mazcity Consulting Design Development

Let's do IT better!
www.mazcity.de +49(0)175-24 22 4 55

Enterprise Application Monitoring
- a Framework -

09.12.2011
Presentation for anderScore Goldschmiede
Cologne

About Author

mazcity Consulting Design Development
Let's do IT better!

- Maz Rashid

- Freelancer
- Senior Java Architect
- Certified Project Manager (IPMA-Level C, PMP)

- Devoted to Object Oriented Software Design
- Working in long term projects
- high performance, high concurrency environment

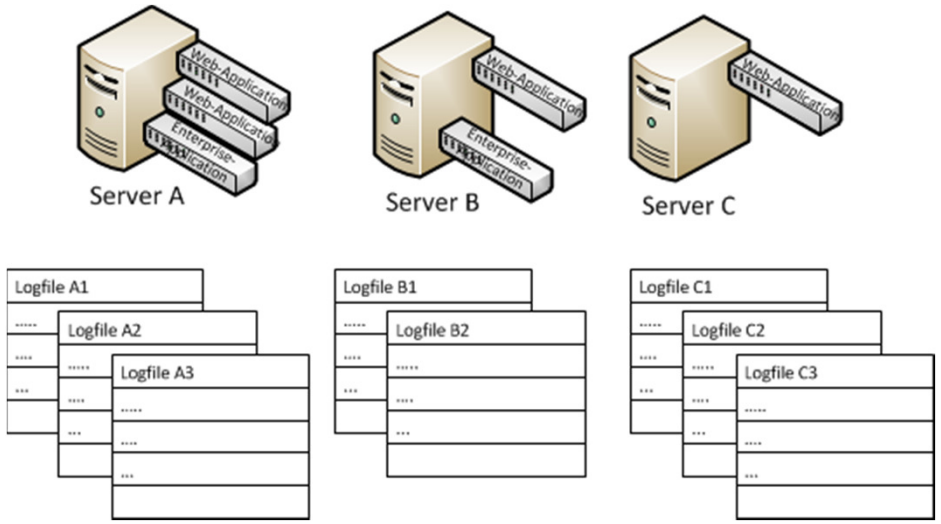
- Finance, Telecom, Logistics, Energy Trading, Workforce Management

- More on http://www.mazcity.de

Motivation

mazcity Consulting Design Development
Let‘s do IT better!

- Stage of the Art Enterprise Applications consist of a series of Applicationservers having a number of WebApps or EARs.



- Every WebApp and EAR produces it‘s own log files
- Commonly log4J is used for logging
- It is hard to monitor all files and have a good overal monitoring.

Goals

mazcity Consulting Design Development
Let‘s do IT better!

- Have an overview if the whole enterprise application is doing well
- Display issues on one screen
- Giving an overview of Warnings and Errors
- Easy Use / Use everywhere

Solution A – Email Notification

mazcity Consulting Design Development
Let‘s do IT better!

- Use log4j Email-Appender and send Emails when Errors occur
- Advantage:
  - Easy
  - „Everybody“ in corporate environment is reachable via Email
- Disadvantage:
  - Annoying people by two many False-Positive (means errors indicated, that are not really an error case)
  - Suppressing „False-Positive“ will maybe lead to suppressing serious Issues.
  - Annoying people because of technical Problems leading in 100s of Emails.

