

2時間集中 Java™トラブル解決ドリル

岡崎 隆之

ソフトウェア・ビジネス統括本部

Java エバンジェリスト

<http://blogs.sun.com/okazaki>



セッション概要

なぜかトラブルにまつわる逸話は身の回りにはたくさん……。

本日はトラブル解決のための技だけではなく、考え方もご紹介します

自己紹介

- Java エバンジェリスト (副業)
 - > ブログを書いたりセミナーに行ったりする
- アイデンティティ管理 (本業)
 - > 製品の紹介や、提案活動など
- そのまえ
 - > Java コンサルティングなど
- その他
 - > Sun 入社もうすぐ丸 6 年目
 - > プログラミング歴 15 ~ 18 年



自己紹介（続き）

- 過去6年間の仕事の傾向
 - > 一つのプロジェクトに長期間参加した事はあまりない
 - > 突然話が舞い込んできて、よくわからないまま参加する
 - > 使用されているテクノロジーが知っているものだけとは限らない
 - > 作業の優先度はあれもこれも「最優先」である

本日の内容に含まれないところ

- 著作権や特許のトラブルで困っている
- がんばっているのに、給料をあげてもらえない
- 対人関係のトラブルで困っている
- プロジェクトが慢性的にデスマーチで、抜け出したい



ドリル 1:

トラブルの時って
こんな感じですよ



思い出してみてください

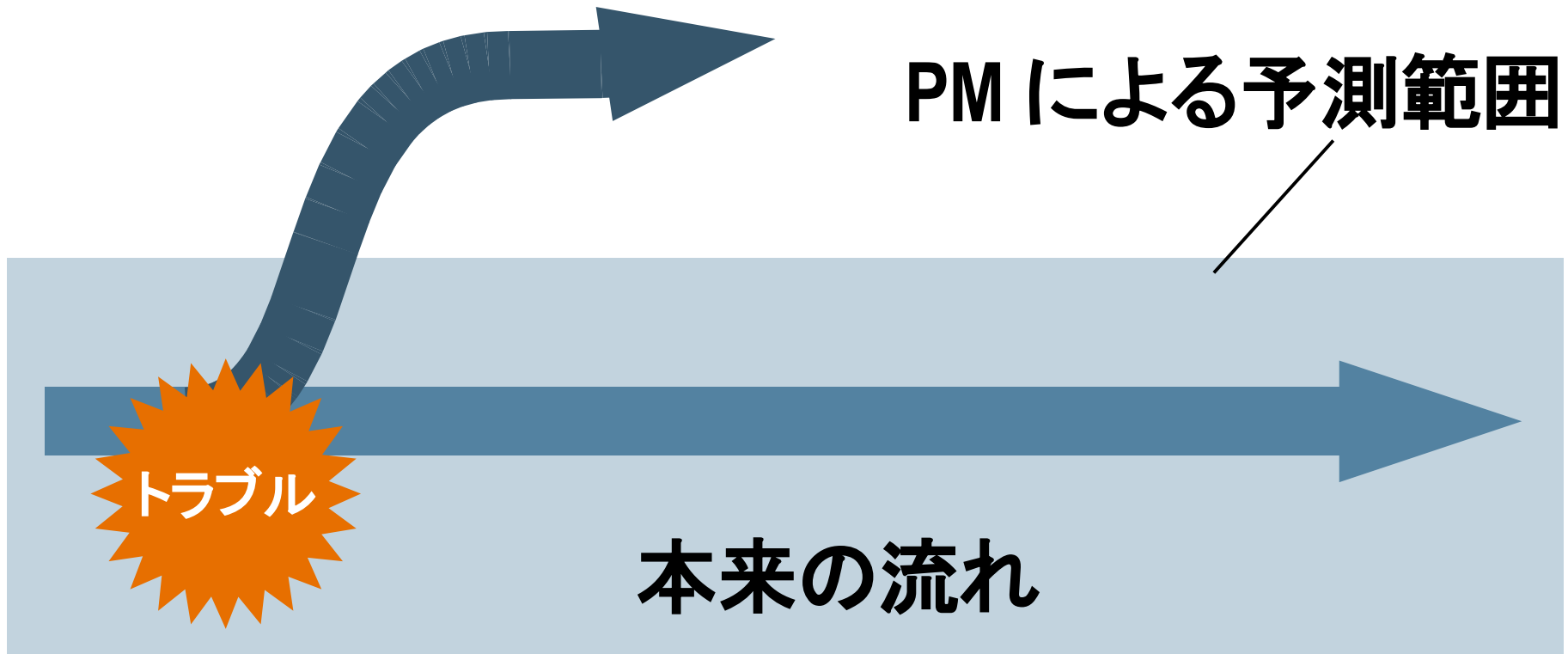
- トラブルで徹夜をしたことがある
- 帰宅しようとしたら電話で呼び戻された
- 朝起きたら携帯に着信履歴がたくさん
- その他

思い出してみてください

- トラブルで徹夜をしたことがある
 - 帰宅しようとしたら電話で呼び戻された
 - 朝
 - そ
- 通常のルーチンは機能していない
 - 情報はうまく整理されていない
 - 現場はコントロールを失っている

