



redhat.

# Profiler, der bessere Debugger?

Heiko W. Rupp  
<[heiko.rupp@redhat.com](mailto:heiko.rupp@redhat.com)>

Java Forum Stuttgart 2008

# Agenda

- Einleitung
- Was ist ein Profiler, Abgrenzung zum Debugger
- Warum Profiler als Debugger?
- Beispiel 3-Tier-Anwendung
- Demo: „Was passiert wenn ich hier clicke?“
- Demo: „Warum ist die Anwendung so langsam?“
- Fazit

# Einleitung

- Vortrag als Denkanstoss !
- Fragen bei der Softwareentwicklung
  - „Wo soll ich anfangen?“
  - „Was passiert wenn ich da klicke?“
- Einfach bei kleinen Anwendungen
  - Aber was macht das Folgende wirklich:

```
Person x = new Person(...);  
entityManager.persist(x);
```

- Hier wurden schon oft Überraschungen erlebt

# Debugger

- Werkzeug zum
  - Durchschreiten von Code
  - Analyse von Variableninhalten
- Applikation wird angehalten
  - Breakpoint
  - Exception
  - Bedingung
- Bei größeren Anwendungen Probleme mit Transaktionen

# Profiler

- Werkzeug zur Analyse von
  - Laufzeit
  - Speicherverhalten
  - ...
- Anwendung läuft während des Profilings
- Produkte (kommerziell und frei)
  - JProfiler / JProbe / ...
  - NetBeans / Eclipse TPTP / ...
  - ...

# NetBeans Profiler

NetBeans IDE 6.1

XsdParser

Start Page x XsdParser.java x CPU: 06:08:52 PM x

View: Methods

**Call Tree - Method**

Call Tree - Method	Time [%]	Time	Invocations
All threads		3548 ms (100%)	1
main		3548 ms (100%)	1
de.bsd.x2svg.X2Svg.main (String[])		3548 ms (100%)	1
de.bsd.x2svg.Runner.run (de.bsd.x2svg.X2Svg)	99,1%	3514 ms	1
de.bsd.x2svg.Runner.setupSvg ()	5,1%	1840 ms	1
Self time	1,6%	569 ms	1
de.bsd.x2svg.parsers.XsdParser.parse (String)	14,6%	516 ms	1
de.bsd.x2svg.RuntimeProperties.load ()	8,7%	308 ms	1
de.bsd.x2svg.Runner.drawContainer (int)	6,3%	222 ms	1
de.bsd.x2svg.ParserLoader.load ()	1,2%	42.9 ms	1
de.bsd.x2svg.Runner.computeTreeSize (int)	0,1%	4.13 ms	1
de.bsd.x2svg.ParserLoader.getParser ()	0%	0.130 ms	1
de.bsd.x2svg.util.IOUtil.close (java.io.InputStream)	0%	0.113 ms	1
de.bsd.x2svg.parsers.XsdParser.setParser (ParserLoader)	0%	0.020 ms	1
de.bsd.x2svg.outputConverter.SvgConverter.convert (String)	0%	0.015 ms	1
de.bsd.x2svg.RuntimeParameters.getParser ()	0%	0.003 ms	1
de.bsd.x2svg.ParserLoader.getLoader ()	0%	0.002 ms	1
de.bsd.x2svg.RuntimeParameters.getParser ()	0%	0.002 ms	1
de.bsd.x2svg.Runner.<init> ()	0,4%	15.0 ms	1
de.bsd.x2svg.X2Svg.parseCommandline (String[] args)	0,4%	13.3 ms	1
Self time	0,1%	4.93 ms	1
de.bsd.x2svg.X2Svg.<init> ()	0%	0.015 ms	1
de.bsd.x2svg.Runner.<clinit> ()	0%	0.013 ms	1

**Controls**

**Status**

Type: CPU  
 Configuration: Analyze Performance  
 Status: Inactive

**Profiling Results**

Take Snapshot Live Results

Reset Collected Results

**Saved Snapshots**

x2svg

Open

**Navigator**

Members View

X2Svg

- handleOutputConversion(RuntimeParameters p, String)
- main(String[] args)
- parseCommandline(String[] args) : RuntimeParameters
- usage()

## Warum aber nun ein Profiler zum Debuggen?

- Start einer Kette von Aufrufen ist nicht bekannt
- Aufrufe dynamisch
  - Reflection
- Komplexe Frameworks
  - Sehr tiefer Stack
- Transaktionstimeouts beim Debuggen

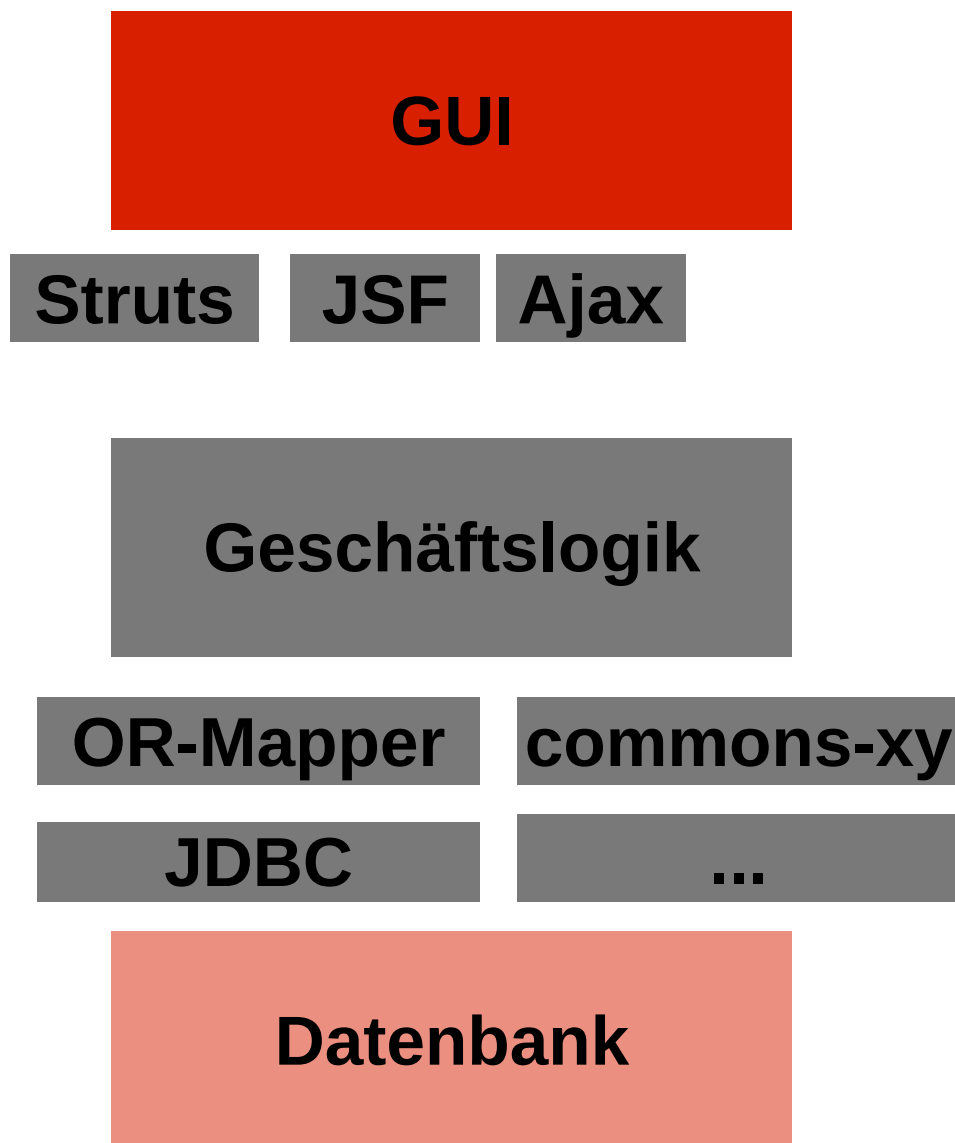
## Beispiel: 3-Schichten-Architektur (Theorie)

