# OSGi Alliance Community Event

## Runtime Diagnosis of Stale References in the OSGi™ Services Platform

Kiev Gama & Didier Donsez
Université Grenoble 1, France
Kiev.Gama@imag.fr
Didier.Donsez@imag.fr

## Objectives

- Bad OSGi™ Programming Practices
- How to diagnosis one (ie Stale References) ?

## Outline

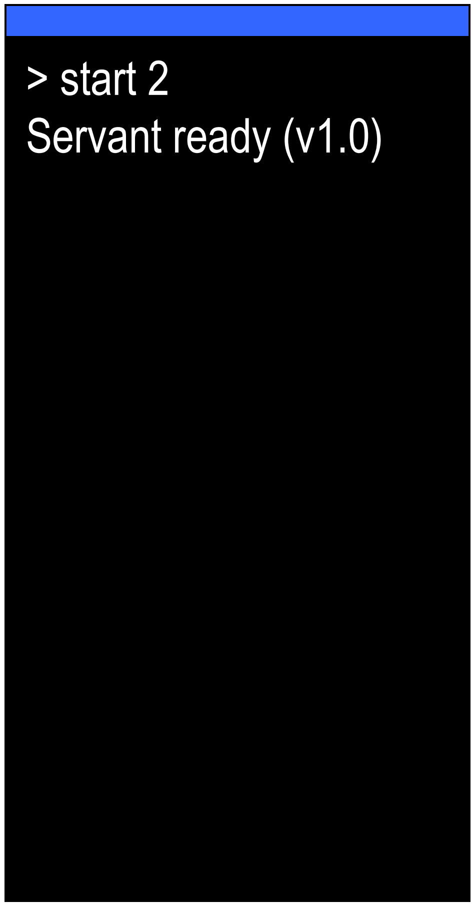
- The Stale References Pathology
- Need for Diagnosis
- The ServiceCoroner tool
- Experimentation
- Conclusion
- Perspectives
- Short demo of the tool

## What are Stale References?

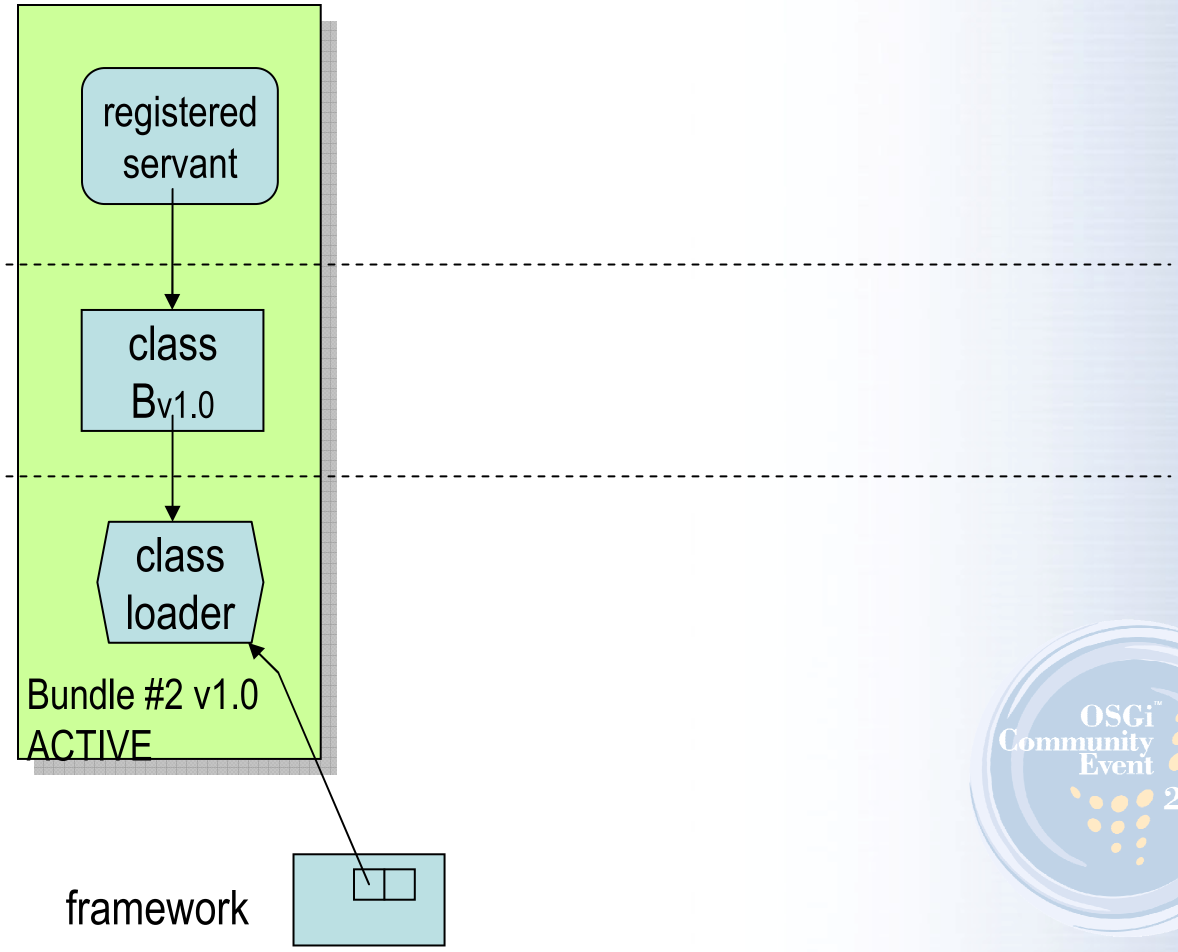
*“a reference to a Java object that belongs to the class loader of a bundle that is stopped or is associated with a service object that is unregistered”*

OSGi R4 Section 5.4

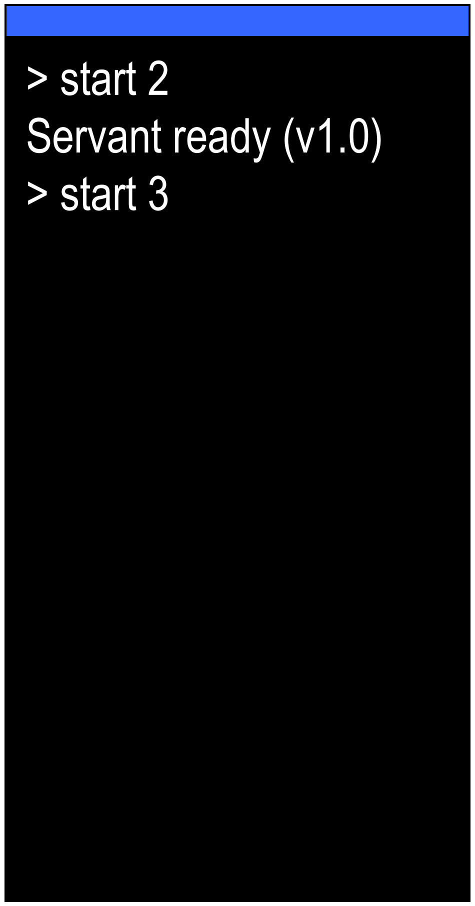
## An example of Stale Reference Pathology? (i) initial