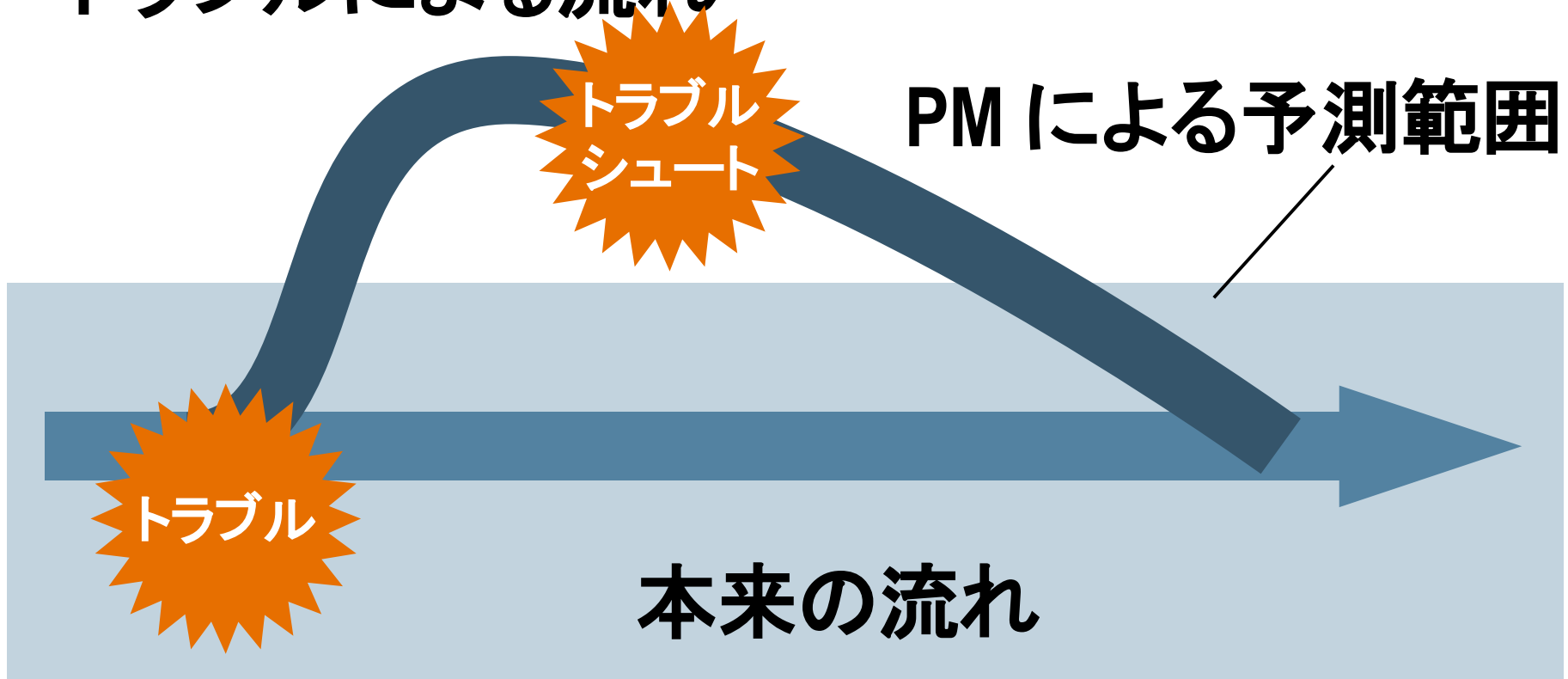
プロジェクトの流れ

トラブルによる流れ



プロジェクトの流れ

トラブルによる流れ



トラブル解決のみちすじ

情報を収集／整理する



コントロールを取り戻す



通常のルーチンに戻す

コントロールを取り戻そう！

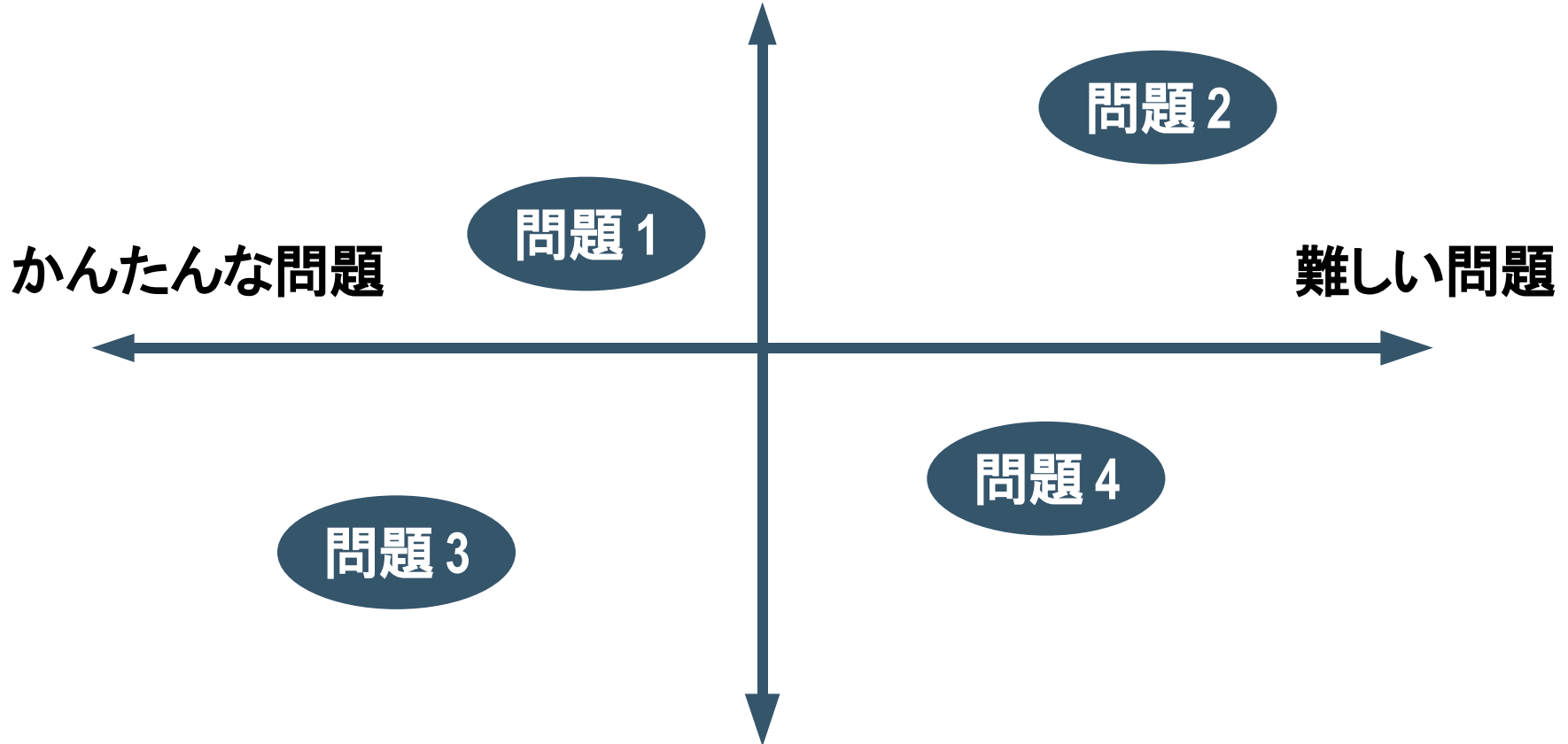
- がんばった人を非難してはいけない
 - > がんばった人は状況をよく知っている
- なるべく客観的な情報を収集する
 - > 混乱している現場ではヒアリング結果にはゆらぎがある
 - > 思い (guess) と事実 (fact) を区別する
 - > 検索可能な場所に置いておく (メールボックス、Wiki …)
- 下手な鉄砲数打ちゃあたる・・・？
 - > あたる見込みが低いので、コスト高
 - > それなら、そもそも単体テストで発見されていたのでは？

コントロールを取り戻そう！(続き)

- 何が最も悪影響を及ぼしているのかを絞り込む
 - > 接続できない？
 - > パフォーマンス？
 - > システムクラッシュ？
 - > バグ？ どんなバグ？
- 何を解決すれば最も顧客満足度が高まるのか？
 - > どういう作業をすれば最も喜ばれるのか？
 - > 説得材料を集める

どこから手を付けるかを定める

解決により顧客満足度が高まる



解決してもあまり満足度はかわらない

どこから手を付けるかを決める（続き）

解決により顧客満足度が高まる

問題2

かんたんな問題

難しい問題

言っている人によって変わりやすい
伝言ゲームによって変動しやすい

解決してもあまり満足度はかわらない

どこから手を付けるかを決める（続き）

- なるべく客観的な表現に置き換えて優先順位付け
 - > ビジネス・インパクト（金額・影響ユーザ数）
 - > なぜ、XX 日までに解決が必要なのか
- 過去のトラブルにまつわるやり取りの傾向
- 最もお客様とパイプを持っている人の直感を信じる
- トラブル解決に加わる事のできるメンバーの力量

ドリル1:まとめ

- なるべく客観的な情報を集める
- どこから手を付けるかを決める
- コントロールを取り戻す

ドリル 2:

