

- Om SigmaT
- Eksempel på indlejring af Groovy
- Overvågning af PEM
- Ønske om dynamisk loaded "Java" uden at fikle med classloaderen
- Groovy til hjælp
- Opsamling – hvad jeg ikke har fortalt om

- Trym Reinholdt Møller, SigmaT
- Freelance J2EE konsulent
- 36 år
- Har to børn og bor i Risskov
- Har tidligere arbejdet ved Creuna, Terma og Acure, An IBM Division

- <http://sigmat.dk>



GroovyRunner i Eclipse

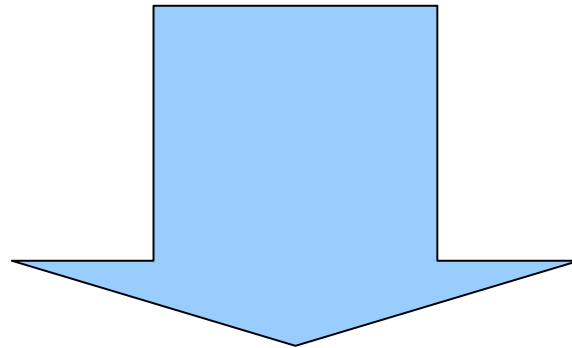
- Personlig Elektronisk Medicinprofil (PEM)
- Integrerer til 250 kørende apotekssystemer, der hvert minut henter informationer eller opdaterer disse
- Integrerer til samtlige lægehuse, der sender elektroniske recepter
- Krav om høj opetid samt stor robusthed
- Dvs. kvartalsvise deploys
- MEN løbende fejlfinding i data og i kommunikationen med integrationssystemerne

- Krav om reaktionstid indenfor 15 minutter ved problemer
- Ønske om at kende problemerne før kunden
- ITD har 24x7 overvågning, der ringer til PEM-vagten
- ITD overvåger specifik log fil på produktionsmiljøet ved hjælp af Tivoli software
- PEM applikationen logger følgende til den overvågede log fil:
 - Hardcoded kendte problemer
 - Et filtreret udsnit af "uforudsete" logninger fra log4j

- PEM er en J2EE applikation, der afvikles på en Trifork applikationsserver med Oracle som database
- Der anvendes log4j frameworket til logging

- Filtreringen var hardcoded og inkluderede alle logninger, der stammede fra "java.sql.SQLException"
- Dette viste sig for
 - Grovkornet, da vi ikke behøvede at håndtere alle sql-exceptions manuelt
 - Ufleksibelt, når vi ønskede at ændre filtreringen og ikke lige kunne deploye en ny version af vores applikation

- Muligt at kunne konfigurere filtreringen dynamisk (ligesom log4j)
- Muligt, uden at lave et nyt xml-sprog, at være præcis angående den ønskede filtrering



- Dvs. vi ville gerne dynamisk/runtime kunne loade en scriptbar filtrering

```
<root>
  <priority value ="info" />
  <appender-ref ref="DEPENDENT_MONITORED" />
  ...
</root>

<appender name="DEPENDENT_MONITORED"
  class="dk.acure.pem.util.log4j.EventDependentAppender">
  <param name="IsGroovy" value="ToMonitoredLog"/>
  <appender-ref ref="ASYNC_MONITORED" />
</appender>

...
```

```
public class EventDependentAppender extends AppenderSkeleton
    implements AppenderAttachable {

    ...
    public void append(LoggingEvent event) {
        synchronized(aai) {
            if (toAppenders(event)) {
                Enumeration e = getAllAppenders();
                while (e.hasMoreElements()) {
                    Appender appender = (Appender)e.nextElement();
                    appender.doAppend(event);
                }
            }
        }
    }

    public void setIsgroovy(String groovyScriptName) { ... }
}
```

"Hardcoded" java filtering (*.java)

ΣΤ

```
package dk.acure.pem.services.v2.util;

import org.apache.log4j.spi.LoggingEvent;...

public class ToMonitoredLog implements
    EventDependentAppenderDecideInterface {
    private LogTargetHelper lth = new LogTargetHelper();

    public boolean toAppenders(LoggingEvent event) {
        if (event.getThrowableInformation() != null) {
            Throwable t = event.getThrowableInformation().getThrowable();
            if (t != null) {
                return lth.isThrowableType(t, java.sql.SQLException.class);
            }
        }
        return false;
    }
}
```

Groovy filtering (*.groovy)

```
import org.apache.log4j.spi.LoggingEvent;
...
public class ToMonitoredLog implements
    EventDependentAppenderDecideInterface {
    private LogTargetHelper lth = new LogTargetHelper();

    public boolean toAppenders(LoggingEvent event) {
        if (event.getThrowableInformation() != null) {
            Throwable t = event.getThrowableInformation().getThrowable();
            if (t != null) {
                return lth.isThrowableType(t, java.sql.SQLException.class);
            }
        }

        return false;
    }
}
```



```
private LogTargetHelper lth = new LogTargetHelper();

public boolean toAppenders(LoggingEvent event) {
    if (event.getThrowableInformation() != null) {
        Throwable t = event.getThrowableInformation().getThrowable();
        if (t != null) {
            return ((lth.isThrowableType(t, java.sql.SQLException.class))
                && !((t.getMessage().indexOf("Closed Connection") > -1)
                && (containsStr("CLOB.length",
                    event.getThrowableInformation().getThrowableStrRep()))));
        }
    }

    return false;
}
```



```
// groovyRoots is the folder containing groovy script files
GroovyScriptEngine myGse = new GroovyScriptEngine(getGroovyRoot());

// load the current instance of the groovy file as a java class
Class myJavaFromGroovyClass = myGse.loadScriptByName(getScriptName());

// create instance of a non-jvm classloaded class.
Object groovyInstance = myJavaFromGroovyClass.newInstance();

// Cast and use the instance
EventDependentAppenderDecideInterface edadiGroovy =
(EventDependentAppenderDecideInterface)groovyInstance;
edadiGroovy.toAppenders(event);
```

- Ønske om dynamisk load af "Java" lignende script, kan løftes af Groovy
- Ikke fortalt om:
 - Performance/caching af loaded script
 - Unittest
 - Anvendelse af et dynamisk sprogs muligheder
 - Sikkerhed ved dynamisk loaded kode
 - Scripts i klar tekst – adgangskontrol til folderen
 - Versionsstyring, nemt at ændre uden deploy
 - Flere versioner af samme klasse loaded i "class-loaderen"/applikationen.
 - Andre måder at integrere Groovy og Java

- <http://groovy.codehaus.org/> - Groovy home
- <http://groovy.codehaus.org/Embedding+Groovy>
- <http://www.ibm.com/developerworks/java/library/j-alj08034.html>

Groovy eksempel fra Eclipse

ΣΤ

```
package dk.sigmat.test.groovy;

import groovy.util.GroovyScriptEngine;

public class GroovyRunner {

    public static void main(String[] args) {
        GroovyRunner runner = new GroovyRunner();
        try {
            runner.run();
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }

    @SuppressWarnings("unchecked")
    private void run() throws Throwable {
        // groovyRoots is the folder containing groovy script files
        GroovyScriptEngine myGse = new GroovyScriptEngine(getGroovyRoot()); // IOException

        // load the current instance of the groovy file as a java class
        Class myJavaFromGroovyClass = myGse.loadScriptByName(getScriptName()); // ResourceException,
ScriptException

        // create instance of a non-jvm classloaded class.
        Object groovyInstance = myJavaFromGroovyClass.newInstance(); // InstantiationException,
IllegalAccessException

        // Cast to a java interface or super class and use the instance
        Runnable HelloWorldGroovy = (Runnable)groovyInstance;
        HelloWorldGroovy.run();
    }

    private String getScriptName() {
        return "HelloWorld";
    }
}
```