The New Profiling Tools in the Oracle JDK!

Klara Ward
Java Mission Control Dev Team

@klaraward
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
“It's an invaluable piece of work.”

“I managed to do in one day what I've failed to do in 2+ weeks using <profiling tool> and <another profiling tool>.”

Maurizio Cimadamore
Ex Lang Tools, Oracle
Agenda

- Introduction to Mission Control/Overview
- Startup
- Tools Walk Through With Demos
  - JMX Console
  - Java Flight Recorder
  - (Customization)
- Questions
JAVA MISSION CONTROL
OVERVIEW
Java Mission Control Overview

A new JDK profiling and diagnostics tools platform

- A tools suite for *production* use (fine in development too)
  - Basic monitoring
  - Production time profiling and diagnostics

- Focus on low overhead
  - Built into the JVM
  - Already available runtime information
History of Mission Control

Very brief

- Started with the very first JVM management console, in JRockit
- ... some additional tools and acquisitions later...
- Merged JRockit features into HotSpot
- Java Mission Control now released with 7u40, and 8!
Java Mission Control Main Tools

The tools available in Java Mission Control 5.3.0

Two main tools:

- JMX Console
- Flight Recorder
  - Low overhead profiler
    - Built into the JVM
JMC INSTALLATION AND STARTUP
JMC installation

- JMC stand alone RCP client in JDK (>=7u40)
- Eclipse plug-ins
  - Install from update site on OTN: 
    [http://download.oracle.com/technology/products/missioncontrol/updatesites/base/5.3.0/eclipse/](http://download.oracle.com/technology/products/missioncontrol/updatesites/base/5.3.0/eclipse/)
- Downloadable experimental plug-ins
  - Install from within RCP, or from 
    [http://download.oracle.com/technology/products/missioncontrol/updatesites/experimental/5.3.0/eclipse/](http://download.oracle.com/technology/products/missioncontrol/updatesites/experimental/5.3.0/eclipse/)
  - JOverflow (hprof heap analyzer, quantifies heap waste)
  - Weblogic Server integration for Flight Recorder
  - ...


Starting the JMC client

- Just double-click on the launcher in bin

- Starting from the command line with regular Eclipse launcher flags
  
  jmc -consoleLog -debug 2>&1 | more

- More info on http://hirt.se/blog/?p=281
TOOLS
WALK-THROUGH
The JMX Console

Highlights

- Real time monitoring
- Triggers
  - Rules + Actions
- Diagnostic commands
  - (same as JCMD)
- Various experimental plug-ins
  - Coherence
  - JConsole plug-in support
  - Twitter ;)

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.
JMX CONSOLE DEMO
(IN JUST A LITTLE WHILE)
FLIGHT RECORDER OVERVIEW
Flight Recorder

101

- Event Recorder
  - Designed for high performance
  - Thread local data buffers to minimize contention

- Binary recordings

- Chunks
  - Self contained
  - Self describing
Different Kinds of Events

- **Instant Event** - *Exception*
  - Data associated with the time the data was captured

- **Requestable Event** – *Method profiling sample*
  - Polled from separate thread
  - Has a user configurable period

- **Duration Event** – *Thread.sleep*
  - Has a start time and a stop time
  - Timed Event – also has configurable threshold
How to Think About Recordings

- **Recording**
  - A named collection of event type settings – “Profile”
  - Active in the JVM for a certain period of time
    - **Time fixed** – Automatically displayed if started from JMC
    - **Continuous** – Needs to be explicitly dumped

- The resulting JFR file is sometimes also referred to as a Recording
How to Think About Recordings

What data does Louie get when he dumps a recording?

Huey starts long **continuous** JFR

Dewey starts short **profile** JFR

Louie dumps part of **continuous** JFR

Expect:

Gets:

# event types

Time
CREATING RECORDINGS
Preparations

Hotspot

- Need to have started the JVM from which to get recording with the appropriate flags (for now)

-XX:+UnlockCommercialFeatures  -XX:+FlightRecorder
Creating Recordings

More than one way

- **Mission Control**
  - Start Flight Recording... wizard from the JVM Browser

- **Startup flags**
  - `-XX:+UnlockCommercialFeatures -XX:+FlightRecorder -XX:StartFlightRecording=delay=20s,duration=60s,name=MyRecording,filename=C:\TEMP\myrecording.jfr,settings=profile`

- **JCMD**
  - `jcmd 7060 JFR.start name=MyRecording settings=profile delay=20s duration=60s filename=c:\TEMP\myrecording.jfr`

For more information, go to [hirt.se/blog?p=370](http://hirt.se/blog?p=370)
Some Common Pitfalls

When analyzing Flight Recordings

- Not accounting for CPU load
  - Don’t make decisions based on method profiling data if there is no load
  - If you have full load, then looking at latencies may be a waste of time

- Not accounting for thresholds
  - Thresholds are very useful for keeping performance up but still detecting outliers
  - Can be confusing. Example:
    Thread T has been running for 2 minutes and sum of latencies is a minute.
    Was the thread T running unblocked for a minute?
CUSTOMIZATION
Adding Your Own Events (unsupported)

```java
import com.oracle.jrockit.jfr.*;

public class Example {
    private final static String PRODUCER_URI = "http://www.example.com/demo/";
    private Producer myProducer;
    private EventToken myToken;

    public Example() throws URISyntaxException, InvalidEventDefinitionException, InvalidValueException {
        myProducer = new Producer("Demo Producer", "A demo event producer.", PRODUCER_URI);
        myToken = myProducer.addEvent(MyEvent.class);
    }

    @EventDefinition(path="demo/myevent", name = "My Event", description="An event triggered by doStuff.", stacktrace=true, thread=true)
    private class MyEvent extends TimedEvent {
        @ValueDefinition(name="Message", description="The logged important stuff.")
        private String text;
        public MyEvent(EventToken eventToken) {
            super(eventToken);
        }

        public void setText(String text) {
            this.text = text;
        }

        public void doStuff() {
            MyEvent event = new MyEvent(myToken);
            event.begin();
            String importantResultInStuff = "";
            // Generate the string, then set it...
            event.setText(importantResultInStuff);
            event.end();
            event.commit();
        }
    }
}
```
Built in GUI editor (unsupported)

- The JMC has a built in designer
- Can be used to both customize the existing GUI and produce entirely new GUIs for events
- The created GUIs can be exported as plug-ins and shared
FINALLY, SOME DEMOS!
What’s next

- JFR auto analysis
- Updated JFR API
- Online heap analyzer
- ...

...
Resources

Homepage:  
http://oracle.com/missioncontrol

Twitter:  
@javamissionctrl  
@klaraward  
@hirt

Blog:  
http://hirt.se/blog?cat=3

Facebook (why not):  
Evaluate This Session

1. Sign-in: [www.eclipsecon.org](http://www.eclipsecon.org)

2. Select session from schedule

3. Evaluate: [+] [0] [-]
Questions?

Psst - We’re Hiring!

Don’t forget!

-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder

Evaluate!

+1          0          -1

@javamissionctrl
oracle.com/missioncontrol

@hirt
http://hirt.se/blog?cat=3

@klaraward
klara.ward@oracle.com

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.