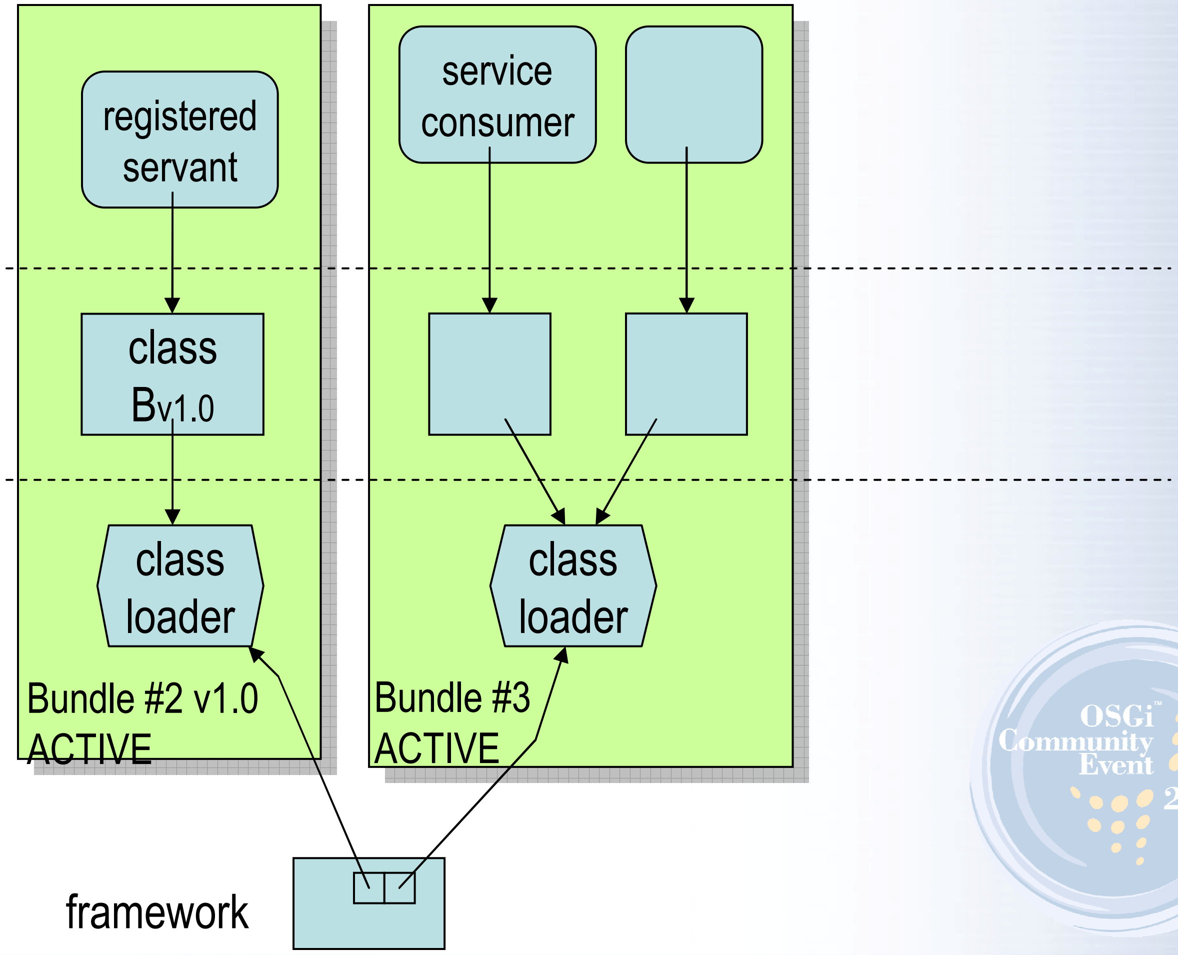
```
> start 2
Servant ready (v1.0)
```



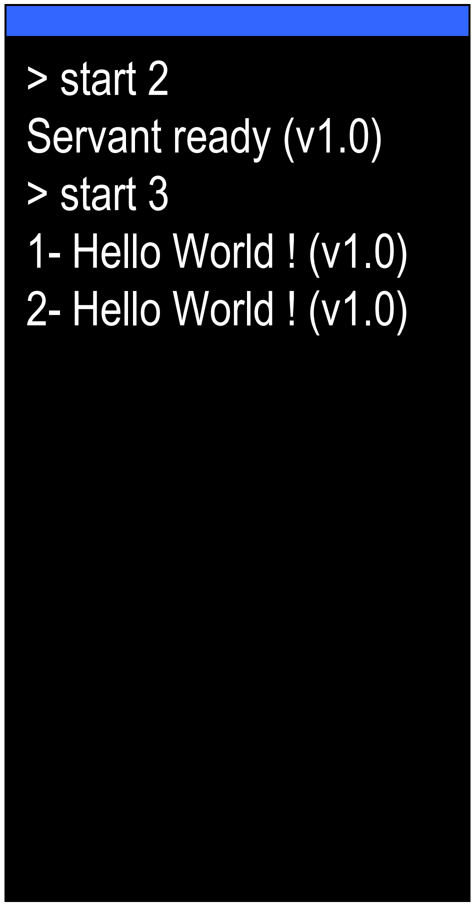
## An example of Stale Reference Pathology? (i) initial



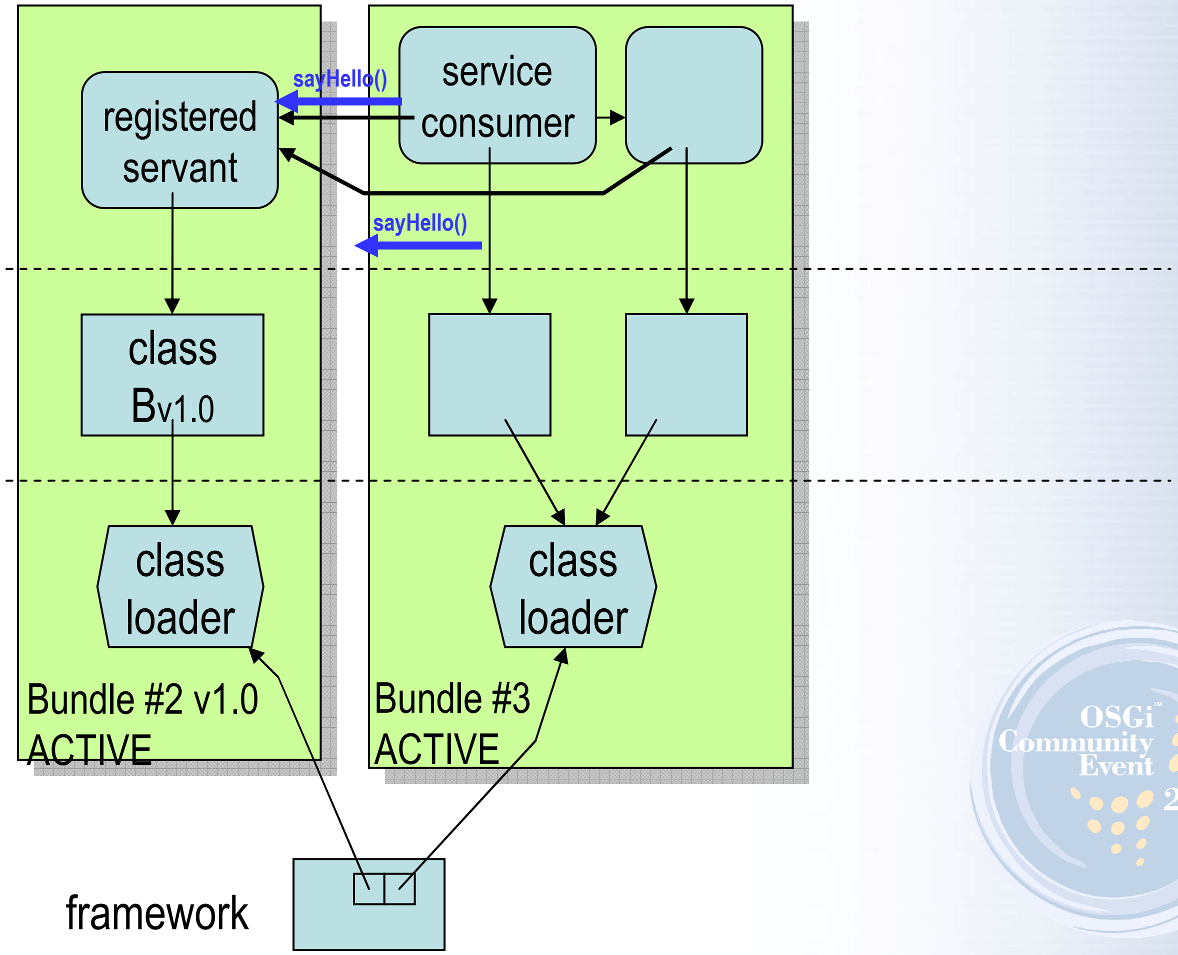
```
> start 2
Servant ready (v1.0)
> start 3
```



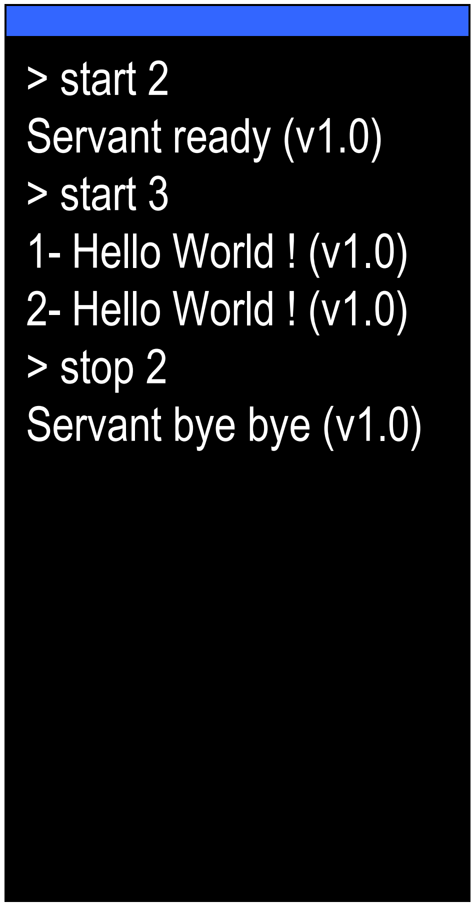
## An example of Stale Reference Pathology? (i) initial



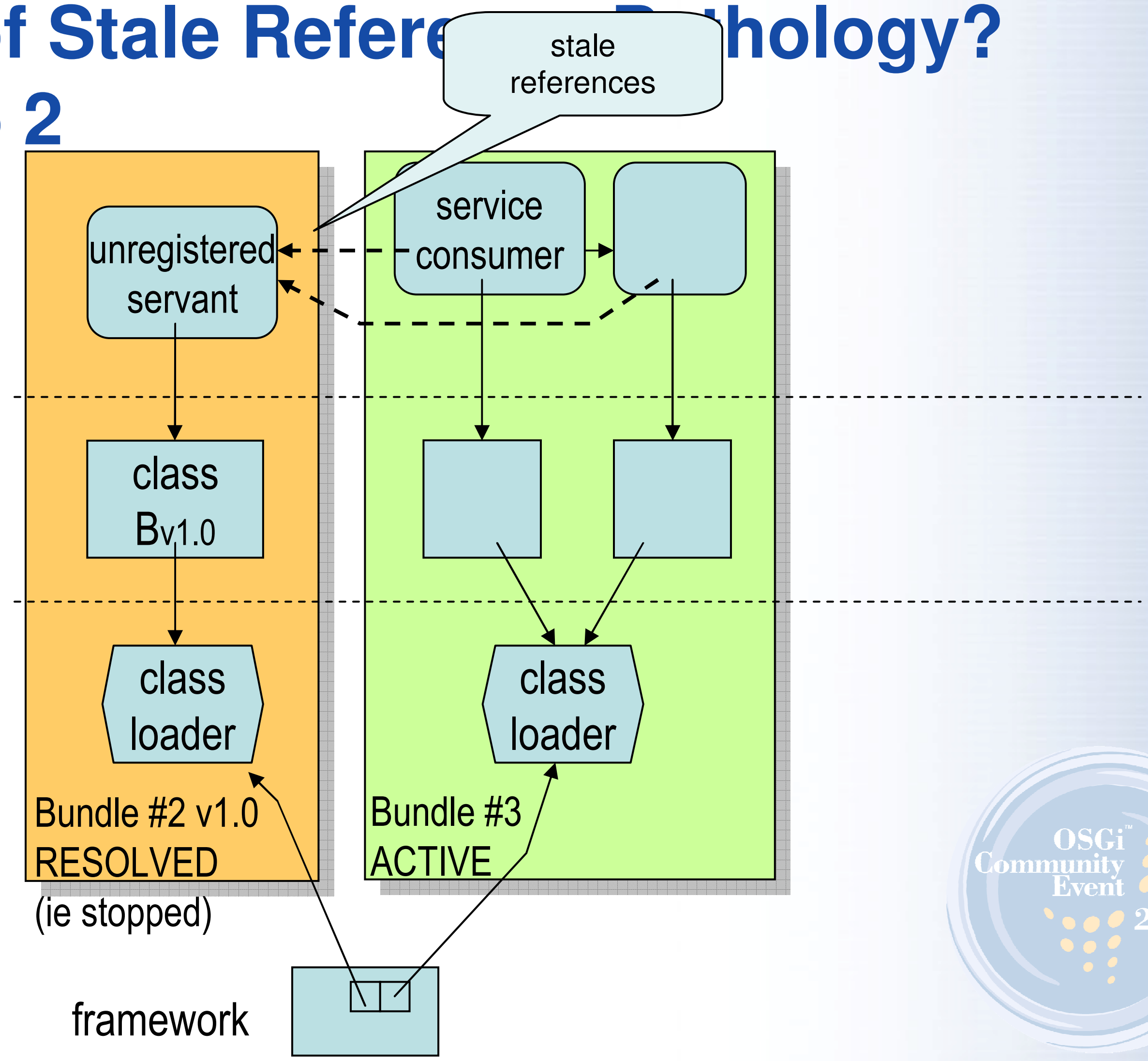
```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
```