トラブルって言うと
こんなやつですね



難しいトラブルといえはば・・・

- 答えが違う
- システムクラッシュ
- 連携先に接続できない
- メモリ不足
- パフォーマンスがでない
- セキュリティの問題
- その他

なぜ難しいのか

- 手がかりが少ない
 - > ログに出力される情報が少ない等
 - 手元の環境では再現しない
 - > 環境やデータに依存？
 - 扱う問題の原因が広範囲
 - > 原因となる部分が至る所に拡散している
 - 根本的な部分に原因がある
 - > システムアーキテクチャ
- 原因不明
- 原因は特定済み

手がかりが少ないトラブル解決

- ログレベルをより詳細にする
 - > デバッグログの有効化
 - > プログラムにログ出力を組み込む
 - > コアダンプを取得できるように設定する
- デバッガやツールを使う
 - > 可能な限り有効利用する
 - > 動的な情報の取得や、動作の把握

手元の環境では再現しない

- 長期戦になる可能性が高いため、交渉の材料を集める
 - > 本番環境でしか発生しない問題の解決は困難
 - > デバッガやツールを容易に導入する事ができない
 - > SOX 法により、本番環境／データをつかった再現テストはより困難に
 - > 過去の類似事例等の情報を集める
- 運用に影響を与えにくいツールの導入を交渉
 - > Dtrace

扱う問題の原因が広範囲

- なるべく自動化して解決できる方法を考える
 - > ツールの導入あるいはスクリプトを作成
- 網羅的に修正／対策ができたかどうかを調べる方法を準備する
 - > ツールやスクリプトによるレポートの作成

根本的な部分に原因がある

- 長期化する可能性が高いので、一時的な回避策を作成
 - > 交渉の材料を集める
 - > 一時的な回避策の作成も検討する
 - > 十分交渉の材料を集めないと、一時的な回避策がそのまま採用されて継続的に採用される事に・・・
- アーキテクチャの設計では” Simple is Best” だが、
“Simple is Hard” であることもよく考慮しておく

ドリル2:まとめ

- ツールやスクリプトを使ってなるべく手間を省く
- プログラムの修正だけでなく、作業がやりやすいように交渉の準備

ドリル 3:

使っていないつもり。
でも、使い過ぎ



Q. OutOfMemoryError の原因は？

1. OS のメモリが足りなくなった
2. Full GC が実行されたため
3. ヒープ領域がこれ以上拡張できない
4. インスタンス数が多すぎる

Java VM におけるメモリ領域

1.4.2

5.0

6

C/ネイティブ
ヒープ領域

Java ヒープ領域

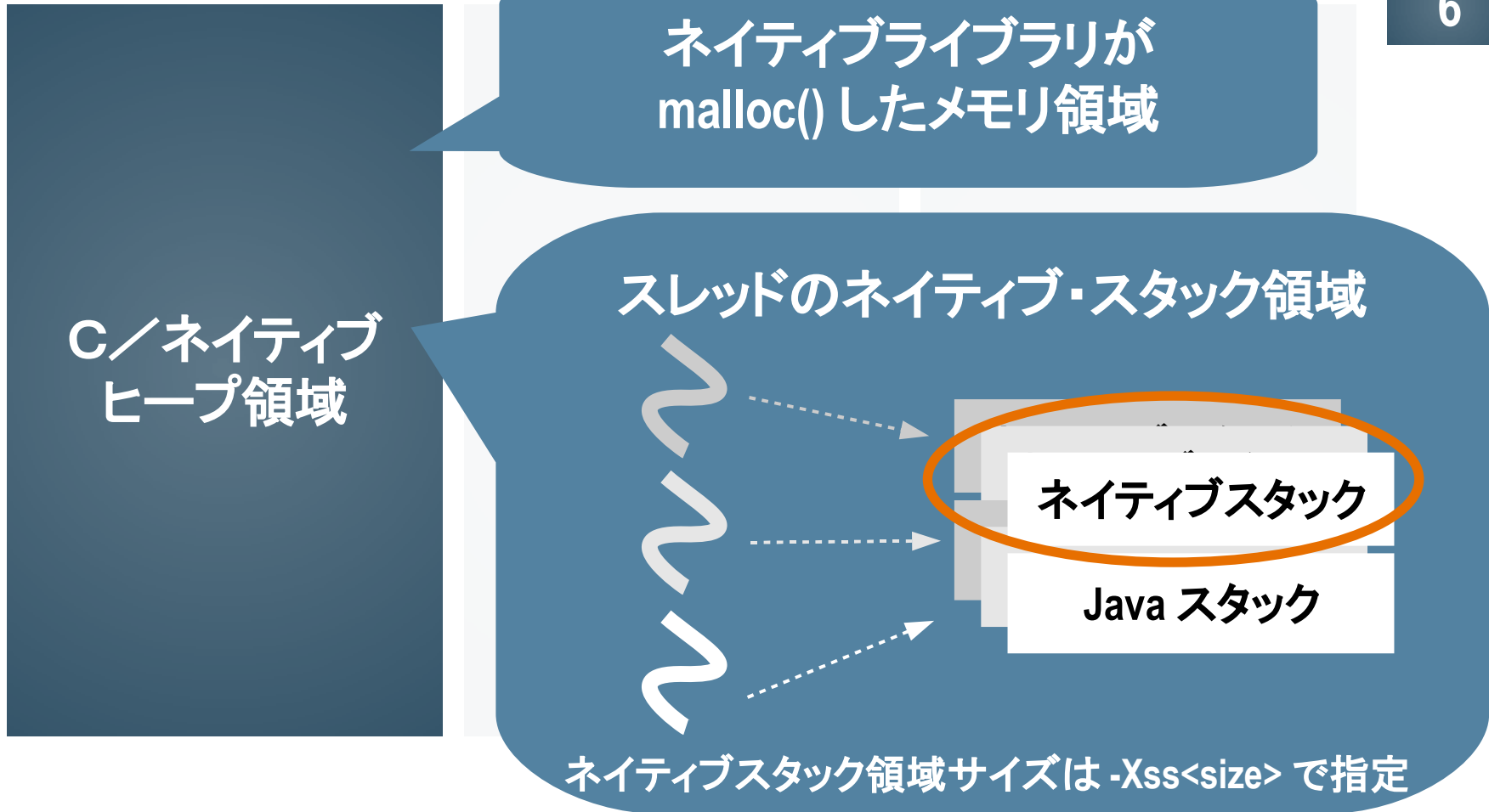
Java コード
PermGen 領域

Java VM におけるメモリ領域

1.4.2

5.0

6



Java VM におけるメモリ領域

1.4.2

5.0

6

C/ネイティブ
ヒープ領域

Java ヒープ領域

Java コード
PermGen 領域

Java オブジェクトのイメージが
格納される領域

スレッドの Java スタック領域

Java スタック領域サイズは `-Xoss<size>` で指定

Java VM におけるメモリ領域

1.4.2

5.0

6

C/ネイティブ
ヒープ領域

Java ヒープ領域

Java コード
PermGen 領域

クラス定義やメソッド定義など

PermGen 領域サイズは `-XX:PermSize=<size>` および
`-XX:MaxPermSize=<size>` で指定

OutOfMemoryError になる状況

1.4.2

5.0

6

どれかひとつの領域でも、
これ以上拡張できなくなった場合

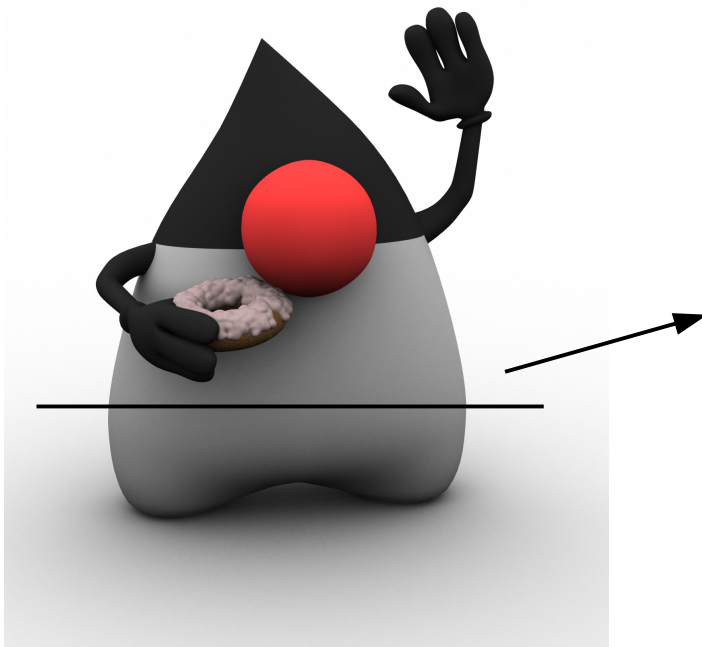
C/ネイティブ
ヒープ領域

Java ヒープ領域

Java コード
PermGen 領域

メモリ不足回避のためのアクション

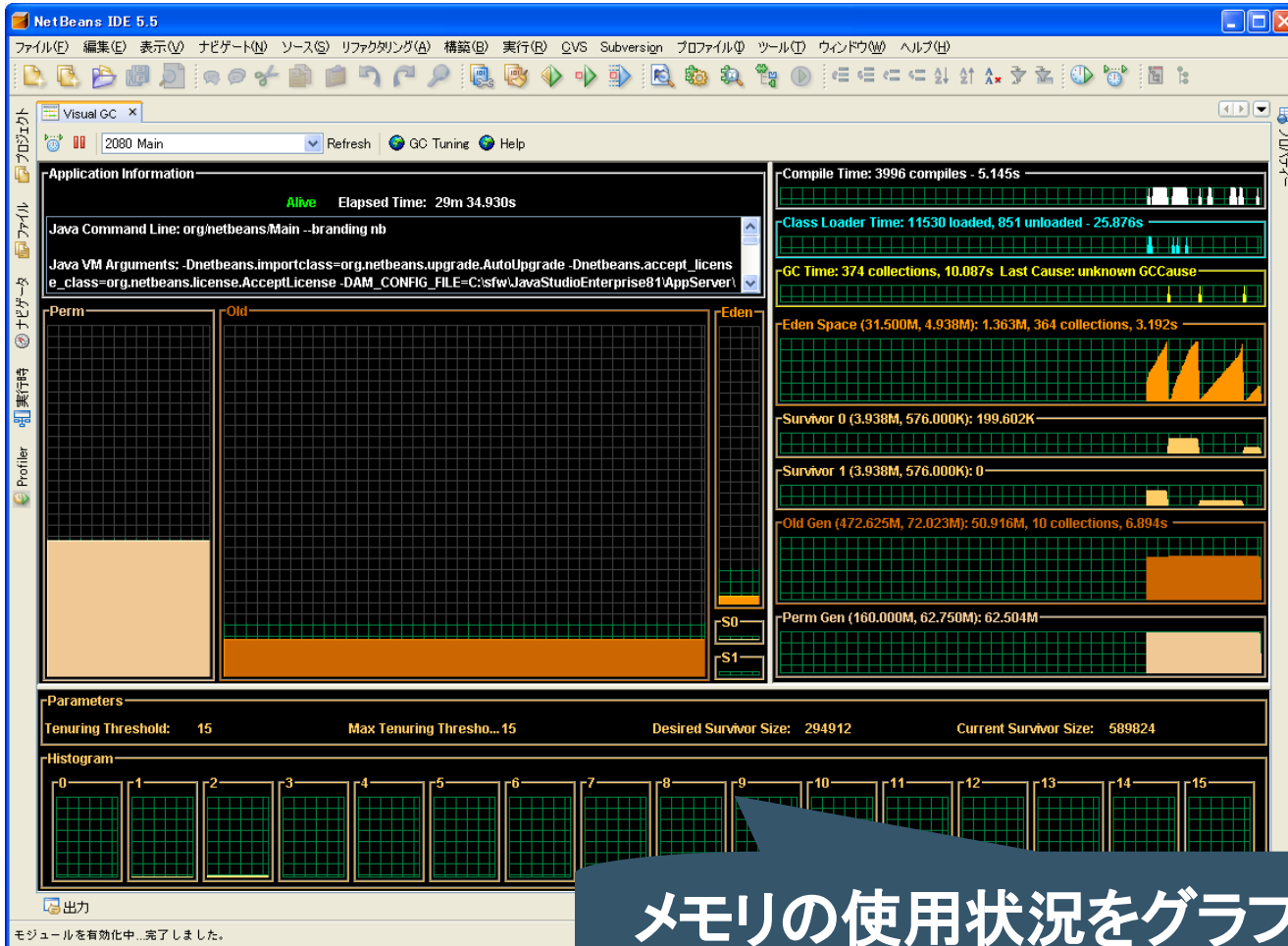
内部がどうなっているのかを
モニタリングする



内臓脂肪が...

VisualGC <http://java.sun.com/performance/jvmstat/>

1.4.2
5.0
6



JRE 1.4.2 では -XX:+UsePerfData の付加が必要

jstat

1.4.2

5.0

6

Visual GC のコマンドライン版

```
$ jstat -gcutil 2080 1000
```

S0	S1	E	O	P	YGC	YGCT	FGC	FGCT	GCT
40.63	0.00	52.78	70.81	99.62	392	3.390	10	6.894	10.284
40.63	0.00	56.92	70.81	99.62	392	3.390	10	6.894	10.284
40.63	0.00	61.10	70.81	99.62	392	3.390	10	6.894	10.284
40.63	0.00	64.65	70.81	99.62	392	3.390	10	6.894	10.284
40.63	0.00	70.04	70.81	99.62	392	3.390	10	6.894	10.284

