SPRING CAMP 2017

비동기 어플리케이션과 모니터링으로 밀당하기

- 초초보 ver. -

비동기 어플리케이션, 어떻게 모니터링 할 것인가?

SCOUTER

Gunhee Lee

SCOUTER commiter

이건희 - 그분은 아닙니다.

그래서 구글서 검색 안됨

LINE

Line+ Programmer (& SNOW corp.)

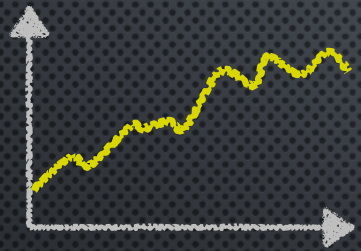
Scouter APM commiter

KOSSLab 3기 - 오픈소스프론티어

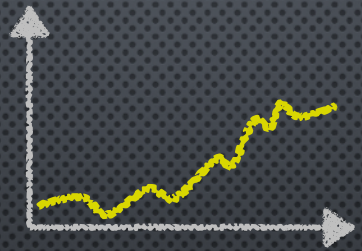
이야기 할 것

- 어플리케이션 모니터링과 APM
- APM의 구현 원리 (BCI)
- 모니터링을 위한 context의 전파 (Servlet)
- 비동기 어플리케이션 모니터링의 어려움
- 비동기 어플리케이션과 모니터링으로 밀당하기

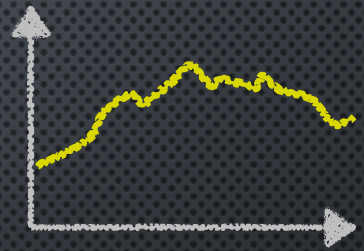
??



CPU



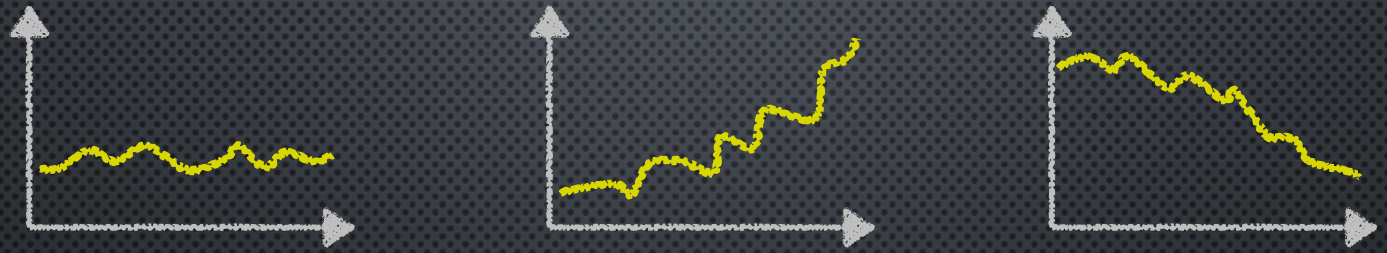
응답시간



처리량

이건 뭐...

??



이건 뭔가 좀 이상하다...

??

시간에 따른 숫자들 - 성능메트릭

-> visualization

??

이것들은 결과일 뿐...

??

원인을 모를때 우리는 이런 일들을 하지요...

??

RESTART !

??

그럼 왜 ???

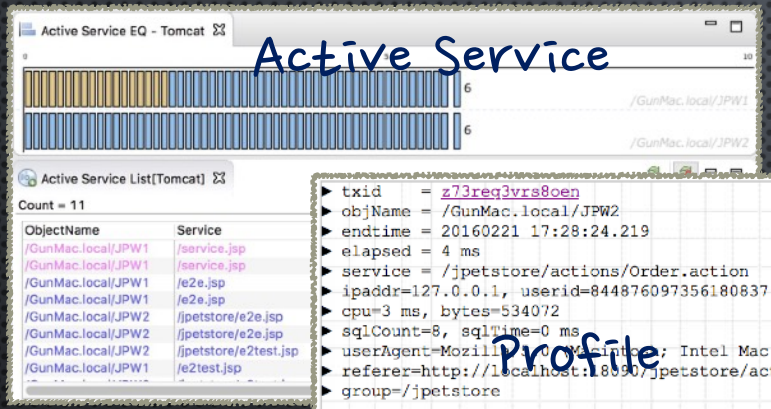
원인이 뭐냐!

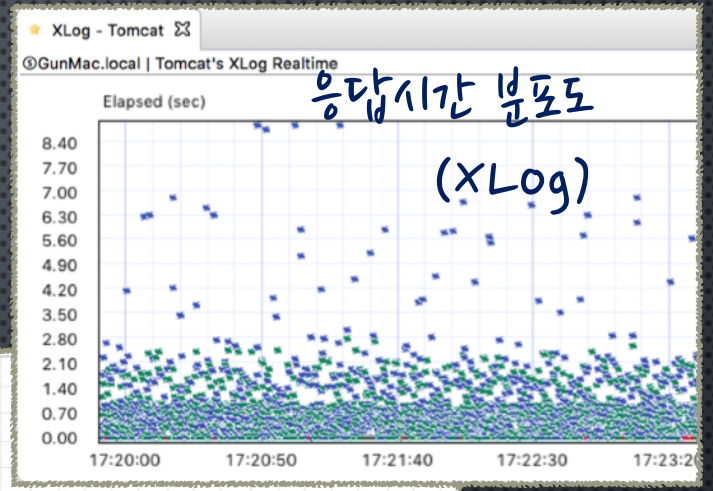
APM 이란?