- Drei Klare Schichten
- Aufrufe eindeutig von oben nach unten





## Beispiel: 3-Schichten-Architektur (Realität)



## Beispiel: Was passiert hier?

- Beispiele aus RHQ (<http://www.rhq-project.org/>)
- Was passiert wenn man hier clickt ?

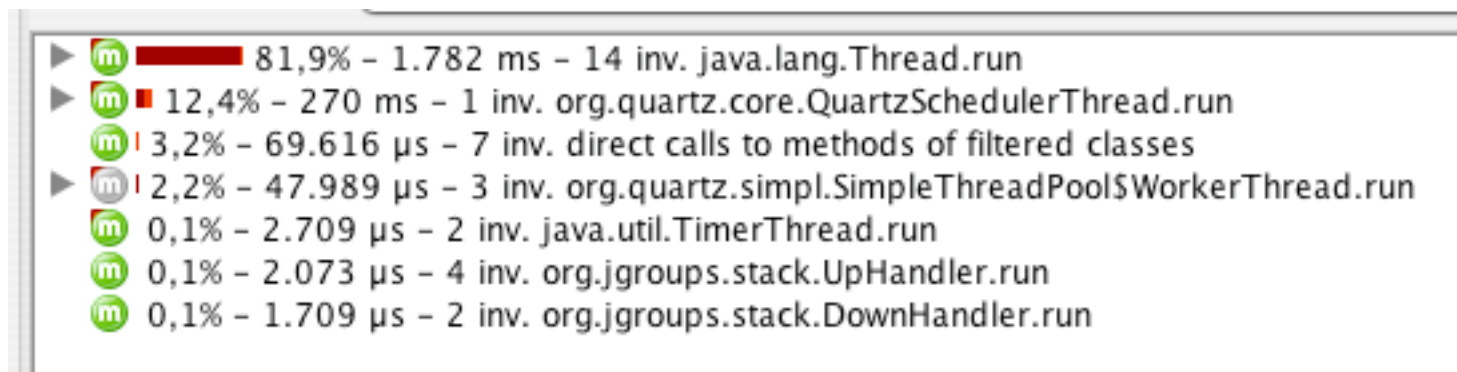
RESOURCES			
Resource Name	Total	Avail	Action
snert	1	✓	▶
File System	4	✓	▶
CPU	2	✓	▶
HttpCheck	1	✓	▶
<b>JBossAS Server</b>		⚠	▶
snert RHQ Agent	1	✓	▶
Network Adapter	6	⚠	▶

- Von Hand durch den Code wühlen ist umständlich



## Vorgehen (1)

- Applikation starten
- Profiler starten
  - Start CPU-Auslastung messen
  - Applikation bedienen
  - Ende CPU-Auslastung messen
- Start Analyse
  - Zeitbaum

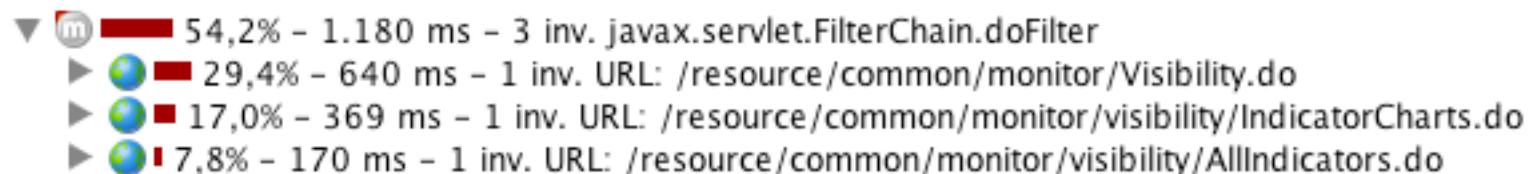


## Vorgehen (2)

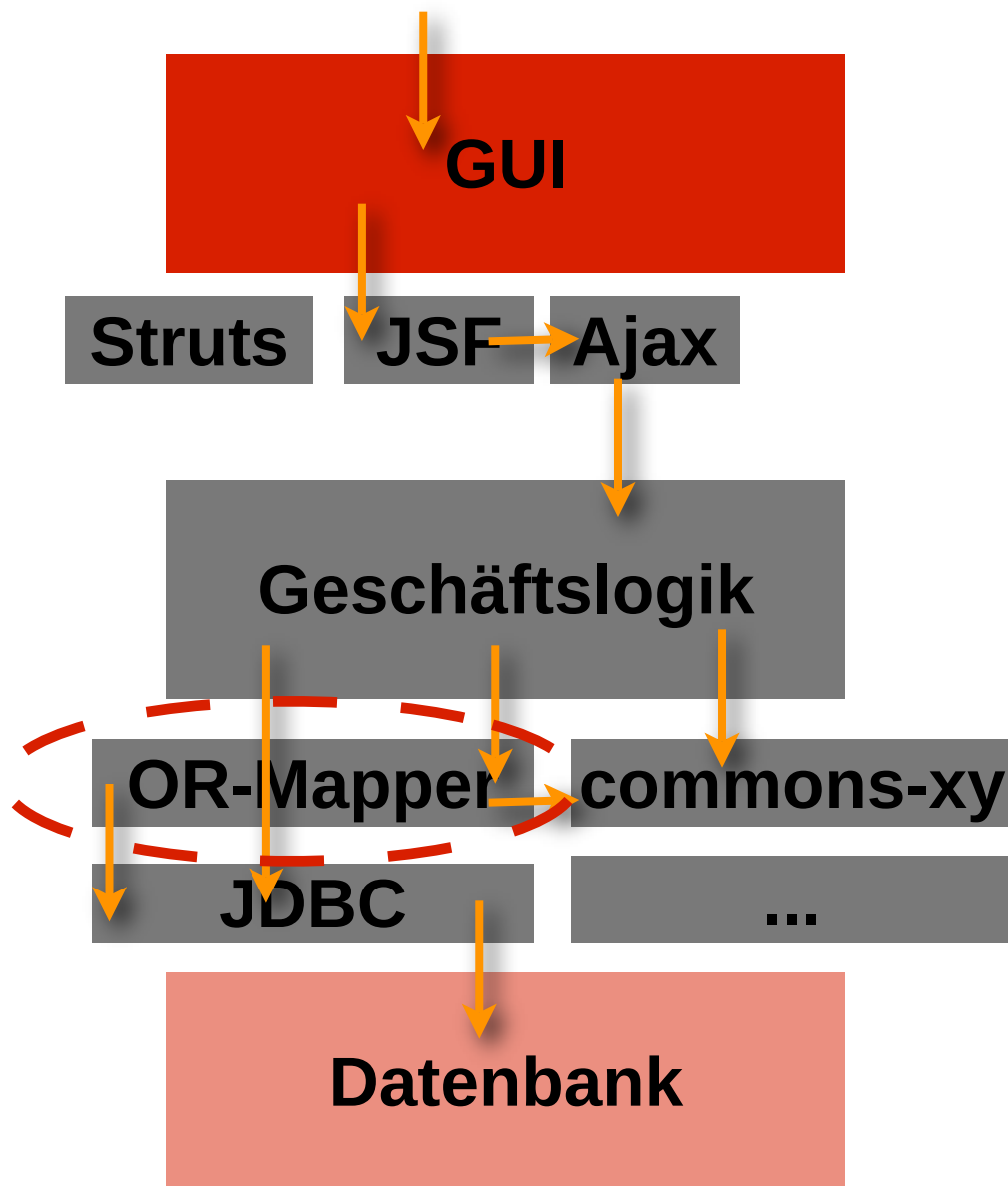
- Baum weiter aufklappen



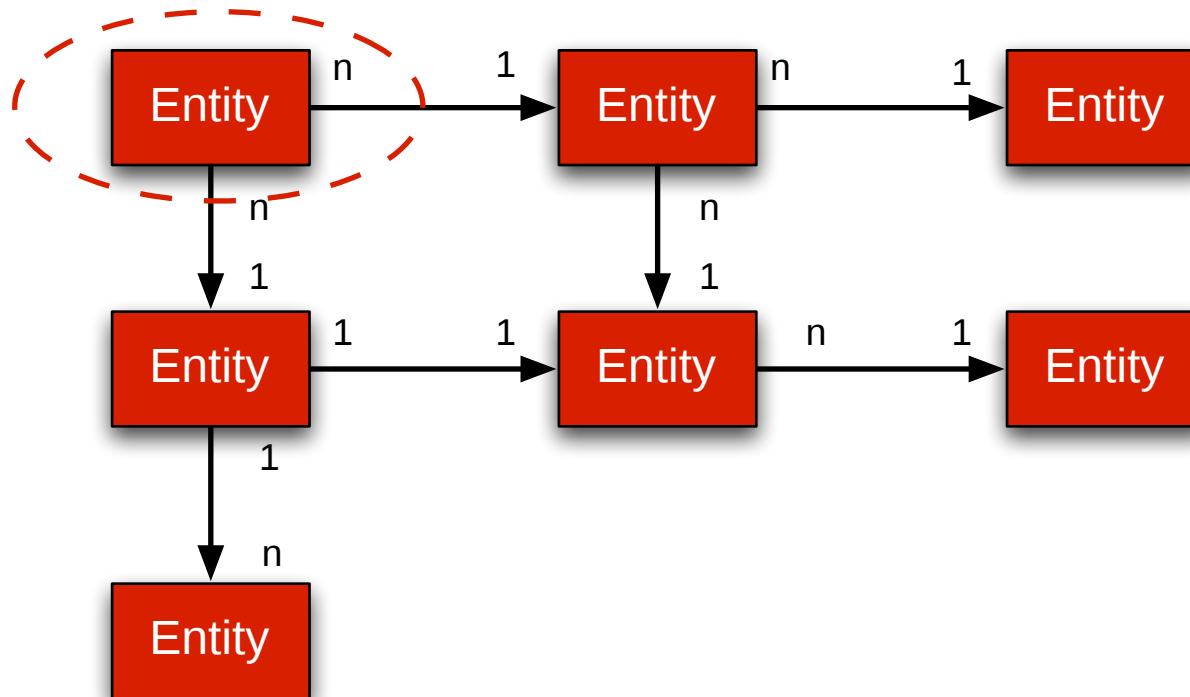
- Entscheiden wie weiter



# Warum ist die Applikation so langsam?



# Komplexes O/R-Modell



- Was passiert bei „SELECT Entity1 WHERE id=1“ ?

## Demo

- Prinzip wie oben
- Durchschreiten der Schichten von Hand fast unmöglich
- Anzeige der JDBC-Aktivität

## Fazit

- Pro
  - Profiler kann einfacher durch unbekanntem Code pflügen
  - Profiler kann Innenleben komplexer Frameworks zeigen
- Contra
  - Profiler kann keine Variablen zeigen
- Profiler ist eine Ergänzung, kein Ersatz !



- Danke fürs Zuhören
- Noch Fragen?
- ==> Red Hat wants you!
  - Wir suchen Java-Experten
  - Meine E-Mail steht auf der ersten Folie