## An example of Stale Refere… …thology? (ii) After stop 2

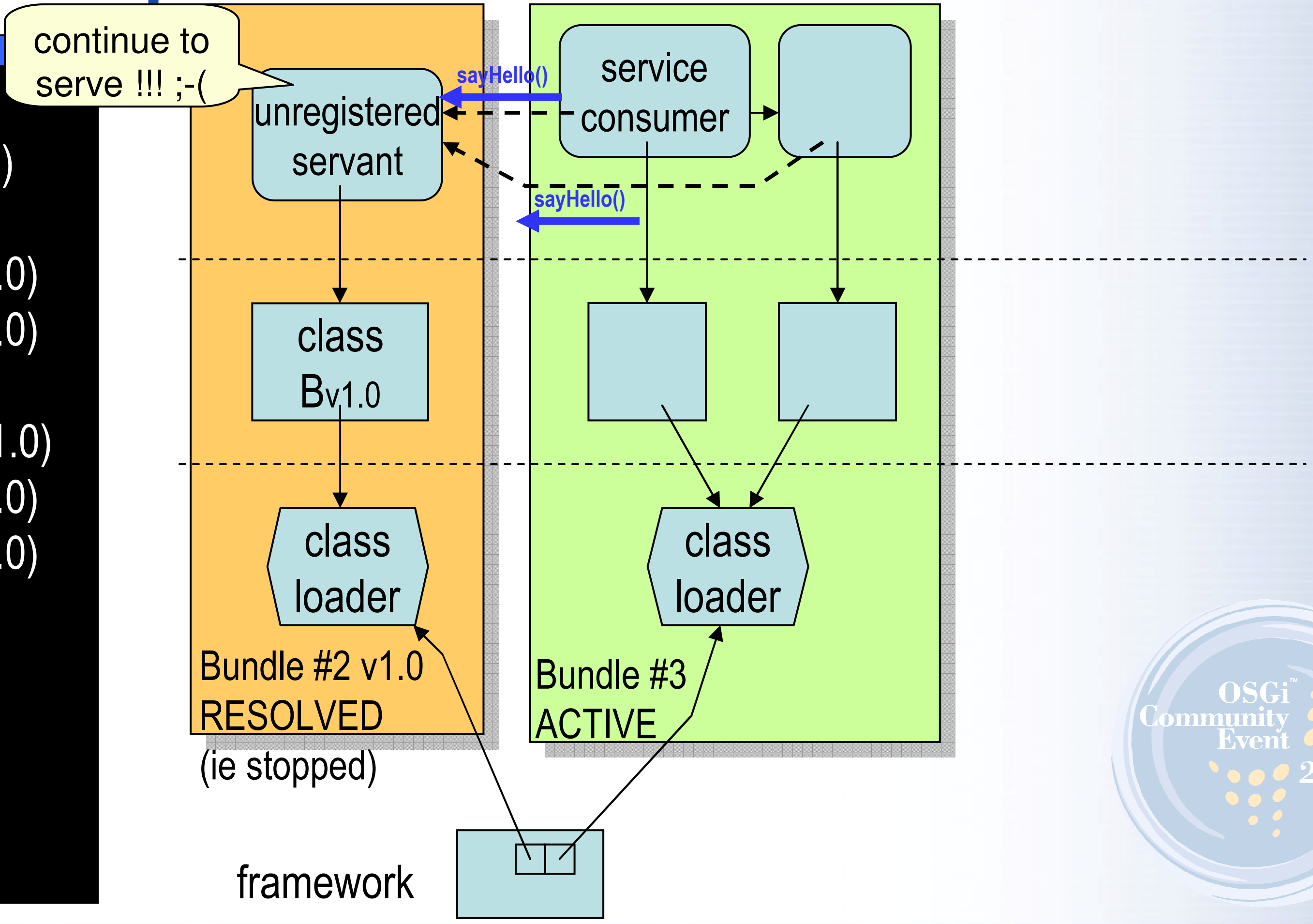


```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
> stop 2
Servant bye bye (v1.0)
```

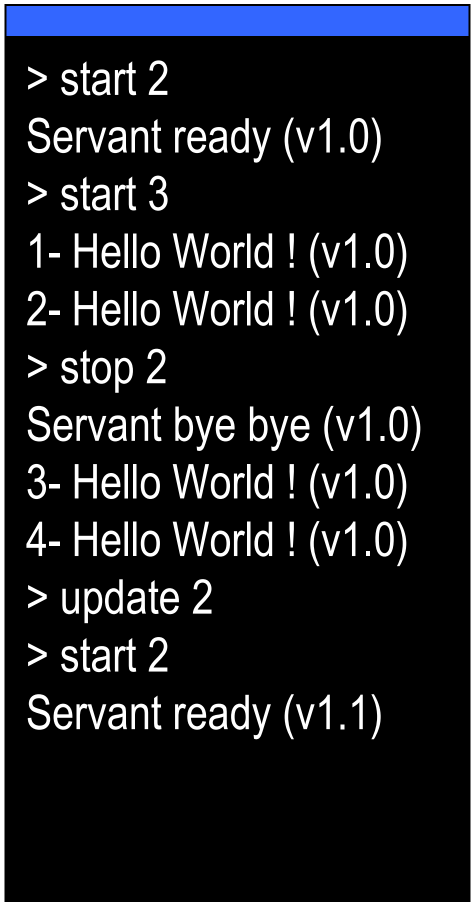


## An example of Stale Reference Pathology? (iii) After stop 2

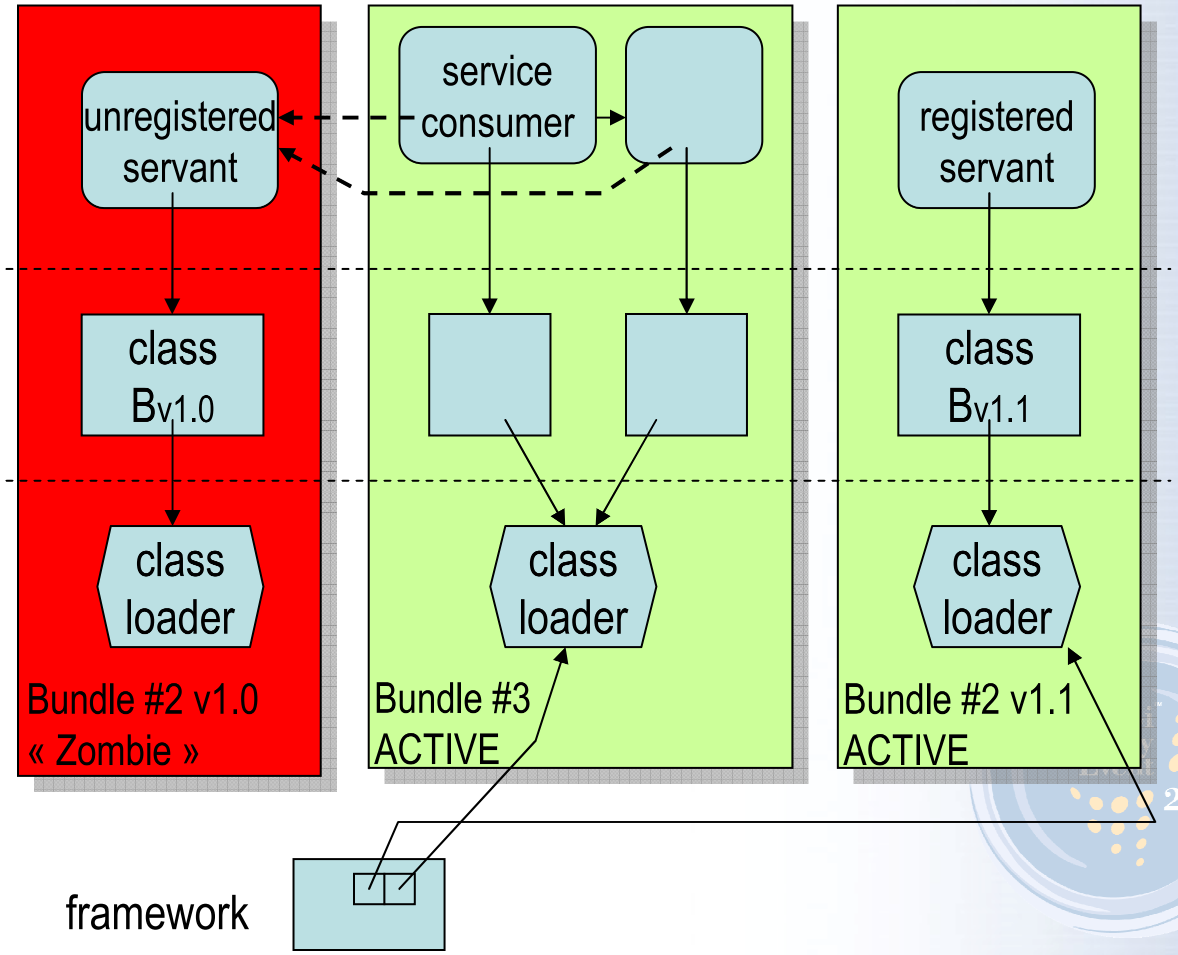
```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
> stop 2
Servant bye bye (v1.0)
3- Hello World ! (v1.0)
4- Hello World ! (v1.0)
```