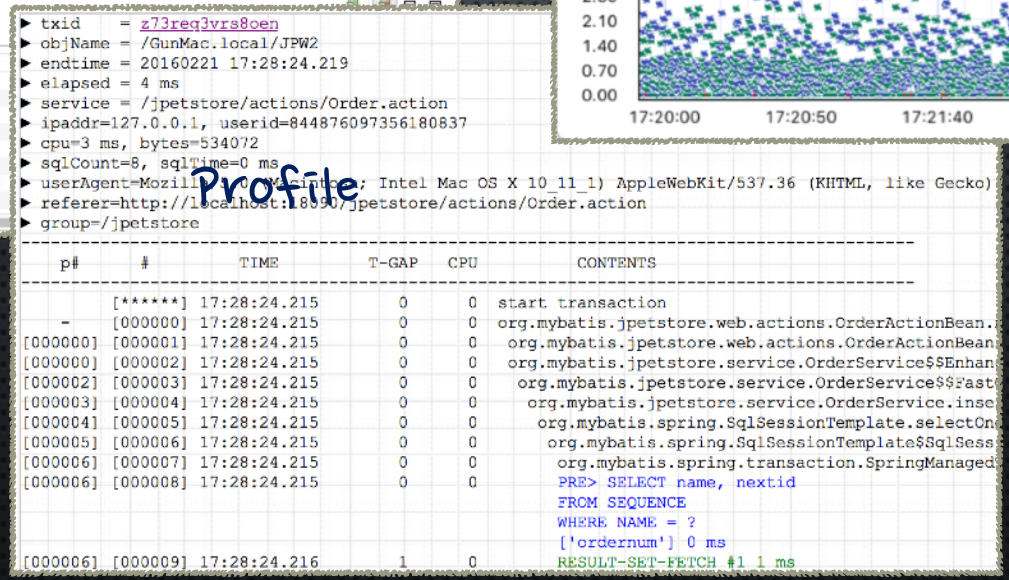
Application performance management

성능 메트릭 + 원인을 찾기위한 α

APM

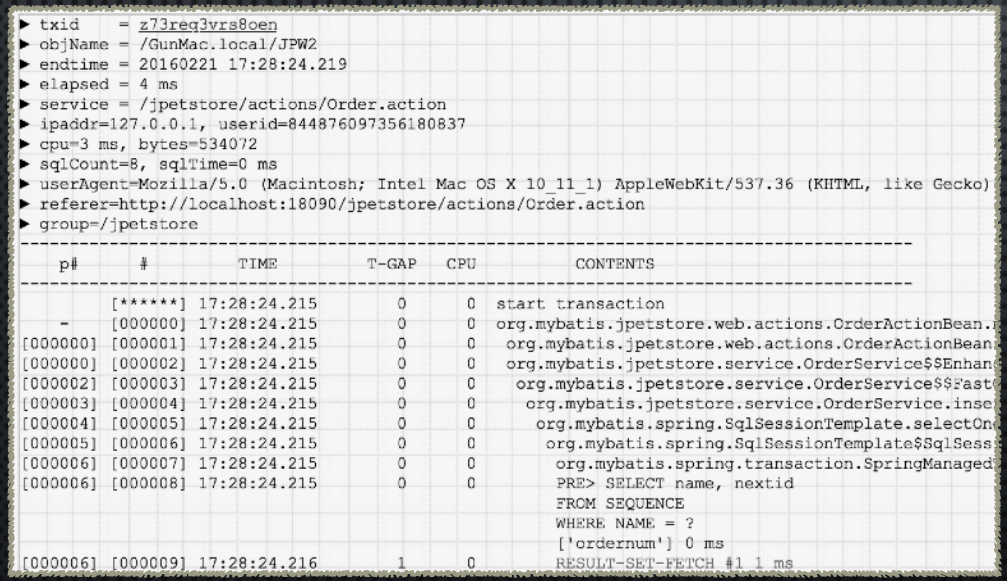

Active Service EQ - Tomcat
Active Service
Active Service List[Tomcat]
Count = 11
ObjectName
Service
/GunMac.local/JPW1 /service.jsp
/GunMac.local/JPW1 /service.jsp
/GunMac.local/JPW1 /e2e.jsp
/GunMac.local/JPW1 /e2e.jsp
/GunMac.local/JPW2 /jpetstore/e2e.jsp
/GunMac.local/JPW2 /jpetstore/e2e.jsp
/GunMac.local/JPW2 /jpetstore/e2test.jsp
/GunMac.local/JPW1 /e2test.jsp


XLog - Tomcat
GunMac.local | Tomcat's XLog Realtime
Elapsed (sec)
응답시간 분포도
(XLog)
8.40
7.70
7.00
6.30
5.60
4.90
4.20
3.50
2.80
2.10
1.40
0.70
0.00
17:20:00
17:20:50
17:21:40
17:22:30


Profile
▶ txid = z73req3vrs8oen
▶ objName = /GunMac.local/JPW2
▶ endtime = 20160221 17:28:24.219
▶ elapsed = 4 ms
▶ service = /jpetstore/actions/Order.action
▶ ipaddr=127.0.0.1, userid=844876097356180837
▶ cpu=3 ms, bytes=534072
▶ sqlCount=8, sqlTime=0 ms
▶ referer=http://localhost:8090/jpetstore/actions/Order.action
▶ group=/jpetstore
p# # TIME T-GAP CPU CONTENTS
[******] 17:28:24.215 0 0 start transaction
[000000] 17:28:24.215 0 0 org.mybatis.jpetstore.web.actions.OrderActionBean.
[000000] [000001] 17:28:24.215 0 0 org.mybatis.jpetstore.web.actions.OrderActionBean
[000000] [000002] 17:28:24.215 0 0 org.mybatis.jpetstore.service.OrderService$$Enhan
[000002] [000003] 17:28:24.215 0 0 org.mybatis.jpetstore.service.OrderService$$Fast
[000003] [000004] 17:28:24.215 0 0 org.mybatis.jpetstore.service.OrderService.inse
[000004] [000005] 17:28:24.215 0 0 org.mybatis.spring.SqlSessionTemplate.selectOn
[000005] [000006] 17:28:24.215 0 0 org.mybatis.spring.SqlSessionTemplate$SqlSess
[000006] [000007] 17:28:24.215 0 0 org.mybatis.spring.transaction.SpringManaged
[000006] [000008] 17:28:24.215 0 0 PRE> SELECT name, nextid
FROM SEQUENCE
WHERE NAME = ?
['ordernum'] 0 ms
[000006] [000009] 17:28:24.216 1 0 RESULT-SET-FETCH #1 1 ms

APM을 구현하는 기술

어떻게 성능 정보를 수집하는가?



```
▶ txid    = z73req3vrs8oen
▶ objName = /GunMac.local/JPW2
▶ endtime = 20160221 17:28:24.219
▶ elapsed = 4 ms
▶ service = /jpetstore/actions/Order.action
▶ ipaddr=127.0.0.1, userid=844876097356180837
▶ cpu=3 ms, bytes=534072
▶ sqlCount=8, sqlTime=0 ms
▶ userAgent=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko)
▶ referer=http://localhost:18090/jpetstore/actions/Order.action
▶ group=/jpetstore
-------------------------------------------------------------------------------
   p#      #          TIME          T-GAP   CPU          CONTENTS
-------------------------------------------------------------------------------
        [******] 17:28:24.215         0      0  start transaction
    -   [000000] 17:28:24.215         0      0  org.mybatis.jpetstore.web.actions.OrderActionBean.
[000000] [000001] 17:28:24.215         0      0   org.mybatis.jpetstore.web.actions.OrderActionBean
[000000] [000002] 17:28:24.215         0      0   org.mybatis.jpetstore.service.OrderService$$Enhan
[000002] [000003] 17:28:24.215         0      0    org.mybatis.jpetstore.service.OrderService$$Fast
[000003] [000004] 17:28:24.215         0      0     org.mybatis.jpetstore.service.OrderService.inse
[000004] [000005] 17:28:24.215         0      0      org.mybatis.spring.SqlSessionTemplate.selectOn
[000005] [000006] 17:28:24.215         0      0       org.mybatis.spring.SqlSessionTemplate$SqlSess
[000006] [000007] 17:28:24.215         0      0        org.mybatis.spring.transaction.SpringManaged
[000006] [000008] 17:28:24.215         0      0        PRE> SELECT name, nextid
                                                       FROM SEQUENCE
                                                       WHERE NAME = ?
                                                       ['ordernum'] 0 ms
[000006] [000009] 17:28:24.216         1      0        RESULT-SET-FETCH #1 1 ms
```

- cpu, memory
- method 정보
- sql, bind 변수
- 구간별 수행시간
- api 호출 url, 응답시간 ...

APM을 구현하는 기술

Super Power

BCI (Byte code instrumentation)

- bytecode에 직접 변경을 가해
  소스코드의 수정없이 원하는 기능을
  삽입할 수 있는 방법

BCI

어렵지 않다.

premain() 만 기억하자!

BCI

premain()

- premain()은 main() 전에 수행된다.
- 여기서 transformer를 통해 class 변경

- Libraries
  - ASM, Javassist, cglib, bytebuddy ···

BCI 예제

HttpServerlet.service()

Thread name enhancer

http-bio-7 :

뭔가 보자르다...

uri ??

Start time ?? 특정 parameter?

BCI 예제1

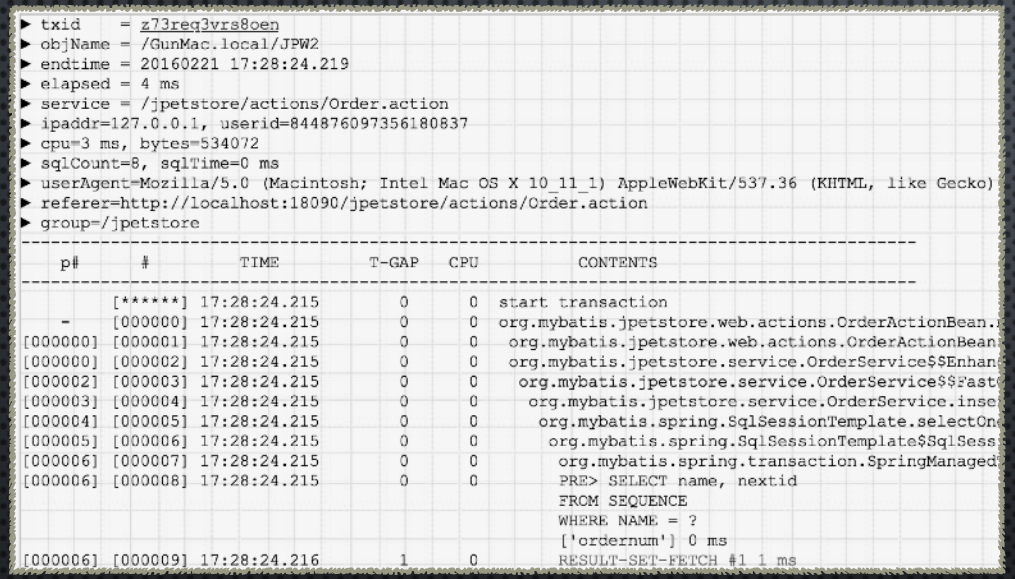
Thread name enhancer

https://github.com/gunlee01/thread-name-enhancer

모니터링을 위한

application Trace context

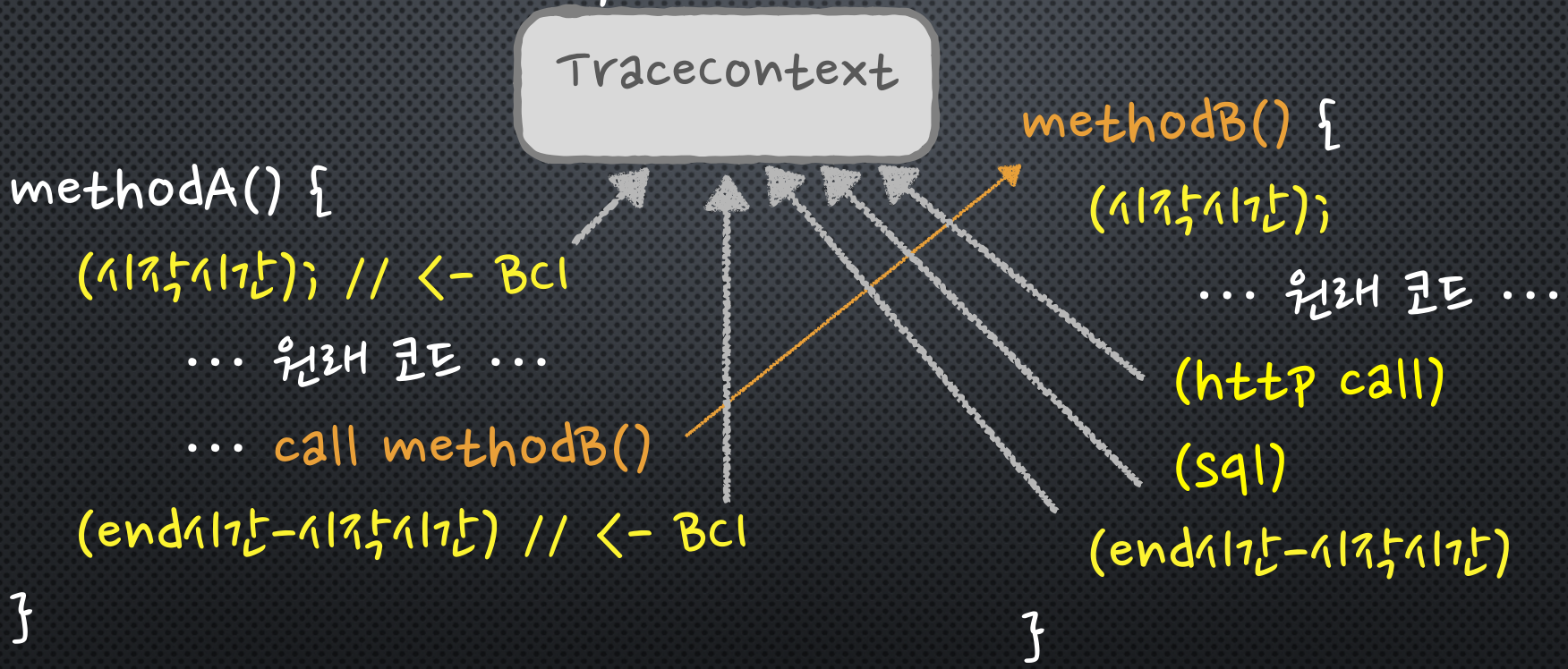
Trace context



```
▶ txid    = z73req3vrs8oen
▶ objName = /GunMac.local/JPW2
▶ endtime = 20160221 17:28:24.219
▶ elapsed = 4 ms
▶ service = /jpetstore/actions/Order.action
▶ ipaddr=127.0.0.1, userid=844876097356180837
▶ cpu=3 ms, bytes=534072
▶ sqlCount=8, sqlTime=0 ms
▶ userAgent=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko)
▶ referer=http://localhost:18090/jpetstore/actions/Order.action
▶ group=/jpetstore
-------------------------------------------------------------------------------------
   p#      #          TIME          T-GAP   CPU          CONTENTS
-------------------------------------------------------------------------------------
        [******] 17:28:24.215         0      0  start transaction
    -   [000000] 17:28:24.215         0      0  org.mybatis.jpetstore.web.actions.OrderActionBean.
[000000] [000001] 17:28:24.215         0      0   org.mybatis.jpetstore.web.actions.OrderActionBean
[000000] [000002] 17:28:24.215         0      0   org.mybatis.jpetstore.service.OrderService$$Enhan
[000002] [000003] 17:28:24.215         0      0    org.mybatis.jpetstore.service.OrderService$$Fast
[000003] [000004] 17:28:24.215         0      0     org.mybatis.jpetstore.service.OrderService.inse
[000004] [000005] 17:28:24.215         0      0      org.mybatis.spring.SqlSessionTemplate.selectOn
[000005] [000006] 17:28:24.215         0      0       org.mybatis.spring.SqlSessionTemplate$SqlSess
[000006] [000007] 17:28:24.215         0      0        org.mybatis.spring.transaction.SpringManaged
[000006] [000008] 17:28:24.215         0      0        PRE> SELECT name, nextid
                                                        FROM SEQUENCE
                                                        WHERE NAME = ?
                                                        ['ordernum'] 0 ms
[000006] [000009] 17:28:24.216         1      0        RESULT-SET-FETCH #1 1 ms
```

- CPU, memory
- method 정보
- SQL, bind 변수
- 구간별 수행시간
- API 호출 URL, 응답시간 ...

Method의 수행 시간 측정



Trace context를 어떻게 전달할 것인가?

```
class WelcomeController {
    void doStart(BizStuff stuff) {
        TraceContext traceContext = new TraceContext();
        traceContext.add(TraceInfo.SERVICE_START);
        welcomeBiz.doWelcomeBiz(stuff);
        springCampBiz.doSpringCampBiz(stuff);
        traceContext.add(TraceInfo.SERVICE_END);
    }
```

```
class WelcomeBiz {
    void doWelcomeBiz(BizStuff bizStuff) {
        traceContext.add(TraceInfo.METHOD_START);
        welcomeDao.doWelcomeDao(bizStuff);
        traceContext.add(TraceInfo.METHOD_END);
    }
```

Trace context를 어떻게 전달할 것인가?

static variable

```
class WelcomeController {
    public static TraceContext traceContext = new TraceContext();

    public void doStart(BizStuff stuff) {
        traceContext.add(new TraceInfo(TraceInfo.SERVICE_START));
        welcomeBiz.doWelcomeBiz(stuff, traceContext);
        springCampBiz.doSpringCampBiz(stuff, traceContext);
        traceContext.add(new TraceInfo(TraceInfo.SERVICE_END));
    }
}

class WelcomeBiz {
    public void doWelcomeBiz(BizStuff bizStuff) {
        traceContext.add(new TraceInfo(TraceInfo.METHOD_START));
        welcomeDao.doWelcomeDao(bizStuff);
        traceContext.add(new TraceInfo(TraceInfo.METHOD_END));
    }
```

Trace context를 어떻게 전달할 것인가?

static variable

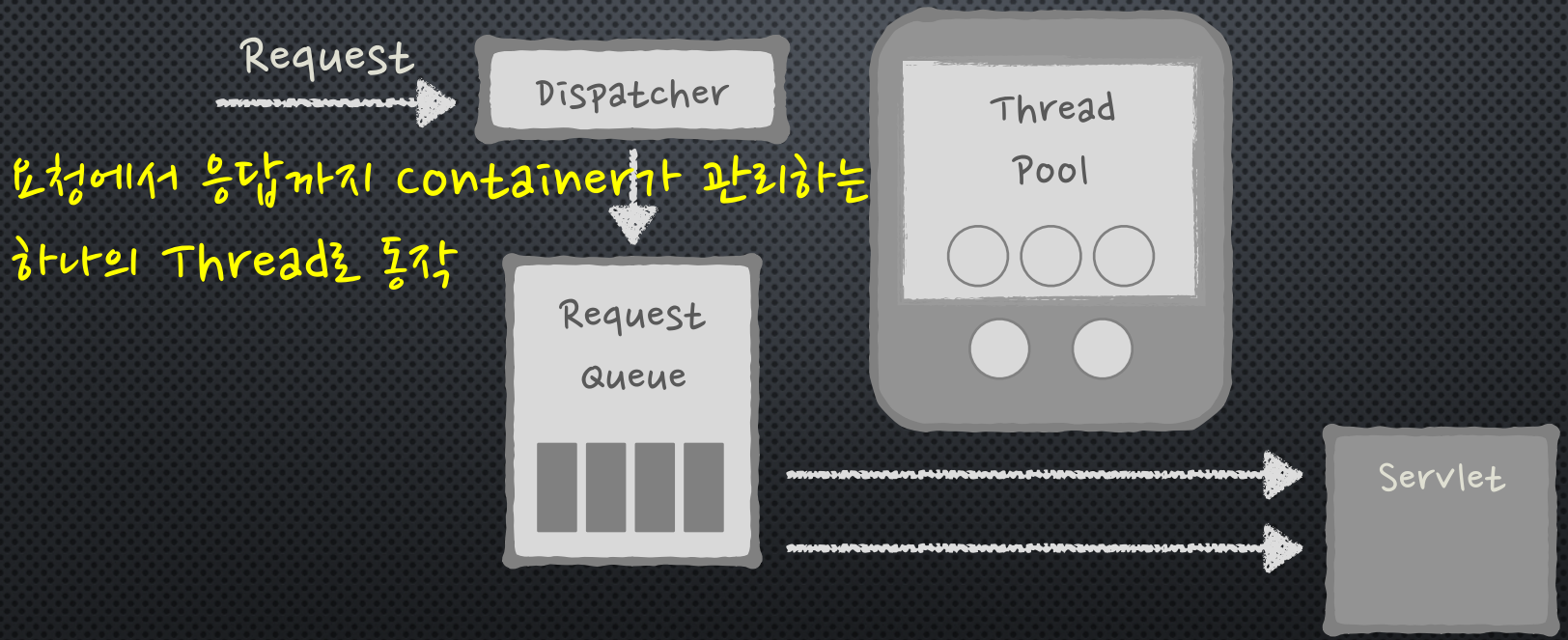
```
class WelcomeController {
    public static TraceContext traceContext = new TraceContext();

    public void doStart(BizStuff stuff) {
        traceContext.add(new TraceInfo(TraceInfo.SERVICE_START));
        welcomeBiz.doWelcomeBiz(stuff, traceContext);
        springCampBiz.doSpringCampBiz(stuff, traceContext);
        traceContext.add(new TraceInfo(TraceInfo.SERVICE_END));
    }
}

class WelcomeBiz {
    public void doWelcomeBiz(BizStuff bizStuff) {
        traceContext.add(new TraceInfo(TraceInfo.METHOD_START));
        welcomeDao.doWelcomeDao(bizStuff);
        traceContext.add(new TraceInfo(TraceInfo.METHOD_END));
    }
}
```

Trace context를 어떻게 전달할 것인가?

```
class WelcomeBiz {
    public void doWelcomeBiz(BizStuff bizStuff, TraceContext traceContext) {
        traceContext.add(new TraceInfo(TraceInfo.METHOD_START));
        welcomeDao.doWelcomeDao(bizStuff, traceContext);
        traceContext.add(new TraceInfo(TraceInfo.METHOD_END));
    }
```

- method의 파라미터로 전달 

Servlet container



ThreadLocal

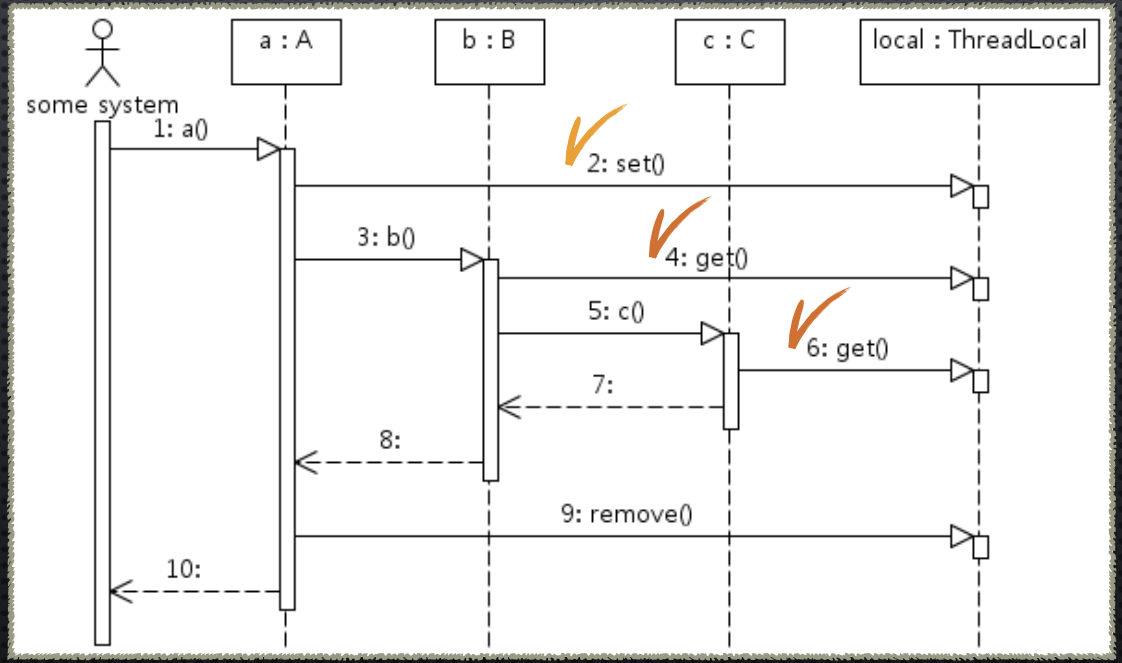
하나의 Thread에서 local 변수처럼
접근할 수 있는 무언가가 필요하다 !

This class provides thread-local variables.

Refs : https://docs.oracle.com/javase/7/docs/api/java/lang/ThreadLocal.html

ThreadLocal

This class provides thread-local variables.



ref: http://javacan.tistory.com/entry/ThreadLocalUsage

Simple ThreadLocal

```
public class SimpleThreadLocal {
    private static Map threadLocalMap = new ConcurrentHashMap();

    public void set(Object o) {
        threadLocalMap.put(Thread.currentThread(), o);
    }

    public Object get() {
        return threadLocalMap.get(Thread.currentThread());
    }
}
```

ThreadLocal 의 활용

- Security context
- Transaction context
- MDC (logging framework)

```
MDC.put("IP", "192.168.1.1");
```

```
<pattern>%d{HH:mm:ss.SSS} [%X{IP}] [%thread] %-5level
```

```
15:23:31.534 [192.168.1.1] [main] INFO
15:23:31.537 [192.168.1.1] [main] INFO
```

비동기 어플리케이션과 모니터링으로 밀당하기

Blocking Multi-thread

Servlet application의 문제점 ??

- Thread per request
  - : Blocking에 의한 Long running thread
  - : 과도한 Thread 사용 (context switching)
  - : thread pool 고갈로 인한 문제, cascading failure

Blocking Multi-thread

Servlet application의 문제점 ??

- Thread per request

그렇다면 장점은?

  : Blocking에 의한 Long running thread

  : 과도한 Thread 사용 (context switching)

  : thread pool 고갈로 인한 문제, cascading failure

Blocking Multi-thread

Servlet application의 장점

- 이해하고 디버깅 하기 쉽다.
  - Thread가 하나의 작업을 처리
  - Stack은 처리에 대한 정확한 Snapshot (threaddump)
  - XXXContext의 전달이 쉽다

blocking Servlet application의 장점

```
"http-nio-8080-exec-5" #34 daemon prio=5 os_prio=31 tid=0x00007fa14dc71800
   java.lang.Thread.State: RUNNABLE
        at java.net.SocketInputStream.socketRead0(Native Method)
        at java.net.SocketInputStream.read(SocketInputStream.java:150)
        at java.net.SocketInputStream.read(SocketInputStream.java:121)
        at sun.net.www.protocol.http.HttpURLConnection.getInputStream0(
        at org.springframework.web.client.RestTemplate.getForObject(RestTemplate.java
        at com.example.tovyreactive5.app1.MyApplication$MyController.callBlocking(MyAp
        at org.springframework.web.servlet.FrameworkServlet.processRequest(
        at org.springframework.web.servlet.FrameworkServlet.doGet(Framework
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:635)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(
        at org.apache.tomcat.util.threads.TaskThread$WrappingR
        at java.lang.Thread.run(Thread.java:745)
```

쉽고 단순하다.

blocking Servlet application의 장점

비동기 메소드 호출시

```
@GetMapping("/callAsync")
public ListenableFuture<String> callAsync() {
    //@Async annotated method
    return myService.asyncMethod();
```

```
"myThreadPool-1" #46 prio=5 os_prio=31 tid=0x00007fedf448d000 nid=0x5b1
   java.lang.Thread.State: RUNNABLE
        at java.net.SocketInputStream.socketRead0(Native Method)
        at java.net.SocketInputStream.read(SocketInputStream.java:150)
        at org.springframework.web.client.RestTemplate.getForObject(RestTemplate.java:309)
        at com.example.tovyreactive5.app1.MyApplication$MyController$MyService.asyncMethod(
        at org.springframework.aop.interceptor.AsyncExecutionInterceptor$1.call(AsyncExecut
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
        at java.lang.Thread.run(Thread.java:745)
```

blocking Servlet application의 장점

Non bloking IO 라면 흔적도 없다. ㅠㅠ

```
@GetMapping("/callNonBlocking")
public ListenableFuture<ResponseEntity<String>> callNonBlocking() {
    return asyncRestTemplate.getForEntity(url, String.class);
}
```

Netflix tech blog

비동기 시스템 장점이 대단하게 들리겠지만 위 혜택은 운영 비용이 발생합니다.
블록킹 시스템은 이해하고 디버깅하기 쉽고... 쓰레드의 스택은 요청 또는 생성된 작업의 처리 과정에 대한 정확한 스냇샵이 됩니다...

반대로 비동기는 ... 이벤트 반복의 스택 트레이스에서는 요청을 따라서 추적하는 것이 아무런 도움이 되지 않습니다. 이벤트와 콜백이 처리될 때 요청을 따라가기가 어렵고, 이러한 문제를 디버깅하도록 지원하는 도구는 매우 부족합니다.

Netflix tech blog

전반적으로 아키텍처 변경으로 얻을 수 있는 가치는 높았습니다. 커넥션 규모를 늘릴 수 있다 점이 중요한 혜택이었지만 비용이 많이 들었습니다. 우리는 디버깅하고, 개발하고, 테스트하기 훨씬 더 복잡한 시스템을 보유하고 있으며...

Netflix tech blog

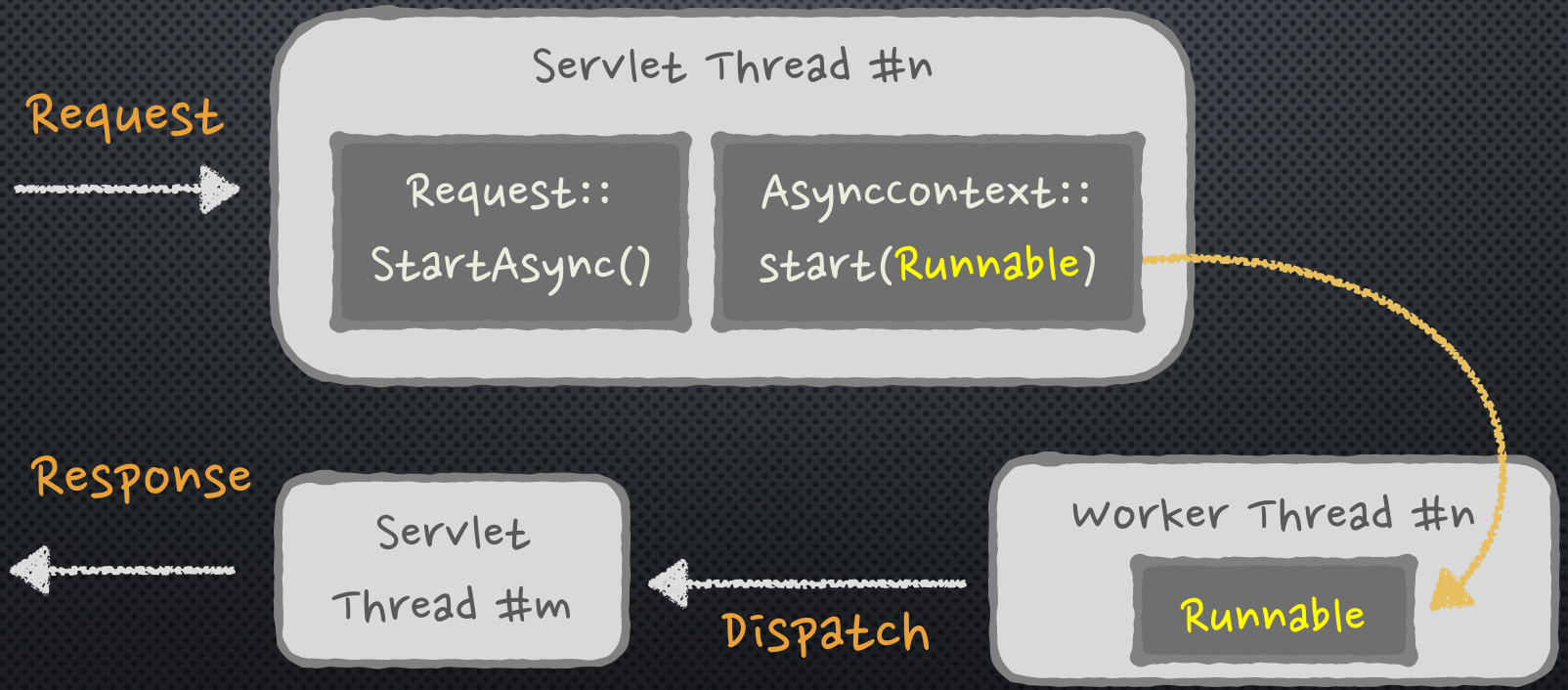
http://techblog.netflix.com/2016/09/zuul-2-netflix-journey-to-asynchronous.html

http://chanwookpark.github.io/reactive/netflix/zuul/async/non-blocking/2016/11/21/zuul2-netflixjourney-to-asynchronous-non-blocking-systems/

비동기 application 모니터링의 목표!

비동기이던 non-blockign이던
기존 어플리케이션 모니터링에서 제공하던
동일한(혹은 유사한) view를
볼수 있게 하는 것.

비동기 Servlet (Servlet 3.0+)



비동기 처리

```
new Thread(new Runnable() {
    public void run() { doSomething();}
}).start();
```

another thread #1

```
executorService.execute(new Runnable() {
    public void run() { doSomething(); }
});
```

another thread #2

```
Future future = executorService.submit(new Callable() {
    public Object call() throws Exception {
        return doSomething();
    }
});
```

another thread #3

이제는 ThreadLocal을 통해
context를 전파할 수 없다 !!

InheritableThreadLocal

Good !

This class extends ThreadLocal
to provide inheritance of values
from parent thread to child thread.
When a child thread is created,
the child receives initial values ...

Refs : https://docs.oracle.com/javase/7/docs/api/java/lang/InheritableThreadLocal.html

InheritableThreadLocal

```
public Thread(Runnable target) {
    init( g: null, target,  name: "Thread-" + nextThreadNum(),  stackSize: 0);
}
```

```
private void init(ThreadGroup g, Runnable target, String name,
                  long stackSize, AccessControlContext acc) {
    if (name == null) {
        throw new NullPointerException("name cannot be null");
    }

    if (parent.inheritableThreadLocals != null)
        this.inheritableThreadLocals =
            ThreadLocal.createInheritedMap(parent.inheritableThreadLocals);
```

InheritableThreadLocal

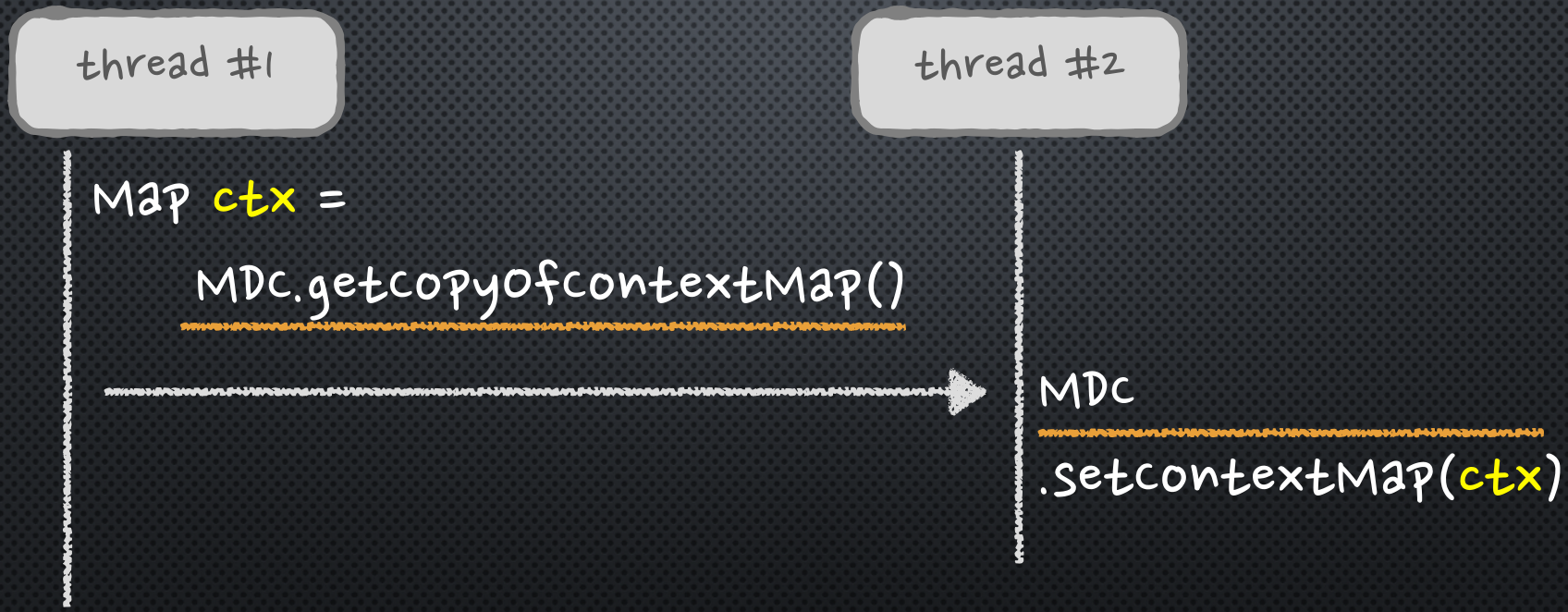
정말 이것으로 가능한가?

Logback manual # MDC

A copy of the mapped diagnostic context
can not always be inherited by worker threads from the initiating thread.
This is the case when java.util.concurrent.Executors is used for thread management. ... creates a ThreadPoolExecutor and like other thread pooling code, it has intricate thread creation logic.
In such cases,
... MDC.getCopyOfContextMap() is invoked on the original (master) thread before submitting a task to the executor.
... When the task runs, as its first action,
it should invoke MDC.setContextMapValues() to associate ...

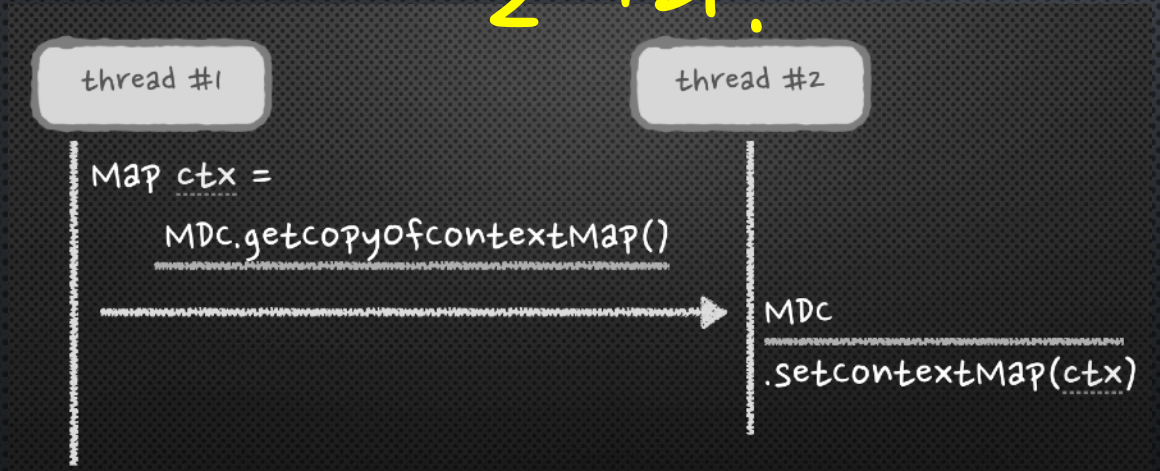
Refs : https://logback.qos.ch/manual/mdc.html#managedThreads

Logback manual# MDC



Logback manual# MDC

결국. 내 알아서 전달해라.



밀당의 시작

#1

알아서 전달하기

Parameter를 활용해 context 전달하기

Scouter APM 에 가장 먼저 제공된 기능

Parameter를 활용해 context 전달하기

```
void doSpringCamp(final BizStuff stuff) {
    executorService.execute(new Runnable() {
        public void run() { springCampBiz.doSpringCampBiz(stuff); }
    });
}
```

Parameter를 활용해 context 전달하기

```
void doSpringCamp(final BizStuff stuff) {

    TraceContextTransfer.set(stuff, TraceContextHoler.get());

    executorService.execute(new Runnable() {
        public void run() {
            TraceContextHoler.set(TraceContextTransfer.get(stuff));
            springCampBiz.doSpringCampBiz(stuff);
        }
    });
}
```

Parameter를 활용해 context 전달하기

전달할 조건을 직접 설정

```
//@forward(class="Test1", name="doSpringCamp", param=0)
//@receive(class="Test1$1", name="run", param=0)
```

```
TraceContextTransfer.set(stuff, TraceContextHoler.get());
```

```
TraceContextHoler.set(TraceContextTransfer.get(stuff));
```

밑당하기

#2

Runnable & callable

Runnable & callable

결국 Runnable 또는 callable !!!

```
new Thread(new Runnable() {
    public void run() { doSomething();}
}).start();
```

```
executorService.execute(new Runnable() {
    public void run() { doSomething(); }
});
```

```
Future future = executorService.submit(new Callable() {
    public Object call() throws Exception {
        return doSomething();
    }
});
```

Runnable & callable

```
class SCamp1 extends AbstractPlugin {
    void doSpringCamp(final BizStuff stuff) {
        executorService.execute(new Runnable() {
            public void run() {
                springCampBiz.doSpringCampBiz(stuff);
            }
        });
    }
```

Runnable & callable

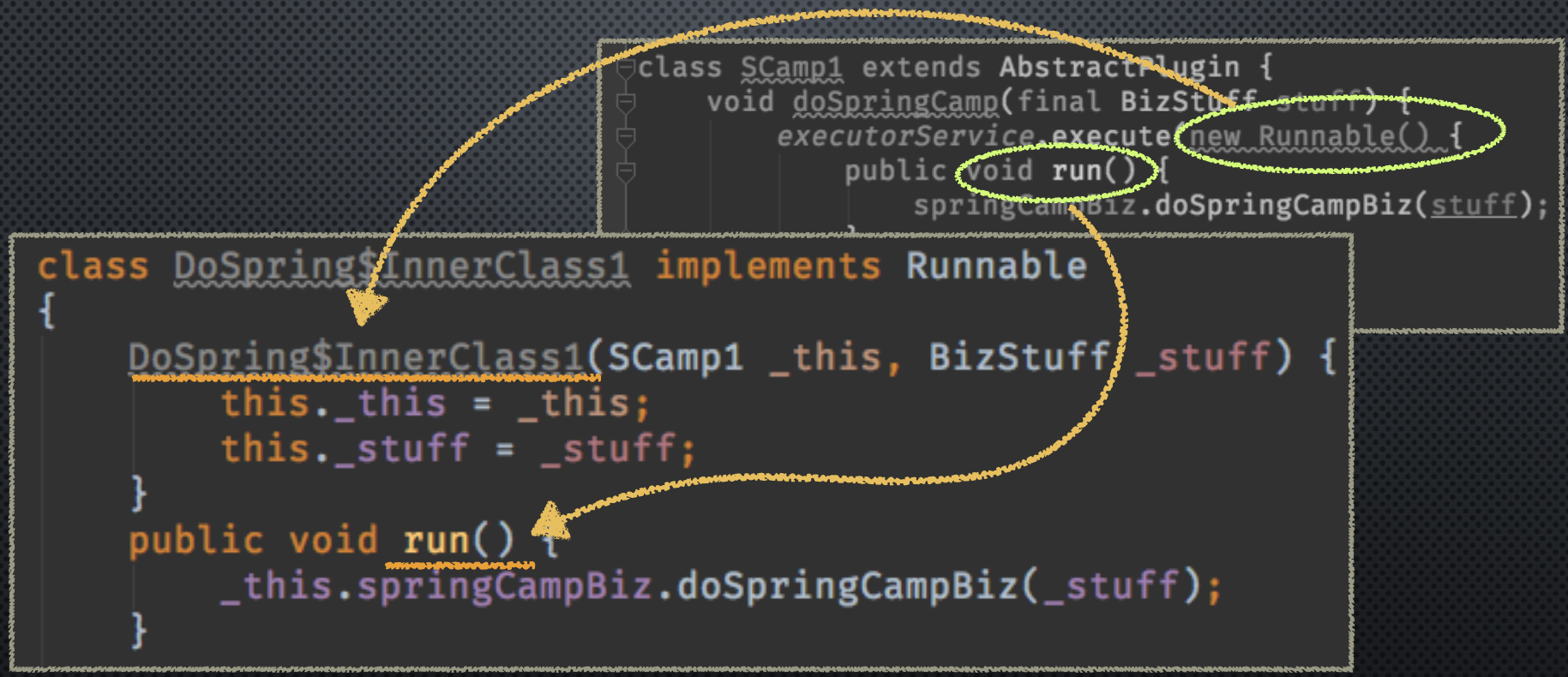
실제로는

익명 Inner class가 생긴다.

```
class SCamp1 extends AbstractPlugin {
    void doSpringCamp(final BizStuff stuff) {
        executorService.execute(new Runnable() {
            public void run() {
                springCampBiz.doSpringCampBiz(stuff);
```

```
class DoSpring$InnerClass1 implements Runnable
{
    DoSpring$InnerClass1(SCamp1 _this, BizStuff _stuff) {
        this._this = _this;
        this._stuff = _stuff;
    }
    public void run() {
        _this.springCampBiz.doSpringCampBiz(_stuff);
    }
```

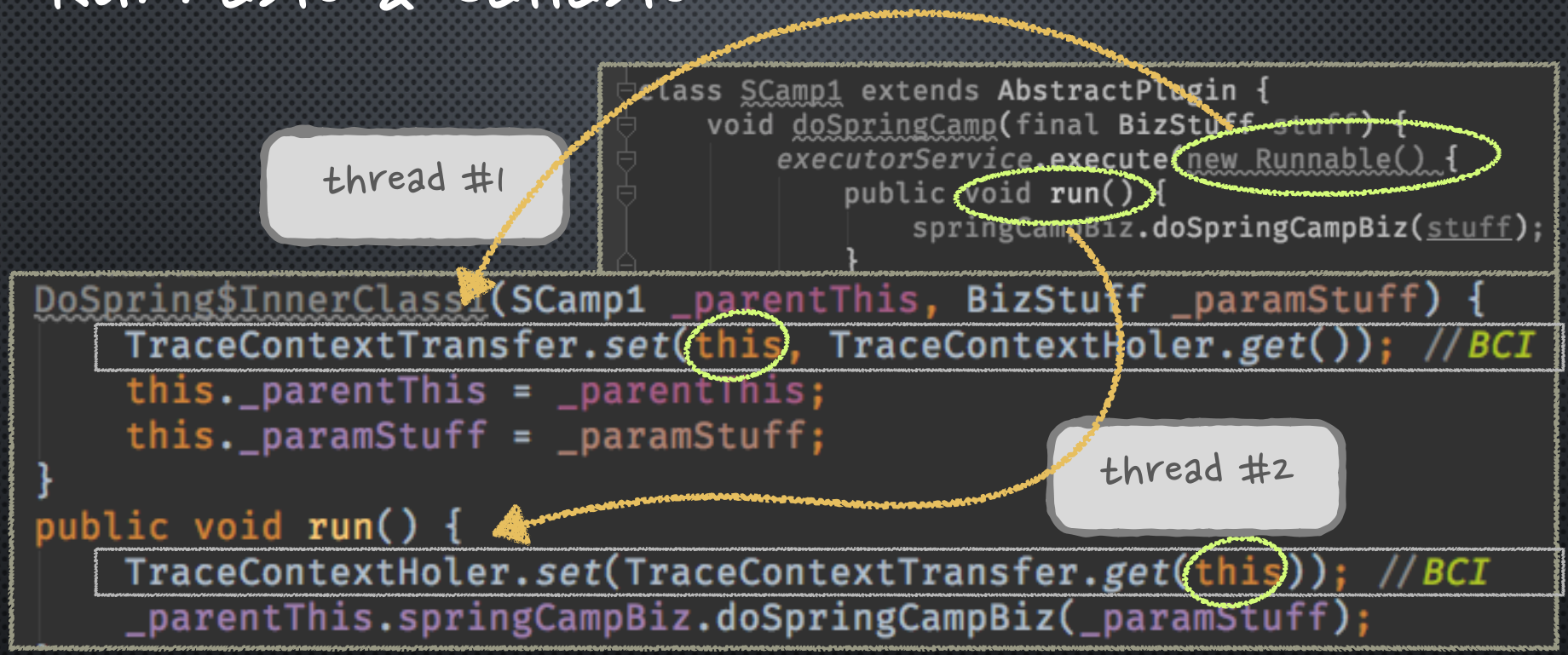
Runnable & callable



Runnable & callable

```
DoSpring$InnerClass1(SCamp1 _parentThis, BizStuff _paramStuff) {
    TraceContextTransfer.set(this, TraceContextHoler.get()); //BCI
    this._parentThis = _parentThis;
    this._paramStuff = _paramStuff;
}
public void run() {
    TraceContextHoler.set(TraceContextTransfer.get(this)); //BCI
    _parentThis.springCampBiz.doSpringCampBiz(_paramStuff);
```

Runnable & callable



Good ~~ !

한방에 해결

(아직도 진행 중인)
끝나지 않은 밀당
#3
Lambda expression

Lambda expression

여는 또 어쩔...

```
public void lambdaEx1(int idx) {
    es.execute(() → {
        System.out.println("inside lambda : " + idx);
    });
}
```

Lambda expression

여는 또 어쩔...

```
public void lambdaEx2(int idx) {
    ListenableFuture f1 = asyncRestTemplate.getForEntity(url, String.class);
    f1.addCallback(
        s → System.out.println("inside callback : " + s),
        e → System.out.println("inside callback error : " + e)
    );
}
```

Lambda expression

앞에서

Runnable을 통해 전달할때와 똑같이 해보자.

(1) 생성자에서 this를 key로 ctx 저장

(2) run() 에서 this로 ctx를 겟!

Lambda expression

이 두가지는 같은 것인가?

```
public class LambdaEx1 {
    void doIt(int idx) {
        Runnable r = () → System.out.println(idx);
    }
}
```

```
void doIt(int idx) {
    Runnable r = new Runnable() {
        public void run() { System.out.println(idx);}
    };
}
```

Lambda expression

anonymous inner class != lambda expression

Lambda expression

그렇다면 Lambda 식은 inner class 가 아닌가?

yes 인데 그냥 yes는 아니고…

Lambda expression

그렇다면 Lambda 식은 inner class 가 아닌가?

Lambda body는 inner class가 아니다.

Lambda body는 내부적으로 생성되는 inner class를 통해 실행된다. (Lambda proxy)

하지만 일반적인 inner class 와는 생성과정이 다르다.

Lambda expression desugar

Lambda의 body는

private (static) method가 된다.

```
//private static synthetic lambda$doIt$0(I)V
private static void lambda$doIt$0(int paramIdx) {
    System.out.println(paramIdx);
}
```

Lambda expression

따라서 Lambda body에서의 this 는 lambda식이 있는 class의 인스턴스이다.

```
public class LambdaEx1 {
    void doIt(int idx) {
        Runnable lambdaRunnable = () -> {
            System.out.println(idx + ":lambda-this:" + this);
        };

        Runnable innerClassRunnable = new Runnable() {
            public void run() {
                System.out.println(idx + ":innerclass-this:" + this);
            }
        };
```

Lambda expression

SPRING CAMP 2017

Lambda expression

앞에서

Runnable을 통해 전달할때와 똑같이 해보자.

(1) 생성자에서 this를 key로 ctx 저장 ??

(2) run() 에서 this로 ctx를 겟! ??

Lambda expression

```
Runnable r = () → System.out.println(idx);
```

```
INVOKEDYNAMIC run(I)Ljava/lang/Runnable; [
  // handle kind 0×6 : INVOKESTATIC
  java/lang/invoke/LambdaMetafactory.metafactory(Ljava/lang/invoke/MethodHandles$Lookup;
  // arguments:
  ()V,
  // handle kind 0×6 : INVOKESTATIC
  springcamp2017/session/asyncmonitoring/gunlee/LambdaEx1Ex2.lambda$doIt$0(I)V,
  ()V
]
```

Lambda expression

```
CallSite metafactory(MethodHandles.Lookup caller,  caller: "springcamp2017.session.asyncm
                     String invokedName,  invokedName: "run"
                     MethodType invokedType,  invokedType: "(int)Runnable"
                     MethodType samMethodType,  samMethodType: "()void"
                     MethodHandle implMethod,  implMethod: "MethodHandle(int)void"
                     MethodType instantiatedMethodType)  instantiatedMethodType: "()void"
```

Lambda expression

* 최초 실행시 bootstraping

—> LambdaMetafacoty::metafactory()

(1) Functional I/F를 구현하는 Innerclass 동적 생성 (ASM으로)

- Lambda Proxy

(2) Innerclass의 instance를 반환하는 factory Method

에 대한 methodHandle(callSite) 반환

Lambda expression

```
public class LambdaEx1 {
    void doIt(int idx) { Runnable r = () → System.out.println(idx);}

    //private static synthetic lambda$doIt$0(I)V
    private static void lambda$doIt$0(int paramIdx) { System.out.println(paramIdx); }

    //On bootstrapping (LambdaMetafactory)
    //Generated by ASM library and vm innerclass loaded by anonymous classloader
    static class LambdaEx1$$Lambda$1 implements Runnable {
        private final int arg$1;
        private LambdaEx1$$Lambda$1(int arg$1) { this.arg$1 = arg$1; }

        private static Runnable get$Lambda(int arg$1) { return new LambdaEx1$$Lambda$1(arg$1); }

        @Override public void run() { lambda$doIt$0(arg$1); }
    }
```

Lambda expression

```
public class LambdaEx1 {
    void doIt(int idx) { Runnable r = () → System.out.println(idx);}

    //private static synthetic lambda$doIt$0(I)V
    private static void lambda$doIt$0(int paramIdx) { System.out.println(paramIdx); }

    //On bootstrapping (LambdaMetafactory)
    //Generated by ASM library and vm innerclass loaded by anonymous classloader
    static class LambdaEx1$$Lambda$1 implements Runnable {
        private final int arg$1;
        private LambdaEx1$$Lambda$1(int arg$1) { this.arg$1 = arg$1; }

        private static Runnable get$Lambda(int arg$1) { return new LambdaEx1$$Lambda$1(arg$1); }

        @Override public void run() { lambda$doIt$0(arg$1); }
    }
```

생성할때 this를 ctx 전파

Run에서 this를 ctx 겟!

Lambda 정복

LambdaMetafacoty에서
Innerclass 생성시에
BCI로 적용하면 모든것이
한방에 끝난다. !!

Lambda ??

그런데 ???

classfile Transformer를 걸어도

transform()에 전달되지가 않는다...

Lambda ??

고난의 시작 ...

구글신께 물어봐도 별다른 답이 없고...

Lambda ??

원인은 바로 이것

```
return UNSAFE.defineAnonymousClass(targetClass, classBytes, objects: null);
```

Lambda ??

원인은 바로 이것

```
return UNSAFE.defineAnonymousClass(targetClass, classBytes, objects: null);
```

anonymous class loader에 의해 class loading

-> anonymous-in-VM-sence class

-> target class의 보안 수준 계승 (private 접근 가능)

Lambda ??

Anonymous class loader에 의해 로드된 클래스는
classfiletransformer를 경유하지 않으므로
transform 할수 없다 !!!

Lambda ??

용자의 등장 !

Lambda ??

Rafael Winterhalter

http://rafael.codes/

Bytebuddy

Rafael Winterhalter

- Bytebuddy#1 author
  http://bytebuddy.net
- code generation and manipulation library
  (Mockito, Hibernate ...)

Bytebuddy

Rafael Winterhalter

- Lambdametafacoty 에서 생성하는 Innerclass 를 일반 class 처럼 transform() 할수 있게 해줘!
- Open JDK groups 를 통해 의견 교환

http://mail.openjdk.java.net/pipermail/core-libs-dev/2016-January/038199.html

Bytebuddy

- 해줘 !

- 왜 안돼?

- using a vm anonymous class is a workaround of the fact that there is currently no way to load a function/method without a class.

- 안돼 !

- 좀다는 좀다야. 이 구현은 바뀔거야. 지금은 방법이 없어서 걍 일케 한거야.
- 좀 더 멋지게 바뀔거라구.
(언젠가는 ...)

Bytebuddy

- 넌 현실을 참 모르는구나!
  Real world에선 필요하다구.

- 자꾸 그럼 내가 만든다!

- 안돼! 이걸로 뭘 하려고 하지마.
  (왜 필요한지도 잘 모르겠구...)

- 증말 하지마.
  JVM의 구현이 바뀌면
  다 쓸모 없어 진다구 ...

Bytebuddy

```
new AgentBuilder.Default()
        .with(AgentBuilder.LambdaInstrumentationStrategy.ENABLED)
        .installOn(instrum);
```

LambdaMetafacoty 자체를 transform()

Lambda class transform

해보니 잘된다 !

Lambda class transform

Scouter v0.5.0에 적용

릴리즈

Lambda class transform

갑자기 밀려드는 버그 리포팅

Lambda class transform

Functional I/F가 Supplier인 경우

```
java.lang.IllegalStateException: size = 0
        at net.bytebuddy.matcher.FilterableList$AbstractBase.getOnly(FilterableList.java:10
        at net.bytebuddy.agent.builder.AgentBuilder$LambdaInstrumentationStrategy$LambdaIns
        at net.bytebuddy.dynamic.scaffold.MethodRegistry$Handler$ForImplementation.compile(
```

결국

No one-size-fits-all strategy

비동기 어플리케이션 모니터링

아직 할 수 있는 방법은 많다.

할 수 있는 것

잘 알려진 F/W, Library의 메커니즘들에 적용

Executor의 execute(), submit()

AsyncExecutionAspectSupport.doSubmit()

...

할 수 있는 것

믿고 쓰는 Spring

- Spring-web-reactive 에서도
  Securitycontext 등을
  F/W 수준에서 전파하는 방법을 제공할 것
- 그 방식을 그대로 사용

밀당은 계속된다 !

하고 싶은 것

비동기 어플리케이션을 (& Non blocking)
쉽게 추적하고 모니터링 할 수 있게 만들기

비동기 application 모니터링의 목표 !

비동기이던 non-blockign이던
기존 어플리케이션 모니터링에서 제공하던
동일한(혹은 유사한) view를
볼수 있게 하는 것.

SCOUTER에 참여하기

Github

https://github.com/scouter-project/scouter

Facebook 사용자 모임

https://www.facebook.com/groups/scouterapm

감사합니다

gunlee01@gmail.com

SCOUTER