program fails at a customer's site with an exception, but you can't (or don't want to) reproduce the problem on your computer, because it's too time consuming. If only you could attach the debugger and collect a few information, you can rapidly proceed on fixing the problem. But running a debugger at a customer's site is practically impossible; if the user isn't a techie, it's out of question. Even if he is, you'd still need the source code loaded up in the IDE, then you have to explain to him where he needs to set breakpoints and what to report back to you. It's just too much work.

That's where YouDebug comes into play. YouDebug is a Java program that lets you script a debug session through **Groovy**. You can think of it as a programmable, non-interactive debugger --- you can create a breakpoint, evaluate expressions, have it dump threads, and a lot more, without requiring any source code. Your customer can just run the tool with the script you supplied, without any knowledge about Java.

YouDebug uses the same **Java Debug Interface** that IDEs use, so from the point of view of your program, YouDebug behaves as a debugger. Therefore you need not do anything special with your program.

In this way, the troubleshooting of your program gets a lot easier.

ISSUES & LIMITATIONS

DEBUG MODE

YouDebug needs **debugging enabled** ;-)

Note: enabling debugging causes some **performance** overhead and has **security** implications! Also, it cannot be enabled after application start.

ERROR HANDLING

Apply proper error handling, otherwise you might end up with a **suspended JVM**, if your debug script throws an **exception**.

MIRRORING

JDI mirrors elements such as types, objects, methods, fields, variables. There some hidden complexities when using the convenience methods provided by YouDebug.

Accessing data or invoking methods is only possible when the JVM is suspended on a breakpoint!

BEWARE OF THE HEISENBUG

Heisenbug

From Wikipedia, the free encyclopedia

For the Linux distribution codenamed Heisenbug, see Fedora (operating system).

In computer programming jargon, a **heisenbug** is a software bug that seems to disappear or alter its behavior when one attempts to study it.[1] The term is a pun on the name of Werner Heisenberg, the physicist who first asserted the observer effect of quantum mechanics, which states that the act of observing a system inevitably alters its state. In electronics the traditional term is probe effect, where attaching a test probe to a device changes its behavior.

Sometimes **println** is still the best solution!

YOUDEBUG

Thank you!

Questions?

LINKS

Introduction	http://youdebug.kohsuke.org/
Userguide	http://youdebug.kohsuke.org/user-guide.html
JDI	http://docs.oracle.com/javase/8/docs/jdk/api/jpda/jdi/index.html
Download	http://central.maven.org/maven2/org/kohsuke/youdebug/1.5/youdebug-1.5.jar
Source Code	https://github.com/kohsuke/youdebug