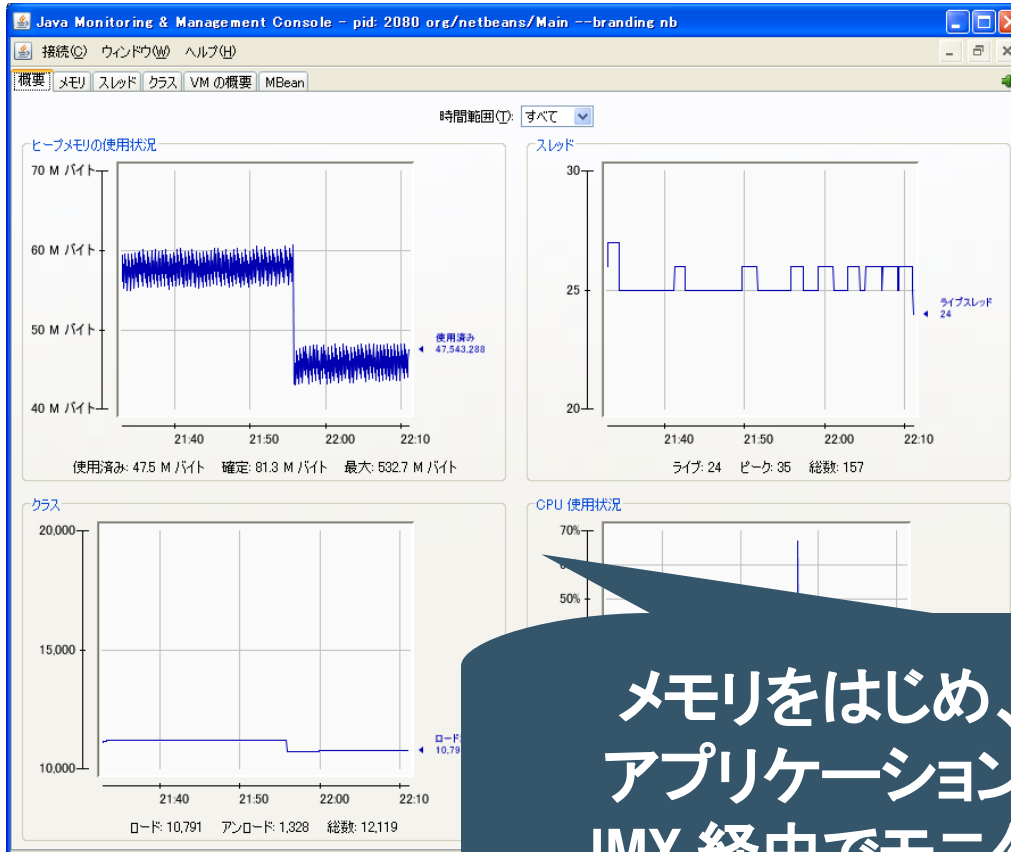
**JRE 1.4.2 では -XX:+UsePerfData の付加が必要
JDK 6 より正式ツールに昇格**

jconsole

1.4.2

5.0

6



メモリをはじめ、Java VM およびアプリケーションの様々な情報を JMX 経由でモニタリング・管理可能

JRE 5.0 では `-Dcom.sun.management.jmxremote` の付加が必要
 JRE 6 では同一マシン、同一ユーザの場合オプション不要

JRE 5.0 以降における改善

1.4.2

5.0

6

Hello, 3642

Hello, 3643

Hello, 3644

Exception in thread "main" java.lang.OutOfMemoryError: PermGen space

C/ネイティブ
ヒープ領域

どの領域で OutOfMemoryError となったか
表示されるように改善

Java SE 6 以降における改善

1.4.2

5.0

6

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
  at java.util.Arrays.copyOf(Arrays.java:2882)
  at java.lang.AbstractStringBuilder.expandCapacity(AbstractStringBuilder.java:100)
  at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:390)
  at java.lang.StringBuffer.append(StringBuffer.java:224)
  at JavaNightSeminar.main(JavaNightSeminar.java:6)
```

スタックトレースも表示されるように

-XX:+HeapDumpOnOutOfMemoryError

1.4.2

5.0

6

```

Hello, 2667
Hello, 2668
java.lang.OutOfMemoryError: PermGen space
Dumping heap to java_pid148.hprof ...
Heap dump file created [3771471 bytes in 0.200 secs]
Exception in thread "main" java.lang.OutOfMemoryError: PermGen space
    
```

C/ネイティブ
ヒープ領域

OutOfMemoryError 発生時にヒープの
ダンプを生成 (後述のツールで解析)

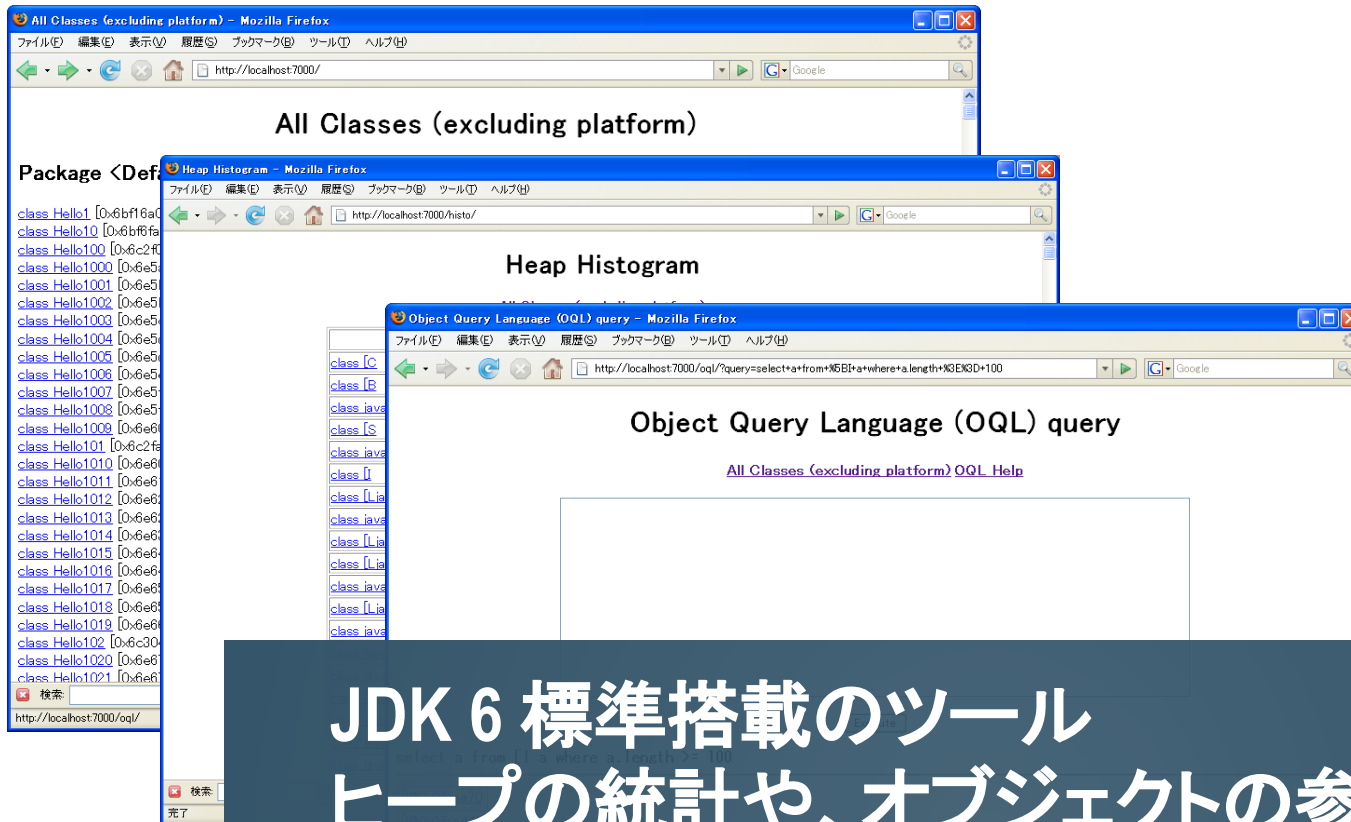
JRE 1.4.2_12 以降、JRE 1.5.0_07 以降、JRE 1.6.0 以降でサポート

jhat (Java Heap Analysis Tool)

1.4.2

5.0

6



**JDK 6 標準搭載のツール
ヒープの統計や、オブジェクトの参照関係を
解析するためのツール
(1.4.2 のダンプも読み込み可能)**

NetBeans 5.5 Profiler

1.4.2
5.0
6

The screenshot shows the NetBeans IDE 5.5 Profiler interface. The main window displays a table of memory usage statistics for various classes. The table has four columns: Class Name, Memory Used, Number of Instances, and Object Count. The 'char []' class is highlighted in red, indicating it is the most memory-intensive class.

クラス名 - 実例当てオブジェクト	実例当てバイト	実例当てバイト数	実例当てオブジェ...
char []	38,304 B (94.5%)	1,269	023.5%
byte []	11,760 B (16.7%)	12	0.2%
java.lang.String	2,256 B (3.2%)	895	0.6%
java.lang.Object []	1,808 B (2.6%)	290	0.4%
java.util.HashMap\$Entry	1,320 B (0.9%)	542	0.01%
com.sun.org.apache.xerces.internal.xni.QName	864 B (0.2%)	333	0.2%
java.util.Hashtable\$Entry	864 B (0.2%)	342	0.4%
int []	848 B (0.2%)	53	0.1%
com.sun.org.apache.xerces.internal.util.SymbolTable\$Entry []	704 B (0.1%)	5	0.01%
java.lang.String []	664 B (0.9%)	114	0.2%
java.util.Hashtable\$Entry []	600 B (0.9%)	74	0.4%
java.util.HashMap\$Entry []	592 B (0.8%)	41	0.08%
java.lang.Object []	392 B (0.6%)	1	0%

Below the table, there is a 'VM 遠隔測定' (VM Remote Profiling) window showing a graph of memory usage over time. The graph has a y-axis from 0 to 4 and an x-axis from 00:00:00 to 00:00:40. A red line shows the memory usage starting at 0, rising to 4, and then fluctuating between 2 and 4.

ヒープの使用状況や、インスタンスの生成・消滅状況に関する統計を取得し問題箇所を切り分け

NetBeans 6.0 Profiler

1.4.2

5.0

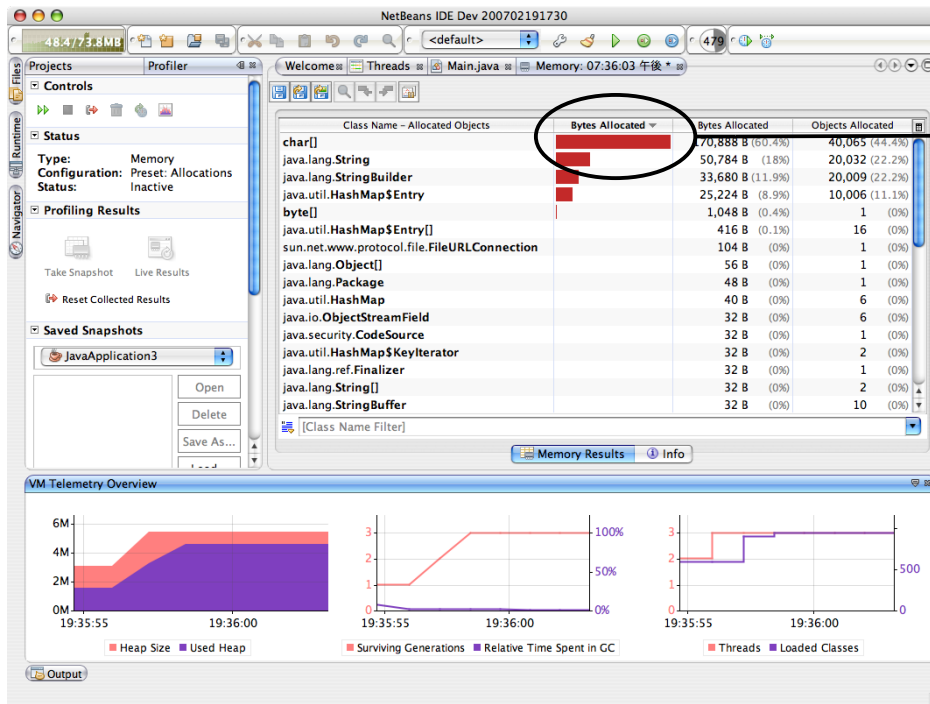
6



現在開発中の新バージョン
 ヒープのダンプを読み込んで、オブジェクトの
 参照関係を分析可能

メモリリークの見つけ方

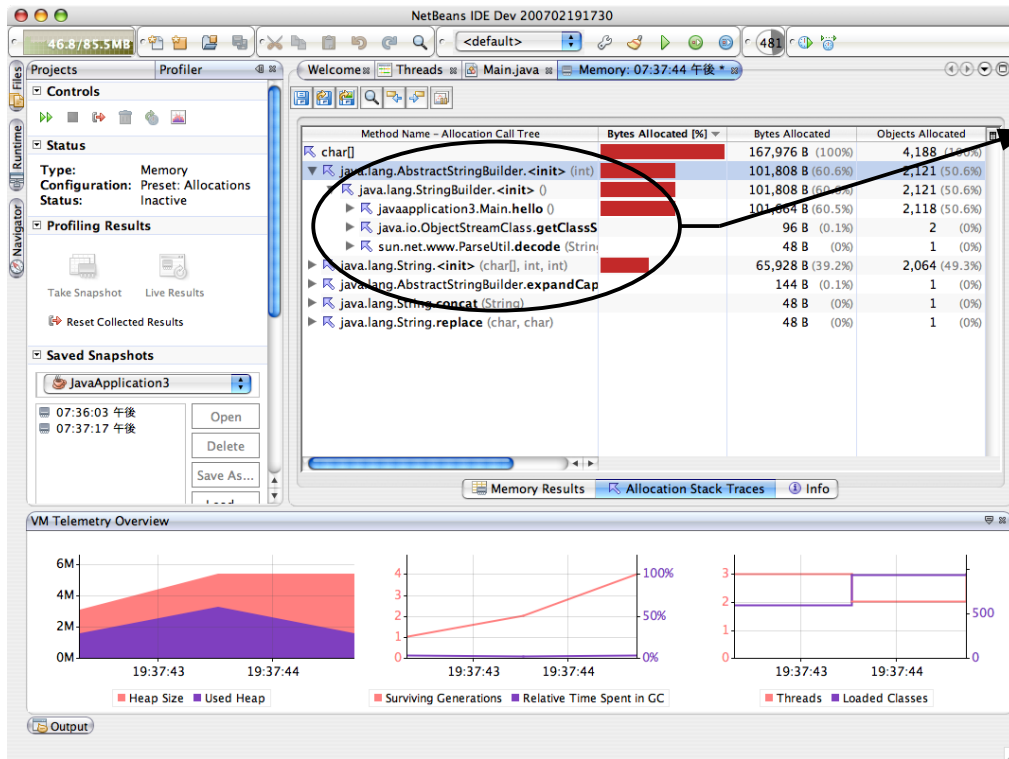
統計情報からメモリを占拠しているインスタンスの種類を判別



例) char[] があやしい

メモリーリークの見つけ方 (続き)

リークを引き起こしている可能性のあるソースを分析



バックトレースを確認して
怪しいインスタンスを作っている
ソースを探る

メモリーリークの見つけ方 (続き)

インスタンスに対する参照を追跡

不要な
インスタンス

参照関係があると
GC で回収されない

他の
インスタンス

The screenshot shows the NetBeans IDE Profiler interface. The 'Instances' tab is selected, displaying a list of instances for the class `java.util.TreeMap$Entry`. The 'References' tab is also visible, showing a tree structure of references. A red circle highlights a specific reference path: `parent` (instance #7) pointing to `left (loop to this)` (instance #1), which points to `parent` (instance #19), which points to `left (loop to t)` (instance #7), which points to `parent` (instance #22), which points to `left (loop)` (instance #19), which points to `parent` (instance #91), which points to `left (lo` (instance #43), which points to `parent` (instance #179), which points to `left` (instance #91), which points to `parent` (instance #10n).

Instance	Field	Type	Value
#1	this	TreeMap\$Entry	#1
#2	color	boolean	true
#3	parent	TreeMap\$Entry	#743
#4	right	TreeMap\$Entry	#8
#5	left	TreeMap\$Entry	#7
#6	value	String	#378
#7	key	<object>	null

ドリル3:まとめ

- OutOfMemoryError になる仕組みを把握しておく
- JVM の機能やツールの使い方をマスターしておく
- 適度な運動をして肥大化しないようにしておく

Q. コードレビューで大事な事

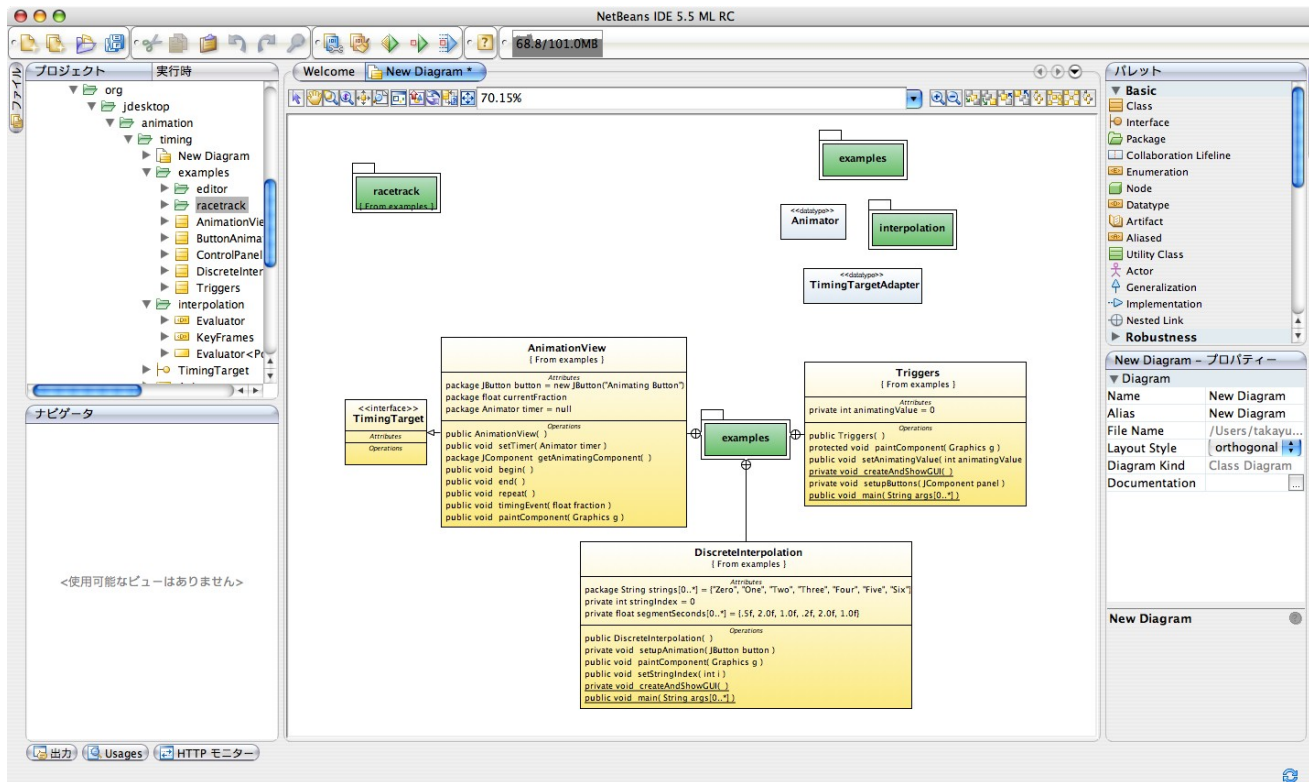
1. 重箱の隅をつつくようにする
2. 事前にある程度当たりを付けておく
3. ツールを活用する
4. 大量印刷の準備
5. 忍耐

事前に当たりを付けておく

- 症状や過去の事例から怪しい部分を特定しておく
- UML リバースエンジニアリングを使って概観を把握
- コードチェックツールを活用する
 - > PMD (<http://pmd.sourceforge.net>)
 - > Checkstyle (<http://checkstyle.sourceforge.net>)
 - > FindBugs (<http://findbugs.sourceforge.net>)

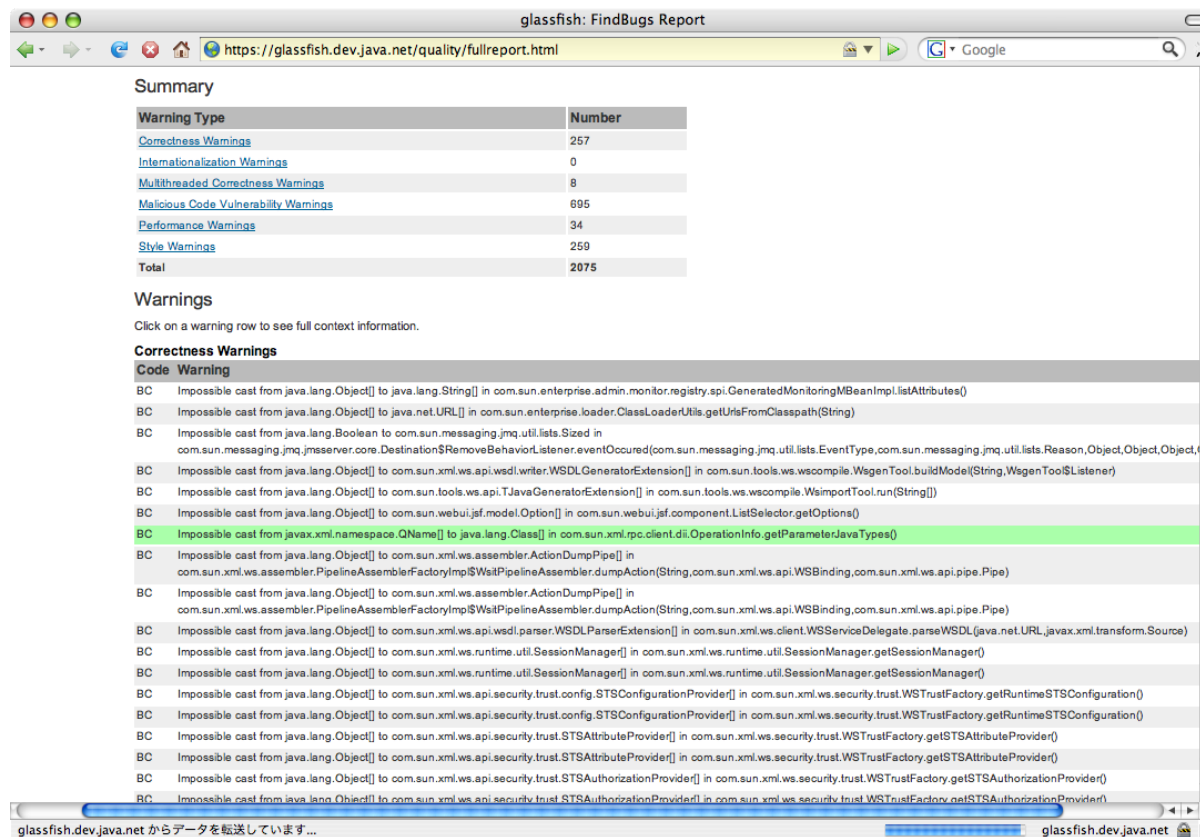
UML リバースエンジニアリング

- NetBeans 5.5 Enterprise Pack or UML モジュール (<http://ja.netbeans.org>)



コードチェックツール

- 設定したルールに従ってソースコードをスキャン
- 網羅性があり、一貫したポリシーで検査可能



Summary

Warning Type	Number
Correctness Warnings	257
Internationalization Warnings	0
Multithreaded Correctness Warnings	8
Malicious Code Vulnerability Warnings	695
Performance Warnings	34
Style Warnings	259
Total	2075

Warnings

Click on a warning row to see full context information.

Correctness Warnings

Code	Warning
BC	Impossible cast from java.lang.Object[] to java.lang.String[] in com.sun.enterprise.admin.monitor.registry.api.GeneratedMonitoringMBeanImpl.listAttributes()
BC	Impossible cast from java.lang.Object[] to java.net.URL[] in com.sun.enterprise.loader.ClassLoaderUtils.getUrlsFromClasspath(String)
BC	Impossible cast from java.lang.Boolean to com.sun.messaging.jmq.util.lists.Sized in com.sun.messaging.jmq.msserver.core.Destination\$RemoveBehaviorListener.eventOccured(com.sun.messaging.jmq.util.lists.EventTopic,com.sun.messaging.jmq.util.lists.Reason,Object,Object,Object)
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.wSDL.writer.WSDLGeneratorExtension[] in com.sun.tools.ws.wscmcompile.WegenTool.buildModel(String,WegenTool\$Listener)
BC	Impossible cast from java.lang.Object[] to com.sun.tools.ws.api.TJavaGeneratorExtension[] in com.sun.tools.ws.wscmcompile.WsimportTool.run(String[])
BC	Impossible cast from java.lang.Object[] to com.sun.webui.jsf.model.Option[] in com.sun.webui.jsf.component.ListSelector.getOptions()
BC	Impossible cast from javax.xml.namespace.QName[] to java.lang.Class[] in com.sun.xml.rpc.client.di.OperationInfo.getParameterJavaTypes()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.assembler.ActionDumpPipe[] in com.sun.xml.ws.assembler.PipelineAssemblerFactoryImp\$WaitPipelineAssembler.dumpAction(String,com.sun.xml.ws.api.WSBinding,com.sun.xml.ws.api.pipe.Pipe)
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.assembler.ActionDumpPipe[] in com.sun.xml.ws.assembler.PipelineAssemblerFactoryImp\$WaitPipelineAssembler.dumpAction(String,com.sun.xml.ws.api.WSBinding,com.sun.xml.ws.api.pipe.Pipe)
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.wSDL.parser.WSDLParseExtension[] in com.sun.xml.ws.client.WSServiceDelegate.parseWSDL(java.net.URL,java.xml.transform.Source)
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.runtime.util.SessionManager[] in com.sun.xml.ws.runtime.util.SessionManager.getSessionManager()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.runtime.util.SessionManager[] in com.sun.xml.ws.runtime.util.SessionManager.getSessionManager()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.config.STSConfigurationProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getRuntimeSTSConfiguration()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.config.STSConfigurationProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getRuntimeSTSConfiguration()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.STSAttributeProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getSTSAttributeProvider()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.STSAttributeProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getSTSAttributeProvider()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.STSAttributeProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getSTSAttributeProvider()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.STSAuthorizationProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getSTSAuthorizationProvider()
BC	Impossible cast from java.lang.Object[] to com.sun.xml.ws.api.security.trust.STSAuthorizationProvider[] in com.sun.xml.ws.security.trust.WSTrustFactory.getSTSAuthorizationProvider()

glassfish.dev.java.net からデータを転送しています...

Q1. バグを探してください