## An example of Stale Reference Pathology? (iii) After update 2 & start 2

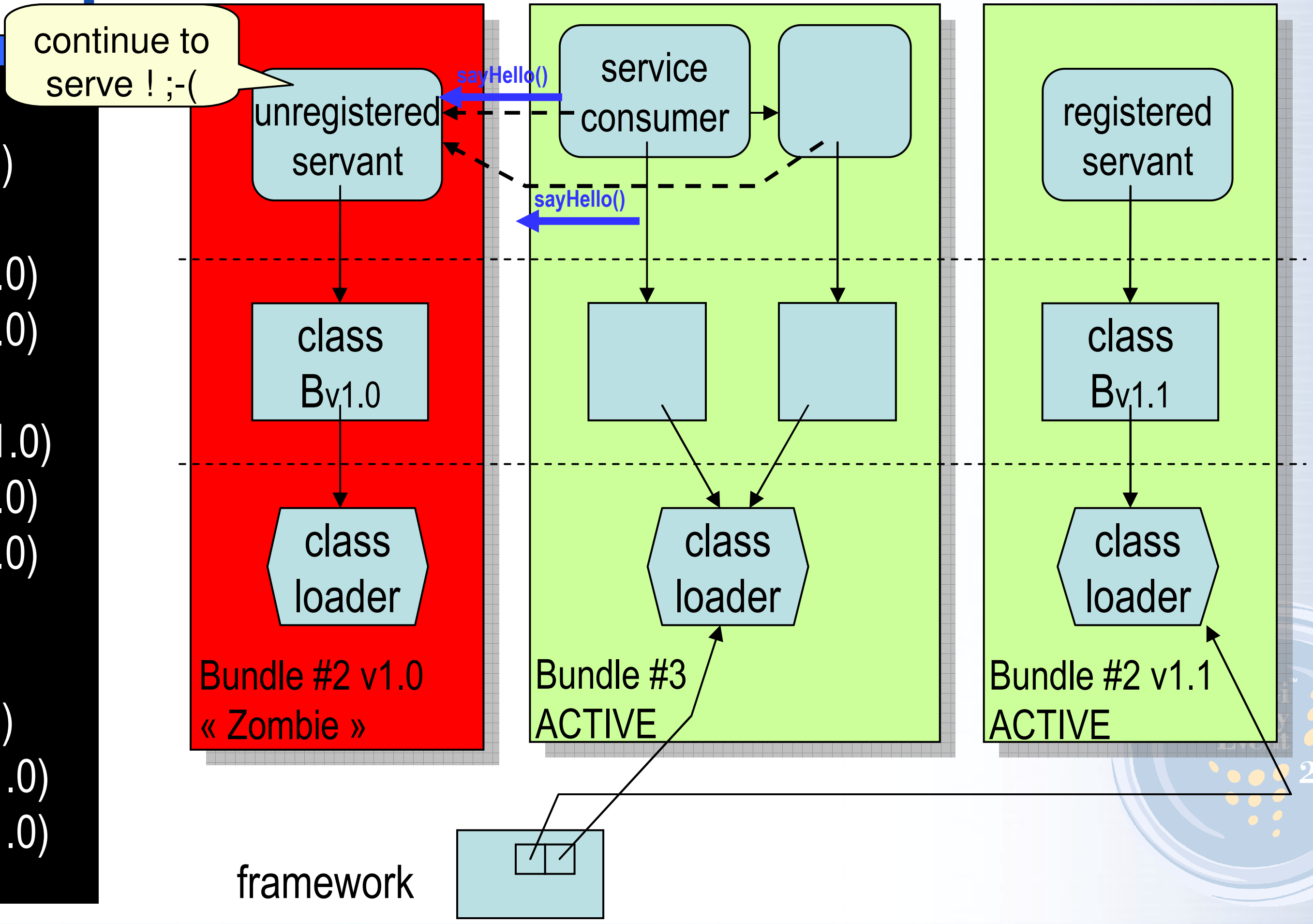


```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
> stop 2
Servant bye bye (v1.0)
3- Hello World ! (v1.0)
4- Hello World ! (v1.0)
> update 2
> start 2
Servant ready (v1.1)
```



# An example of Stale Reference Pathology? (iii) After update 2 & start 2

```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
> stop 2
Servant bye bye (v1.0)
3- Hello World ! (v1.0)
4- Hello World ! (v1.0)
> update 2
> start 2
Servant ready (v1.1)
 5- Hello World ! (v1.0)
 6- Hello World ! (v1.0)
```



## Bad Consequences in OSGi-based SW

- Memory leaks
  - Retention of the classloader of a stopped or uninstalled bundle
  - Retention of all java.lang.Class loaded by that bundle
- Utilization of invalid services → Inconsistencies!
  - Service is unregistered but still used (wrong!)
  - Its context is most likely inconsistent
    - e.g. closed connections
  - Possible exceptions upon service calls
    - good because we can see the problem
  - Silent propagation of incorrect results (worst case!)
    - E.g. Returning old cached-data

## Other « stale » pathologies
*(Bad OSGi™ Programming Practices)*

- “Forwarded references”
  - From one bundle to another
- “Stale” threads *(ie orphan threads)*
  - bundle has stopped but created threads have not
- Unregistered MBeans, RemoteObjects, …
- Unreleased resources
  - sockets, file descriptors, locks, …

## How to ensure « stale reference free » applications?

2 cases of OSGi™ SW projects

- From-scratch OSGi™ development
- Bundlization of Legacy codes
  - Really frequent (Eclipse 2.0 to 3.0, JOnAS, WebLogic, …)
  - Module with or without Services/Extension Points

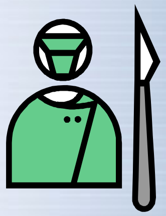
***Gurus' advice (Peter, BJ, Rick (in the other room)…)***

1. Follows Good OSGi™ programming practices
   - Who trusts their developers ?
2. Uses Component Models
   - Necessary but not enough

- Stale references may be there but we can't see them…

→ We need **Diagnosis**
**victim** bundles x **guilty** bundles

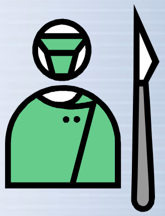
## The ServiceCoroner tool



- A diagnosis tool for detecting
  stale references in OSGi™ applications
- “Inspector” of services death
- Runtime diagnosis
- Points out victim bundles/services
  and possible suspects

*The coroner is a legal examiner that investigates the causes of unnatural deaths in English speaking countries. Not all coroners have forensic pathology knowledge, but for illustration purposes we have named our tool as ServiceCoroner.

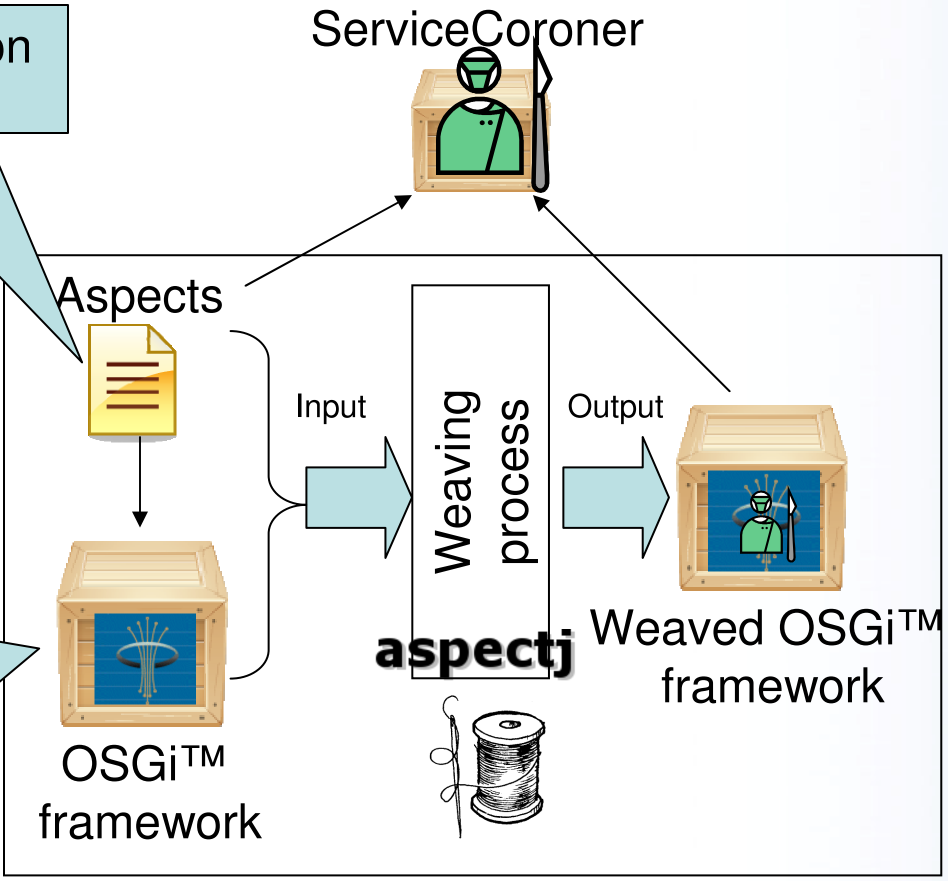
## The ServiceCoroner tool (cont.)



- Diagnosis of service references “pathologies”
- How to enable OSGi™ to provide that info?
  - Use AOP: diagnosis as a separate concern; portability
- Relies on weak references to know if a service has been GCd
  - Small delays (wait for GC) to get actual info
- Listens to service and bundle events and log them
- Minimal performance impacts
  - Weaving Service Registration; Class Loader and Thread Creation