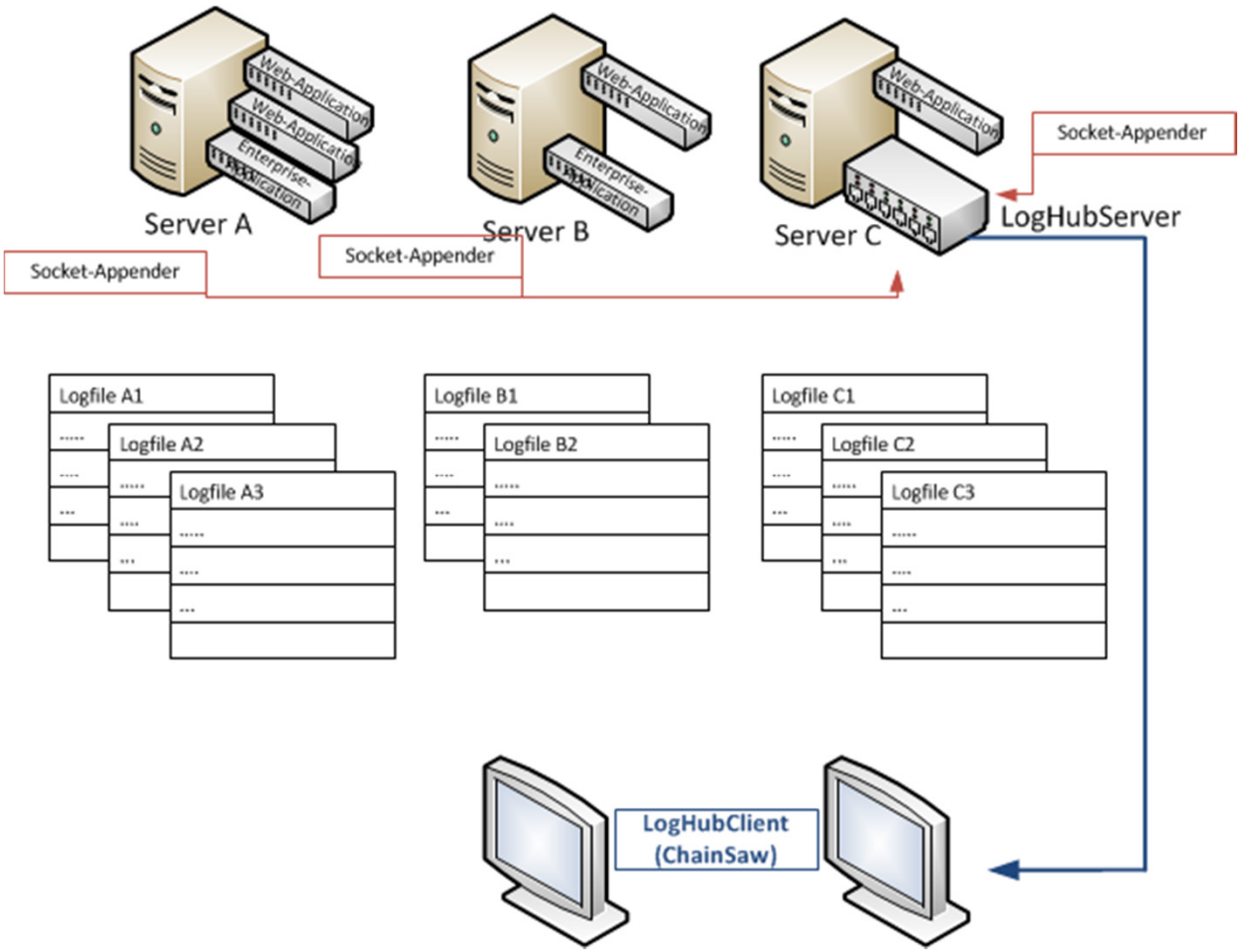
Solution of Choice I

mazcity Consulting Design Development
Let‘s do IT better!

- Every WebApp/EAR:
  - Add an addition SocketAppender sending to LogHubServer
  - Set Level to WARN (so only Warnings / Errors are transfered)
- LogHubServer:
  - A very small Application or WebApplication
  - Serving as a hub
  - Buffersize configurable
  - Will start a SocketReceiver and buffer all delivered logs
  - Clients (ChainSaw) will be served via SocketHubAppender
- LogHubClient (ChainSaw from Log4j):
  - Connects via SocketHubReceiver
  - Will get the buffer from LogHubServer

Solution of Choice II

mazcity Consulting Design Development
Let‘s do IT better!



Configuration - WebApps

mazcity Consulting Design Development
Let‘s do IT better!

- Sample appender definition in log4j.properties

```
log4j.appender.loghub=org.apache.log4j.net.SocketAppender
log4j.appender.loghub.remoteHost=${loghub.host}
log4j.appender.loghub.port=${loghub.port}
log4j.appender.loghub.Threshold=Warn
log4j.appender.loghub.application=sample-application-name
```

- Just add the appender to the loggers
- Host/Port of LogHubServer are defined as System-Parameters

LogHubServer

mazcity Consulting Design Development
Let's do IT better!

```
import org.apache.log4j.LogManager;
import org.apache.log4j.PropertyConfigurator;
import org.apache.log4j.net.SocketReceiver;
import org.apache.log4j.xml.DOMConfigurator;
import org.springframework.stereotype.Service;

@Service
public class LogHubServerBean {
    private static final org.apache.log4j.Logger logger = org.apache.log4j.LogManager.ge

    private static String envName;
    public static String getEnvName() {return envName;}
    public void setEnvName(String name) {envName = name;}

    private int port;
    public int getPort() {return port;}
    public void setPort(int port) {this.port = port;}



    public void init()
    {
/*      logger.info("reset log4j config....");
        LogManager.resetConfiguration();
        DOMConfigurator.configure("config/log4j.xml");
        logger.info("reset log4j done.");
    */
        logger.info(" start receiver port: " + port + ", env: " + envName + " ...");
        if(port==0)
            throw new IllegalArgumentException("Port may not be 0");
        SocketReceiver rec = new SocketReceiver(port,logger.getLoggerRepository());
        rec.activateOptions();
        logger.info("started!");

    }
}
```

LogHubServer Filter

- Add additional data to every Log-Entry-Object

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

public class LogHubAppEnvSetterFilter extends Filter {

    @Override
    public int decide(LoggingEvent e) {

        // set always the app.env variable in the event
        String envName = LogHubServerBean.getEnvName();
        if(envName!=null)
            e.setProperty("app.env", envName);

        return ACCEPT;
    }

}
```

LogHubServer Config

mazcity Consulting Design Development
Let‘s do IT better!

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
  <log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>
  <appender name="Console" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="SRV: %-5p [%C:%M:%L] %X{application} %X{app.env} %m%n" />
    </layout>
    <filter class="net.amazers.EAM.LogHubAppEnvSetterFilter"/>
  </appender>
  <appender name="Chainsaw" class="org.apache.log4j.net.SocketHubAppender">
    <param name="Port" value="${loghub.serverport}"/>
    <!--<param name="Port" value="4580"/>-->
    <param name="BufferSize" value="300"/>
    <filter class="net.amazers.EAM.LogHubAppEnvSetterFilter"/>
  </appender>
  <logger name="org.springframework">
    <!-- level info logs -->
    <level value="WARN" />
    <!--<appender-ref ref="Console" />-->
    <appender-ref ref="Chainsaw" />
  </logger>
  <logger name="org.apache.log4j.net">
    <!-- level info logs -->
    <level value="WARN" />
    <!--<appender-ref ref="Console" />-->
    <appender-ref ref="Chainsaw" />
  </logger>

  <root>
    <level value="debug" />
    <!--<appender-ref ref="Console" />-->
    <appender-ref ref="Chainsaw" />
   </root>
</log4j:configuration>
```

Thank You

mazcity Consulting Design Development
Let‘s do IT better!

- Comments, questions, suggestions?
  Contact me on: