JUDCon 2013: United States
JBoss Users & Developers Conference

Full Control and Transparency
Advanced Troubleshooting and Testing

Josh West & Andrew Dinn
Introductions

• Josh West - JBoss Solution Architect
  • Previously JBoss Senior Consultant
  • Red Hat for 6+ years

• Andrew Dinn - Principal Software Engineer
  • OpenJDK team
  • Byteman team lead
  • 30 years as Developer & Researcher in Computing
Goals

- Java Containers and Dependencies aren't black boxes
- Expecting the unexpected with break testing
- Showcase tools for emergencies, and while being proactive
  - Byteman
  - Tattletale
  - Java Decompilers
- Get you using JBoss Tools
- Get you involved with the JBoss Community
Meet Jim the Admin

- Works for ACME Corp
- Used to be a developer, now an administrator
- Under pressure to deploy application to production
- Journey he takes while troubleshooting an application
- Discovers new tools and approaches
Simple Web Application

- WAR, Spring, Hibernate, JDBC on Tomcat
- Experiencing Database connection errors
- Errors happening intermittently
Jim’s Initial Approach

- If at first you don’t succeed on Google...
- Doesn’t have Red Hat JBoss EWS Subscription
- GrepCode.com for Hibernate & JDBC driver
JBoss Tattletale

- Generates report on the full Tomcat install, with app included
- Version of Hibernate in is the application
- Location of class he saw in stacktrace
Using Tattletale

- Download & Unzip Tattletale from: http://www.jboss.org/tattletale/downloads
  ```
  java -jar tattletale.jar [-exclude=<excludes>] <input-directory> <output-directory>
  ```
- Open report at <output-directory>/index.html
- Can also use ant or maven
- Add script/alias for quick report generation
  ```
  export TATTLE_HOME=~/.apps/tattletale-1.2.0.Beta2
tattle() {
    java -Xmx1024m -jar $TATTLE_HOME/tattletale.jar $1 ~/$dev/tattletale/$2
  }
  $ tattle ~/dev/servers/tomcat7029 tomcat7029
  ```
Tattletale Uses

- Identify dependencies between JAR files
- Find missing classes from the classpath
- Spot if a class/package is located in multiple JAR files
- Spot if the same JAR file is located in multiple locations
- With a list of what each JAR file requires and provides
- Verify the SerialVersionUID of a class
- Find similar JAR files that have different version numbers
- Find JAR files without a version number
- Find unused JAR archives
- Identify sealed / signed JAR archives
- **Locate a class in a JAR file**
- Get the OSGi status of your project
- Remove black listed API usage
- And generate the same reports for your .WAR and .EAR archives
Browsing the Source

• Jim looks at source of class where exception is thrown
• Any Java Class or JAR can be decompiled
• Some vendors obfuscate but interfaces are intact (eg: javax)
Browsing the Source
Java Decompilers

- **JD-GUI** by Emmanuel Dupuy
  - Has plugins for Eclipse and IntelliJ
  - Works on Windows, OSX, or Linux
  - http://java.decompiler.free.fr/jd-gui/downloads

- **JAD** by Pavel Kouznetsov
  - Can be run from command line
  - http://www.varaneckas.com/jad/

- **cavaj** by SureShot Software
  - Windows only :(

- No known open source decompilers
  - **Brad Davis** is working on one, called Candle
  - https://github.com/bradsdavis/candle-decompiler
Severed Connection Theory

- Tries to increase logging for modules
  - Value he needs to see to prove his theory is not logged
- Remote debugging the container
  - Not all JARs have debug symbols
- Decompiles source, modifies it, and recompile it
  - ...un successfully
- What other options does he have?
  - Static analysis alone won't lead him to a conclusion
  - Can't debug remotely
  - Can't modify binaries
Byteman to the Rescue

• Jim uses Byteman to inject logging of connection value
• Byteman allows you to inject java code at runtime
• Write declarative rule to match injection point
• Execute java code as consequence

Download:
• https://www.jboss.org/byteman/downloads

Documentation:
• https://www.jboss.org/byteman/documentation
Anatomy of a Rule

- Rule Skeleton

```
RULE <rule name>
CLASS <class name>
METHOD <method name>
BIND <bindings>
IF <condition>
DO <actions>
ENDRULE
```

- Sample Rule

```
RULE Connection Pool info
CLASS org.apache.tomcat.jdbc.pool.ConnectionPool
METHOD setupConnection
AT LINE 330
IF TRUE
  DO debug("JIMS TEST tomcat connection info:" + $con.toString());
ENDRULE
```
Executing Byteman

1) Add Java Agent and script to JAVA_OPTS

```bash
export JAVA_OPTS="-Dorg.jboss.byteman.verbose -Dorg.jboss.byteman.compileToBytecode -javaagent:$/BYTEMAN_HOME/lib/byteman.jar=
  sys:$/BYTEMAN_SCRIPTS/jimsHelperClasses.jar,
  script:$/BYTEMAN_SCRIPTS/jimsscript2.btm"
```

2) Install into a running java application

```bash
$JAVA_HOME/bin/jps -l # lists all the java processes
$BYTEMAN_HOME/bin/bminstall.sh <pid> # install agent process <pid>
$BYTEMAN_HOME/bin/bmsubmit.sh -s <jarfile> # add helper Jar to classpath
$BYTEMAN_HOME/bin/bmsubmit.sh -i <script> # install script
```
Byteman Tips

- Byteman agent can listen to external interfaces
  
  bminstall.sh [-p port] [-h host]  # defaults to localhost:9091
  bmsubmit.sh [-p port] [-h host] script

- Parse and typecheck script offline
  
  bmcheck.sh [-cp classpath] [-p package] script1 [...scriptN]

- Print rule parse, typecheck, compile, and execute on System.out
  
  -Dorg.jboss.byteman.verbose

- Compile byteman script for faster execution
  
  -Dorg.jboss.byteman.compileToBytecode
Root Cause Proven

• Jim proves root cause with byteman

• Determines that developer’s exception handling isn’t adequate

• Wants to make sure this doesn't happen in other applications

ONE DOES NOT SIMPLY

MOVE ON AFTER FIXING AN ISSUE
Injecting Faults

- Instead of injecting logging lines, inject exception throw statements
- Alternatively change variable values...public or private
- Use Byteman instead of “kill -9” for large distributed tests
- Note the keyword “THROW”

```
RULE Connection Pool Fault Injection
CLASS org.apache.tomcat.jdbc.pool.ConnectionPool
METHOD setupConnection
AT ENTRY
IF TRUE
  DO THROW new java.sql.SQLException("Injected exception, assert this is handled properly")
ENDRULE
```
Using BMUnit

- JUnit 4 Sample -->
- BMUnitRunner
- @BMScript(<script_name>)
- @BMRule(<props>)
- Pair with Arquillian for embedded container tests
- http://arquillian.org/

```java
package com.acme.testing;

import org.jboss.byteman.contrib.bmunit.BMScript;
import org.jboss.byteman.contrib.bmunit.BMUnitRunner;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(BMUnitRunner.class)
public class SampleBytemanTest {
  @Test(expected=MyException.class)
  @BMScript("faultinject.btm")
  public void test() {
    // prepare
    // execute
    // assert
  }
}
```
Conclusion

- Jim discovered new approaches
- Tattletale for visibility of classes / archives
- Java Decompiler to analyze how stuff works
- Byteman to inject logic for troubleshooting
- Byteman for testing
Conclusion

- Download and try the tools
  - http://www.jboss.org/byteman
  - http://www.jboss.org/tattletale

- Find uses in your projects

- Look at more real world byteman samples
Q & A