## The Weaving Process

Portable aspects on the OSGi R4 API



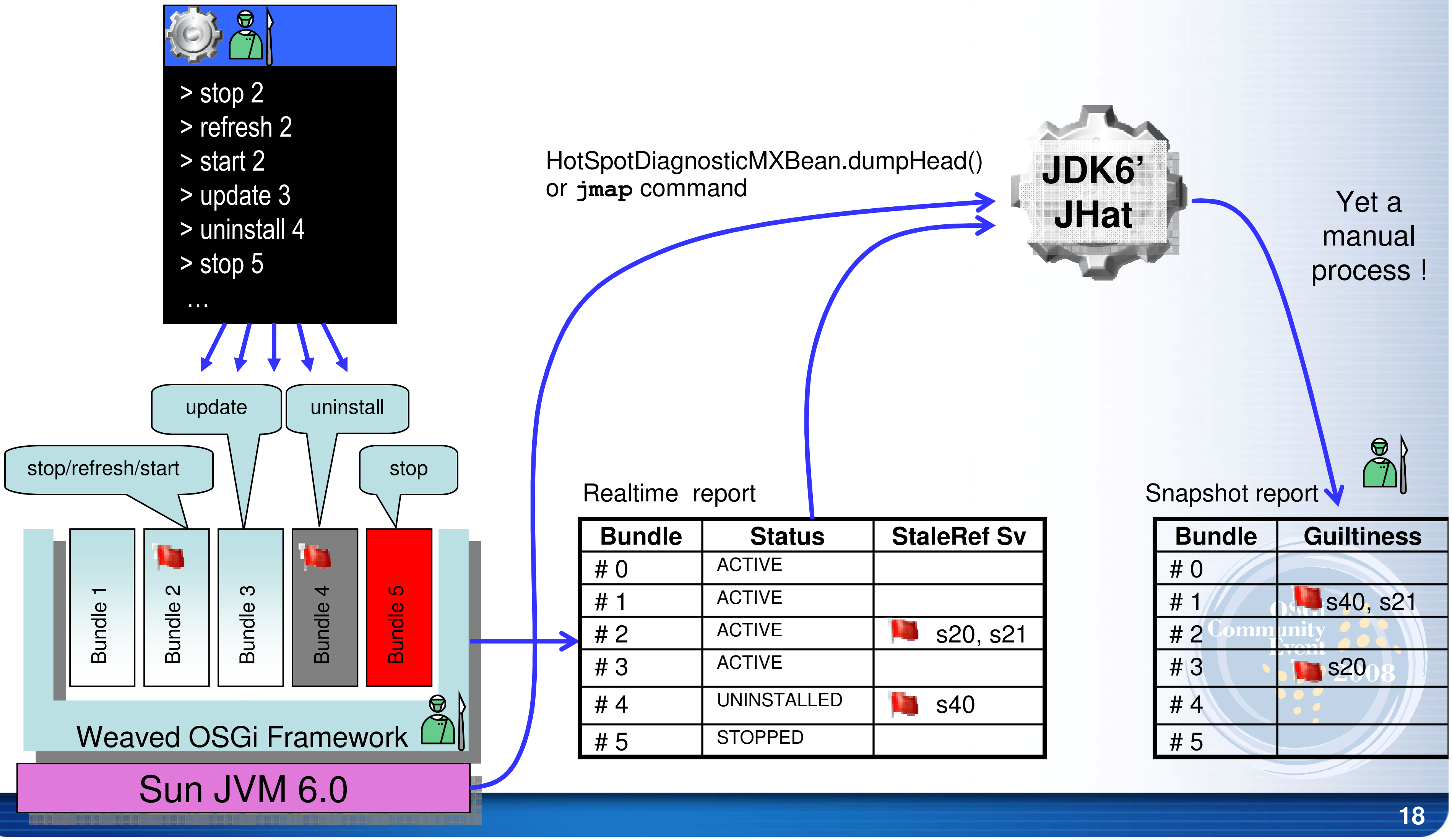
**Tested Frameworks:**

- Apache Felix v1.0
- Equinox v3.2.0
- Equinox v3.3.0*
- Knopflerfish v2.0.4

…

* That version uses signed jars. We manually removed the class hashes information from the original equinox jar manifest in order to bypass checking

## The Diagnosis Process



```
> stop 2
> refresh 2
> start 2
> update 3
> uninstall 4
> stop 5
 …
```

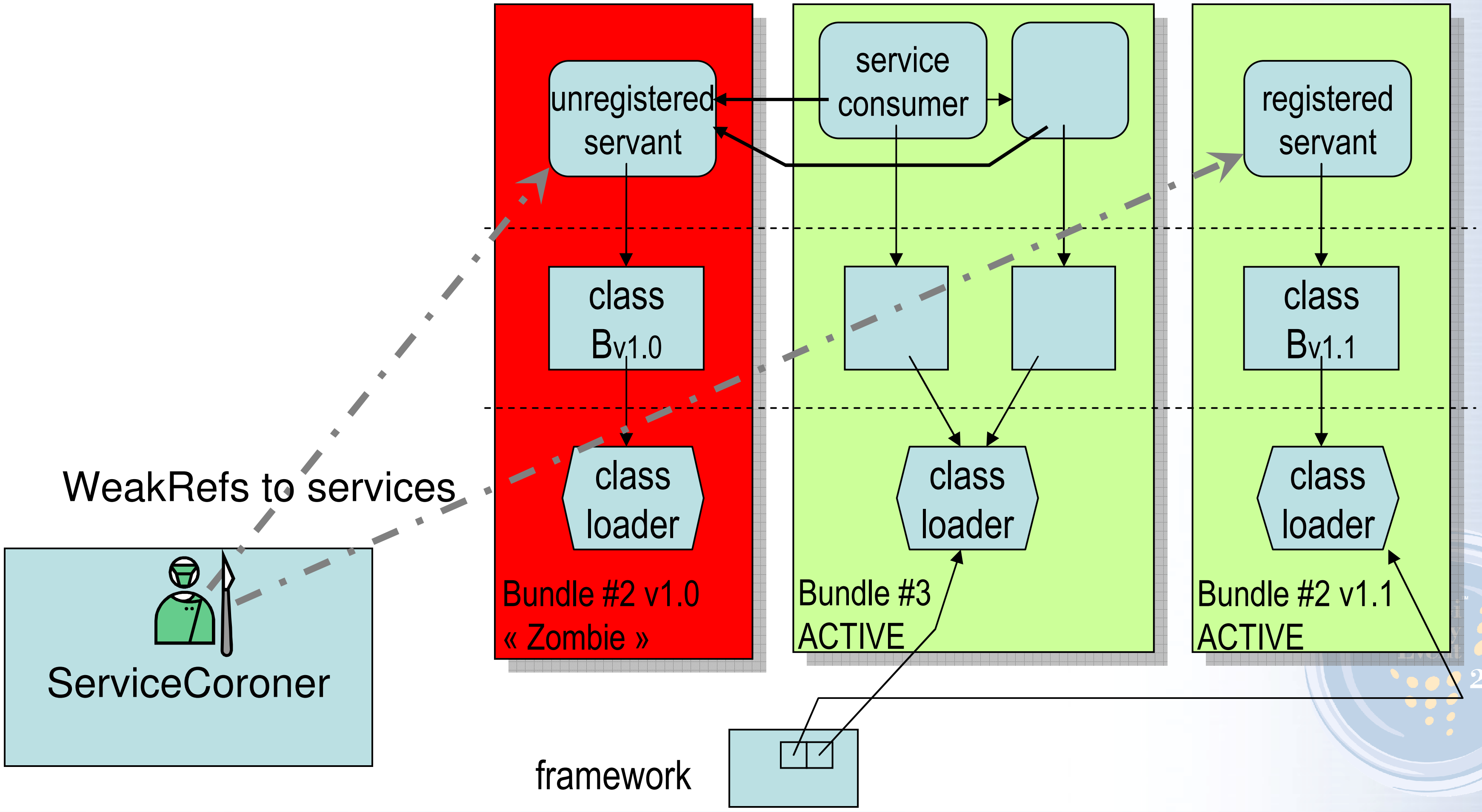
Realtime report

| Bundle | Status | StaleRef Sv |
|---|---|---|
| # 0 | ACTIVE | |
| # 1 | ACTIVE | |
| # 2 | ACTIVE | 🚩 s20, s21 |
| # 3 | ACTIVE | |
| # 4 | UNINSTALLED | 🚩 s40 |
| # 5 | STOPPED | |

Snapshot report

| Bundle | Guiltiness |
|---|---|
| # 0 | |
| # 1 | 🚩 s40, s21 |
| # 2 | |
| # 3 | 🚩 s20 |
| # 4 | |
| # 5 | |

## Watching services

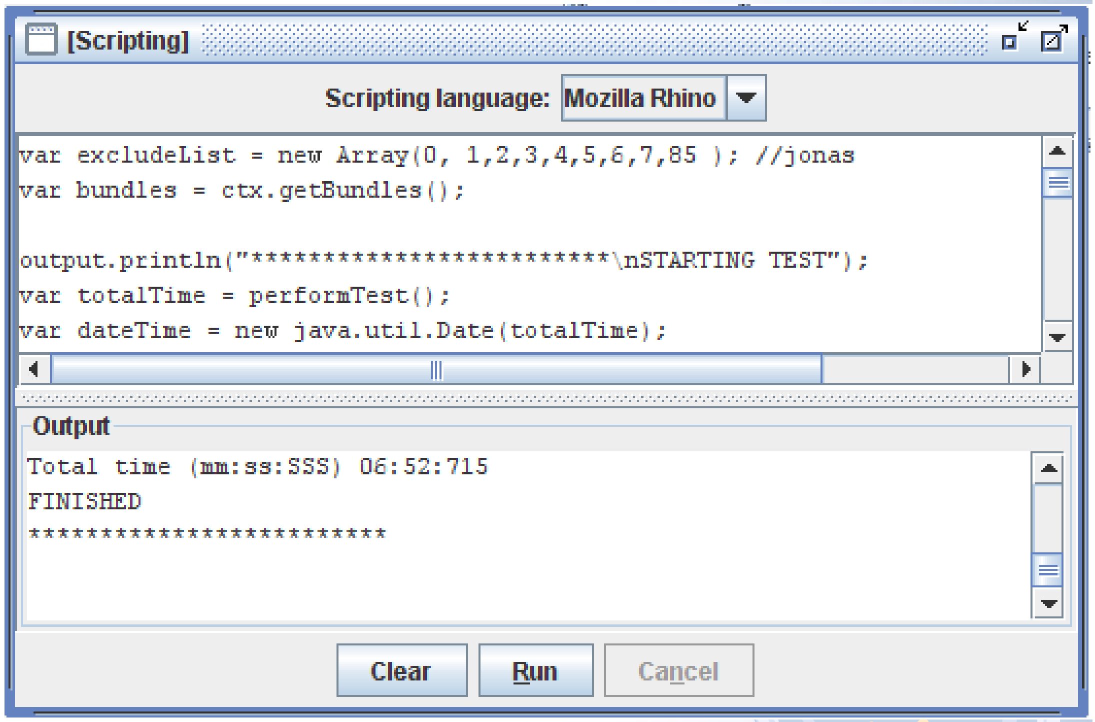


## The Diagnosis Process (cont.)

- In vitro (active)
  - Force life cycle events
  - Not ideal for a production environment.
  - Reasonable for a testing environment
  - Faster results
  - "Brute force" may not lead to events that reflect the application's architecture
- In vivo (passive)
  - Wait for "normal" life cycle events
    - resulted from normal administration tasks
  - Ideal for production environments
  - Results are more precise
  - Take longer (maybe days!)

## Executing the Active Process Diagnosis

- Run a script in the ServiceCoroner scripting console
- Script performs a call to update in bundles that have registered services
- 10 second interval between each update call
- Core bundles are not updated (e.g. bundle 0, libraries, …)
- Use an “exclude list” containing such bundles



```
var excludeList = new Array(0, 1,2,3,4,5,6,7,85 ); //jonas
var bundles = ctx.getBundles();

output.println("************************\nSTARTING TEST");
var totalTime = performTest();
var dateTime = new java.util.Date(totalTime);
```

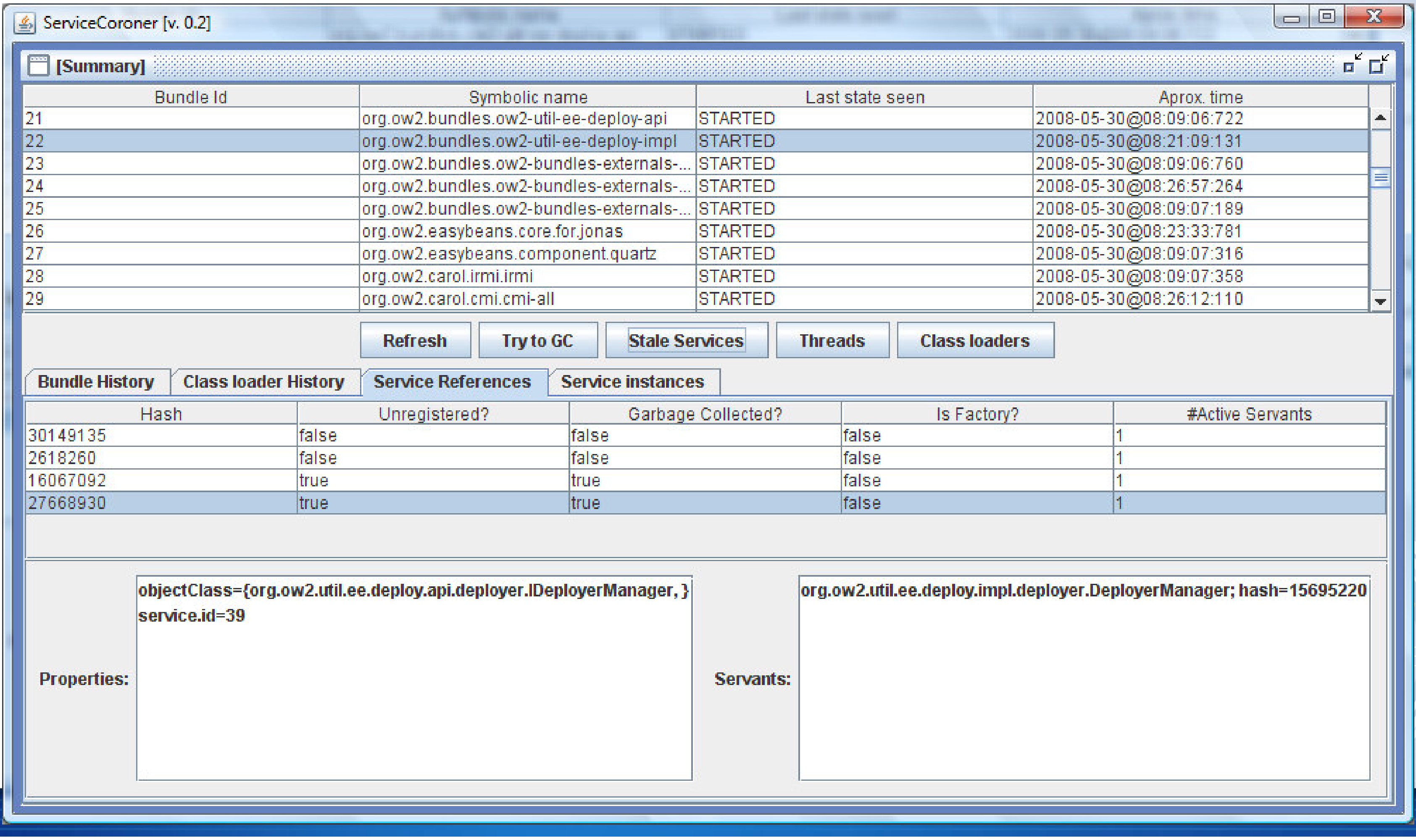
```
Total time (mm:ss:SSS) 06:52:715
FINISHED
*************************
```

## Issues

- Fine grained analysis to find out object referrers
  - Used jhat and jmap embedded in the application
  - Semi-automated process
  - Only in Sun JVM
  - Limitations: Large memory footprint;
  - Weaving at bundle load time

- How to find out the bundle classloader
  - During bundle activation is fine, but…
  - …what about the extender model case and library bundles?
  - We need an accurate mechanism to infer a bundle's classloader

## ServiceCoroner Graphical User Tools (i) Standalone



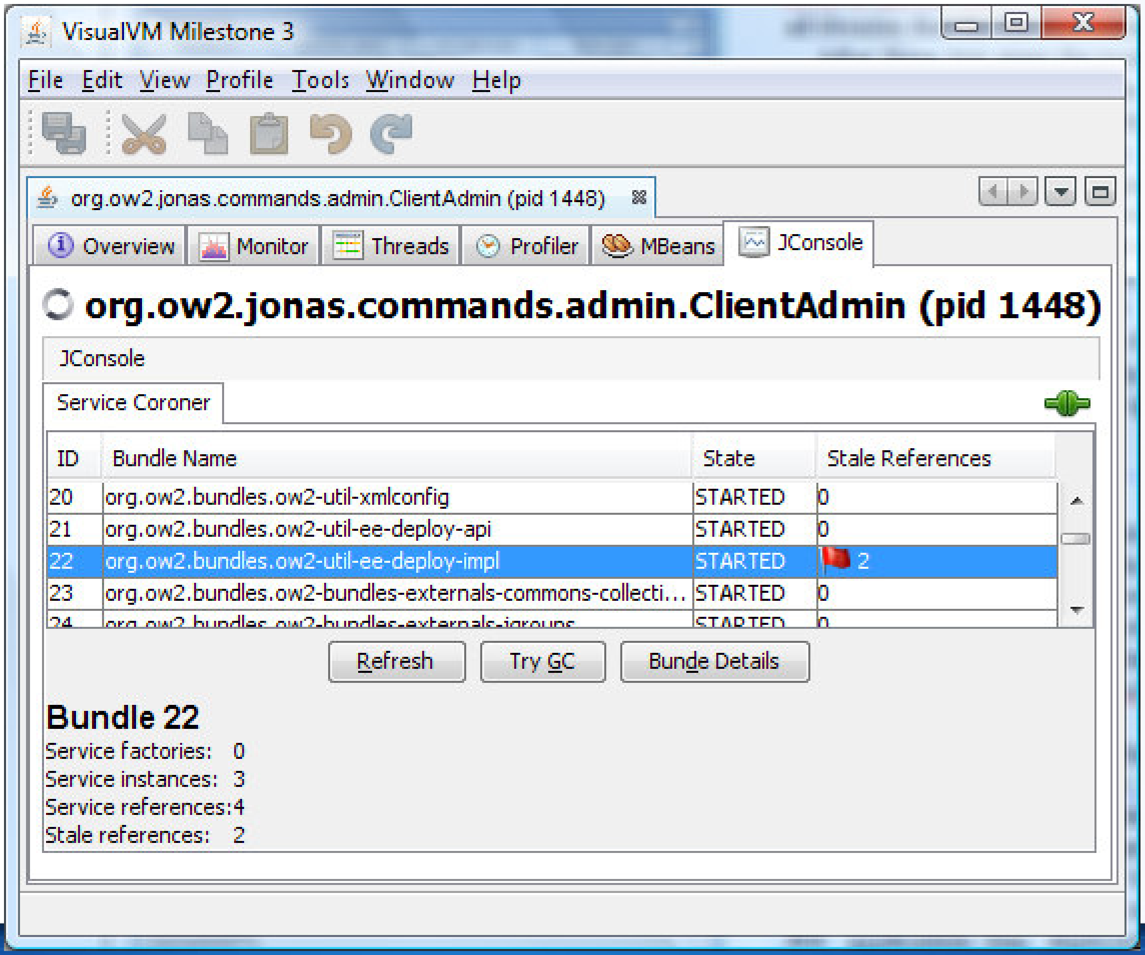
| Bundle Id | Symbolic name | Last state seen | Aprox. time |
|---|---|---|---|
| 21 | org.ow2.bundles.ow2-util-ee-deploy-api | STARTED | 2008-05-30@08:09:06:722 |
| 22 | org.ow2.bundles.ow2-util-ee-deploy-impl | STARTED | 2008-05-30@08:21:09:131 |
| 23 | org.ow2.bundles.ow2-bundles-externals-... | STARTED | 2008-05-30@08:09:06:760 |
| 24 | org.ow2.bundles.ow2-bundles-externals-... | STARTED | 2008-05-30@08:26:57:264 |
| 25 | org.ow2.bundles.ow2-bundles-externals-... | STARTED | 2008-05-30@08:09:07:189 |
| 26 | org.ow2.easybeans.core.for.jonas | STARTED | 2008-05-30@08:23:33:781 |
| 27 | org.ow2.easybeans.component.quartz | STARTED | 2008-05-30@08:09:07:316 |
| 28 | org.ow2.carol.irmi.irmi | STARTED | 2008-05-30@08:09:07:358 |
| 29 | org.ow2.carol.cmi.cmi-all | STARTED | 2008-05-30@08:26:12:110 |

| Hash | Unregistered? | Garbage Collected? | Is Factory? | #Active Servants |
|---|---|---|---|---|
| 30149135 | false | false | false | 1 |
| 2618260 | false | false | false | 1 |
| 16067092 | true | true | false | 1 |
| 27668930 | true | true | false | 1 |

Properties:
objectClass={org.ow2.util.ee.deploy.api.deployer.IDeployerManager, }
service.id=39

Servants:
org.ow2.util.ee.deploy.impl.deployer.DeployerManager; hash=15695220

## ServiceCoroner Graphical User Tools (ii) JConsole/VisualVM Plugin


VisualVM Milestone 3
File Edit View Profile Tools Window Help
org.ow2.jonas.commands.admin.ClientAdmin (pid 1448)
Overview
Monitor
Threads
Profiler
MBeans
JConsole

org.ow2.jonas.commands.admin.ClientAdmin (pid 1448)

JConsole

Service Coroner

| ID | Bundle Name | State | Stale References |
|---|---|---|---|
| 20 | org.ow2.bundles.ow2-util-xmlconfig | STARTED | 0 |
| 21 | org.ow2.bundles.ow2-util-ee-deploy-api | STARTED | 0 |
| 22 | org.ow2.bundles.ow2-util-ee-deploy-impl | STARTED | 🟥 2 |
| 23 | org.ow2.bundles.ow2-bundles-externals-commons-collecti... | STARTED | 0 |
| 24 | org.ow2.bundles.ow2-bundles-externals-jgroups | STARTED | 0 |

**Bundle 22**
Service factories: 0
Service instances: 3
Service references:4
Stale references: 2

## Experiments

- Motivation
  - Validate ServiceCoroner on real-life OSGi-based SW
    - Widely used
    - OSS and Non-Commercial OSGi apps to avoid court trials or man hunts ;-(
    - More than 100,000 LoC (Not « HelloWorld » Toys)
  - Answer to « Is the Stale Reference pathology so frequent ? »
- Choices : SW using Services
  - JOnAS, Sling, SIP Communicator, Newton
  - Remark: some use (partially) Component Models
  - Remark: Eclipse (Extension Points) & GlassFish (HK2 comp.) are not pertinent !
- And the results are …

*Stale References are not a myth !*

## Experiment results

| I | OSGi-based software | **JOnAS (JavaEE server)** | **SIP Comm. (multiprotocol VoIP and Chat UA)** | **Newton (SCA container)** | **Sling (Content Repository)** |
|---|---|---|---|---|---|
| II | Version | 5.0.1 | Alpha 3 | 1.2.3 | 2.0 incubator snapshot |
| III | OSGi Impl. | Felix 1.0 | Felix 1.0 | Equinox 3.3.0 | Felix 1.0 |
| IV | Bundles using Component Models | 20 iPOJO | 6 Service Binder | 0 | 18 Declarative Services |
| V | Lines of Code | Over 1 500 000 | Aprox. 120 000 | Aprox. 85 000 | Over 125 000 |
| VI | Total Bundles | 86 | 53 | 90 | 41 |
| VII | Initial No. of Service Refs. | 82 | 30 | 142 | 105 |
| VIII | No. of Bundles w/ Stale Svcs. | 4 | 17 | 25 | 2 |
| IX | **No. of Stale Services Found** | **7** | **19** | **58** | **3** |
| X | **No. of Stale Threads** | **2** | **4** | **0** | **0** |
| XI | **Stale Services Ratio (IX/VII)** | **8.5 %** | **63 %** | **40.8%** | **2.8%** |

[1] Actually the whole Newton implementation is an SCA constructed on top of OSGi, but its bundles did not use an OSGi component model like the other analyzed applications did.

## Conclusion

- Stale References are not a myth !
- But Component Models are helpful !
  - JOnAS bundles that used a component model (iPOJO) did not present stale references
  - Same for Sling
  - SIP Communicator errors were mostly due to GUI objects retaining references, and services kept as class members
  - Newton does not used identified OSGi component model …

## Perspectives

- Release ServiceCoroner in an OSGi OSS Community
- Automate guilty bundles identification
- Add other pathologies diagnostics to ServiceCoroner
  - “Stale” extension points
    - Eclipse IDE & RCP’ plugins
  - Other “stale pathologies” related to the R4.1’ Extender Model
    - HK2, SCA …
- Collaborations to improve current OSGi-based SWs
  - JOnAS but others are welcome

## More about the ServiceCoroner

- 5000 word-long paper to appear in the 34th EuroMicro SEAA CBSE track: “Service Coroner: A Diagnostic Tool for locating OSGi Stale References”

- Videos, documentations and tools available on
  - http://www-adele.imag.fr/users/Kiev.Gama/dev/osgi/servicecoroner

  Or googlize "ServiceCoroner"

- Extra stuff : JConsole & VisualVM Plugins for OSGi
  - Bundle admin, Felix/Equinox/KF remote shells, …
  - http://www-adele.imag.fr/users/Didier.Donsez/dev/osgi/jconsole.osgi/

# OSGi Alliance Community Event

Very short demo !

*Only the victims detection*

# OSGi Alliance Community Event

## Q & A

## Abstract

- The OSGi™ Service Platform allows the dynamic loading and unloading of bundles and their classes during JVM execution. However, developers must take special care to handle the departure of services and bundles. Since OSGi™ bundles are not isolated from each other in separate object spaces, when they are stopped there is no guarantee they are safely removed from runtime. There is a high possibility of inconsistencies due to the mishandling of such events. The platform cannot ensure that objects from a stopped bundle will no longer be referenced by other bundles – a problem referred by OSGi™ specification (Core R4 section 5.4) as stale references. This happens as an invisible problem that compromises application integrity: Stale References cause memory leaks and prevent the classes of a bundle to be unloaded from memory; inconsistencies can silently propagate errors throughout the system due to calls to an unregistered service that returns stale data (e.g., old cached data).
- This presentation details: different patterns of stale references occurrence; situations where that problem may compromise application correctness; techniques based on Aspect Oriented Programming to detect such problems during application runtime; a fail-stop mechanism on services to avoid the propagation of incorrect results due to calls to stale references; and the results of an experiment on four open source OSGi™ technology based applications.
- It is difficult to say that OSGi™ applications and components are ready to cope with the OSGi™ dynamics, since there are no custom mechanisms to measure or evaluate that. The usage of component models does not necessarily avoid the occurrence of stale references. We have developed a tool called Service Coroner, which implements the techniques that we present and is able to provide information on stale references objects.
- We have validated this diagnostic tool by doing a runtime analysis in four open source applications constructed on top of OSGi™: OW2 JOnAS 5.0.1, SIP Communicator Alpha 3, Newton 1.2.3 and Apache Sling. All applications are of significant size, especially JOnAS, whose core is about 400 000 lines of code but comes to over 1 500 000 when the other components are taken into account. Some of those applications are partially developed with component models for the OSGi™ Platform: Service Binder, R4 Declarative Services and iPOJO. The experiment shows that even using such mechanisms applications still present stale references are not completely ready to handle the dynamic update of components. After the simulation of some life cycle events (update, start, stop) on a limited range of bundles in each the application we found out a number of stale references. The stale services proportion in relation to the initial number of registered services in JOnAS, SIP Communicator, Newton and Sling were 8.5 %, 63%, 40.8% and 2.8 %, respectively. JOnAs presented 2 stale threads and SIP Communicator presented 4.
- The presentation would be concluded with a 5-minute demonstration of the ServiceCoroner diagnostic tool and its 2 GUIs: standalone and remote (on JConsole6/VisualVM)

## Bios

### Kiev Gama

Kiev Gama (kiev.gama@imag.fr) is currently a Master Student at Université Grenoble 1 (France). He has a bachelor's degree in Computer Science from Universidade Catolica de Pernambuco (Brazil) and has earned a one year post-graduate degree in Mobile and Converging Systems from Universidade do Estado do Amazonas (Brazil). He has 6 years of experience of development in Java, J2ME, JavaEE and .NET technologies having worked in several companies of the brazilian information technology market. He is interested in researches on service oriented architecture and component-based software engineering.

### Didier Donsez

Didier Donsez (didier.donsez@imag.fr) is a full professor of computer science at the University Grenoble 1 (France). His research is focused on service oriented architecture and component-based software engineering in the context of Machine-to-Machine applications. He had 7 years of experience in OSGi software engineering for J2ME to JavaEE runtimes. He is the current chairman and co-founder of the OSGi Users Group France. He contributes also to OSS communities (Apache, OW2 …). He earned his PhD in Computer Sciences (1994) at University Paris 6 and a HDR in Computer Sciences (2006) at University Grenoble 1.

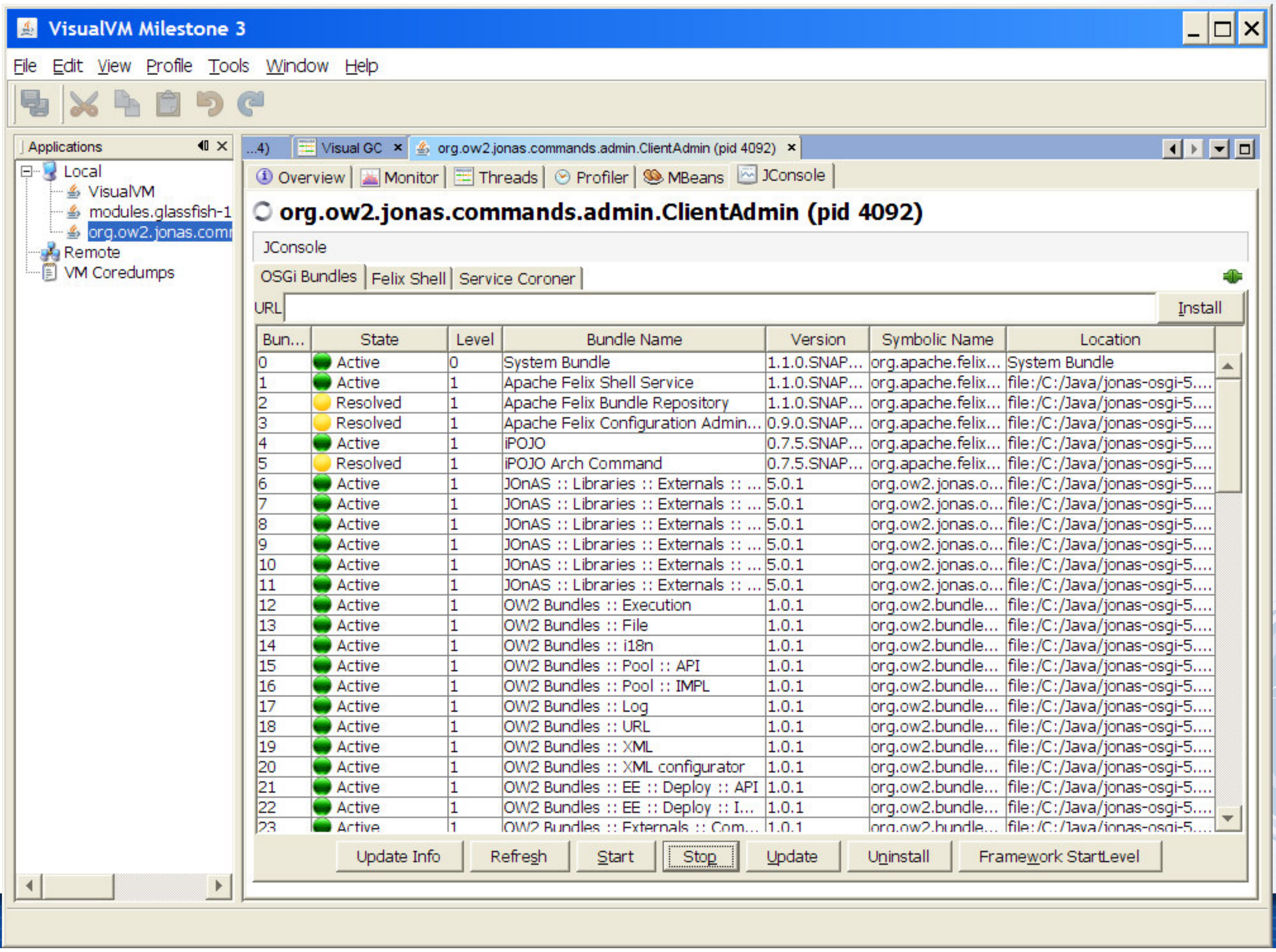
# OSGi Alliance Community Event

## Bonus Track

## Metrics

- ServiceCoroner
  - (Felix.jar 330KB)
  - Core + MBean : 48KB
    - Number of ligne of code: 1615 in Java, 79 in AspectJ
    - Number of classes: 37
    - Number of pointcuts (AspectJ): 5
  - Swing GUI : 53KB
    - Number of ligne of code: 1067 in Java, 97 in JavaScript
    - Number of classes: 43
- ServiceCoronerPlugin (JConsole & VisualVM)
  - Jar: 123 KB
  - 4 classes and 254 LoC

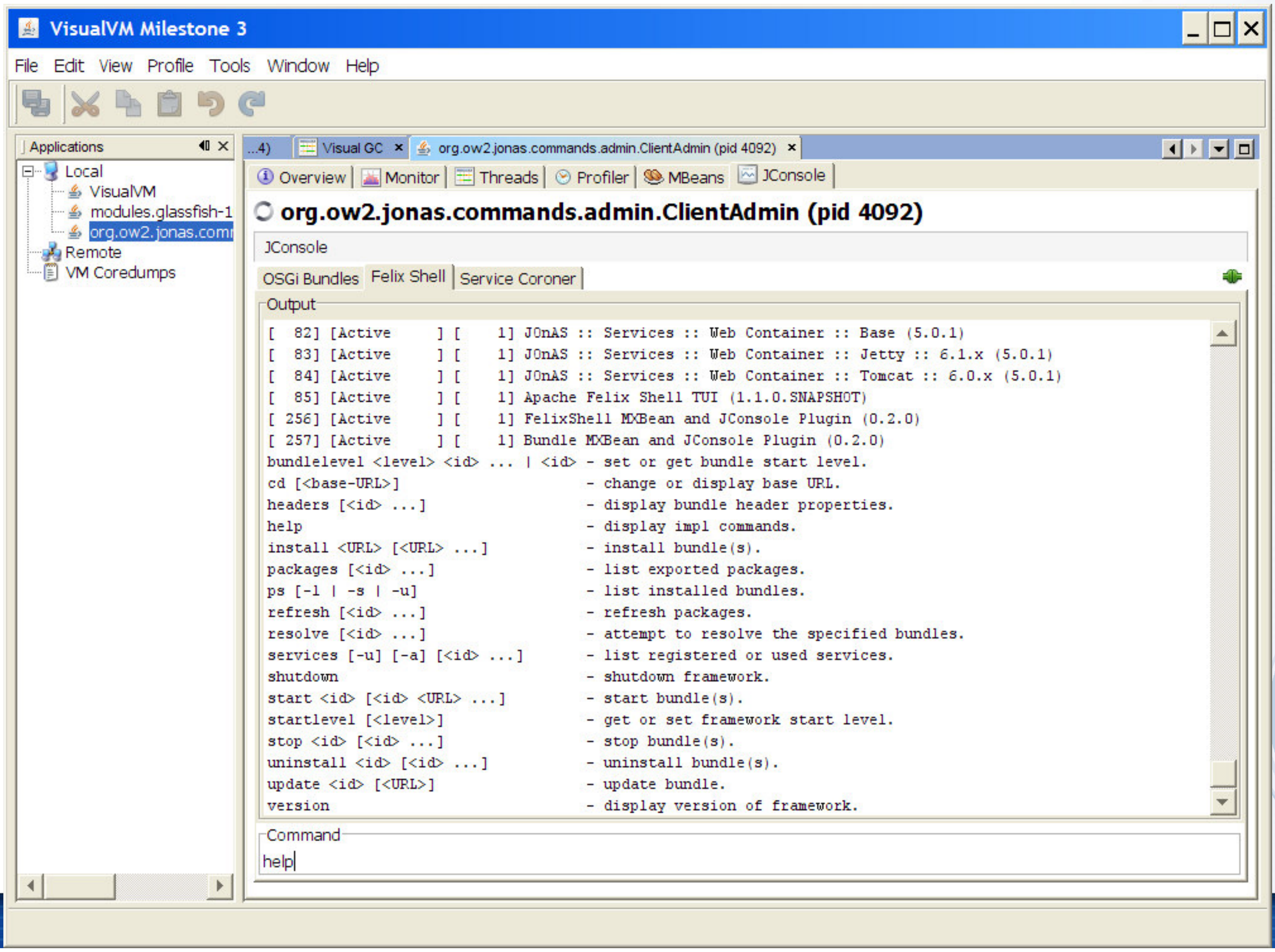
## MBeans & JConsole/Visual plugins
## OSGi console



org.ow2.jonas.commands.admin.ClientAdmin (pid 4092)

| Bun... | State | Level | Bundle Name | Version | Symbolic Name | Location |
|---|---|---|---|---|---|---|
| 0 | Active | 0 | System Bundle | 1.1.0.SNAP... | org.apache.felix... | System Bundle |
| 1 | Active | 1 | Apache Felix Shell Service | 1.1.0.SNAP... | org.apache.felix... | file:/C:/Java/jonas-osgi-5.... |
| 2 | Resolved | 1 | Apache Felix Bundle Repository | 1.1.0.SNAP... | org.apache.felix... | file:/C:/Java/jonas-osgi-5.... |
| 3 | Resolved | 1 | Apache Felix Configuration Admin... | 0.9.0.SNAP... | org.apache.felix... | file:/C:/Java/jonas-osgi-5.... |
| 4 | Active | 1 | iPOJO | 0.7.5.SNAP... | org.apache.felix... | file:/C:/Java/jonas-osgi-5.... |
| 5 | Resolved | 1 | iPOJO Arch Command | 0.7.5.SNAP... | org.apache.felix... | file:/C:/Java/jonas-osgi-5.... |
| 6 | Active | 1 | JOnAS :: Libraries :: Externals :: ... | 5.0.1 | org.ow2.jonas.o... | file:/C:/Java/jonas-osgi-5.... |
| 7 | Active | 1 | JOnAS :: Libraries :: Externals :: ... | 5.0.1 | org.ow2.jonas.o... | file:/C:/Java/jonas-osgi-5.... |
| 8 | Active | 1 | JOnAS :: Libraries :: Externals :: ... | 5.0.1 | org.ow2.jonas.o... | file:/C:/Java/jonas-osgi-5.... |
| 9 | Active | 1 | JOnAS :: Libraries :: Externals :: ... | 5.0.1 | org.ow2.jonas.o... | file:/C:/Java/jonas-osgi-5.... |
| 10 | Active | 1 | JOnAS :: Libraries :: Externals :: ... | 5.0.1 | org.ow2.jonas.o... | file:/C:/Java/jonas-osgi-5.... |
| 11 | Active | 1 | JOnAS :: Libraries :: Externals :: ... | 5.0.1 | org.ow2.jonas.o... | file:/C:/Java/jonas-osgi-5.... |
| 12 | Active | 1 | OW2 Bundles :: Execution | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 13 | Active | 1 | OW2 Bundles :: File | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 14 | Active | 1 | OW2 Bundles :: i18n | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 15 | Active | 1 | OW2 Bundles :: Pool :: API | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 16 | Active | 1 | OW2 Bundles :: Pool :: IMPL | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 17 | Active | 1 | OW2 Bundles :: Log | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 18 | Active | 1 | OW2 Bundles :: URL | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 19 | Active | 1 | OW2 Bundles :: XML | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 20 | Active | 1 | OW2 Bundles :: XML configurator | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 21 | Active | 1 | OW2 Bundles :: EE :: Deploy :: API | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 22 | Active | 1 | OW2 Bundles :: EE :: Deploy :: I... | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |
| 23 | Active | 1 | OW2 Bundles :: Externals :: Com... | 1.0.1 | org.ow2.bundle... | file:/C:/Java/jonas-osgi-5.... |

## MBeans & JConsole/Visual plugins Shell (for Felix, Equinox, KF)



```
[  82] [Active     ] [    1] JOnAS :: Services :: Web Container :: Base (5.0.1)
[  83] [Active     ] [    1] JOnAS :: Services :: Web Container :: Jetty :: 6.1.x (5.0.1)
[  84] [Active     ] [    1] JOnAS :: Services :: Web Container :: Tomcat :: 6.0.x (5.0.1)
[  85] [Active     ] [    1] Apache Felix Shell TUI (1.1.0.SNAPSHOT)
[ 256] [Active     ] [    1] FelixShell MXBean and JConsole Plugin (0.2.0)
[ 257] [Active     ] [    1] Bundle MXBean and JConsole Plugin (0.2.0)
bundlelevel <level> <id> ... | <id> - set or get bundle start level.
cd [<base-URL>]                     - change or display base URL.
headers [<id> ...]                  - display bundle header properties.
help                                - display impl commands.
install <URL> [<URL> ...]           - install bundle(s).
packages [<id> ...]                 - list exported packages.
ps [-l | -s | -u]                   - list installed bundles.
refresh [<id> ...]                  - refresh packages.
resolve [<id> ...]                  - attempt to resolve the specified bundles.
services [-u] [-a] [<id> ...]       - list registered or used services.
shutdown                            - shutdown framework.
start <id> [<id> <URL> ...]         - start bundle(s).
startlevel [<level>]                - get or set framework start level.
stop <id> [<id> ...]                - stop bundle(s).
uninstall <id> [<id> ...]           - uninstall bundle(s).
update <id> [<URL>]                 - update bundle.
version                             - display version of framework.
```

```
help
```