```
for (int i = 0; i < 10; i++) {  
    for (int j = 0; j < 20; i++) {  
        total += a[i][j];  
    }  
}
```

Q2. バグを探してください

```
public class Seminar {  
    private String name;  
    public boolean equals(Seminar s) {  
        return name.equals(s.name)  
            && name != null;  
    }  
}
```

大量のコードを解析するとき

- IDE の機能は便利だけれど重すぎる
- コード検索エンジンや HTML 化を活用
 - > 大量のコードを”ブラウズ”という意味では Web ブラウザは使いやすい
 - > 専用ソフトは検索もできて高速
- コマンドラインの強みは健在
 - > grep, find, awk, perl などなど
 - > IDE の機能になくても自作する (なるべくコンピュータに仕事をさせる)

コード検索 : OpenGrok

- OpenGrok
 - > <http://www.opensolaris.org/os/project/opengrok>
 - > OpenSolaris プロジェクトで作成された検索／クロスリファレンスエンジン
 - > Java だけでなく、C/C++、シェルスクリプトなどにも対応
 - > 導入がかんたん

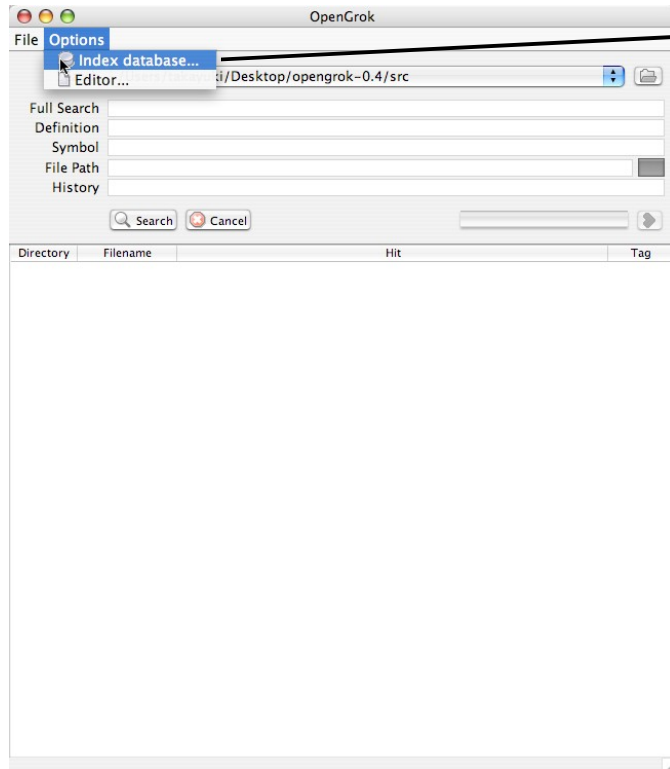
{OpenGrok

OpenGrok の導入

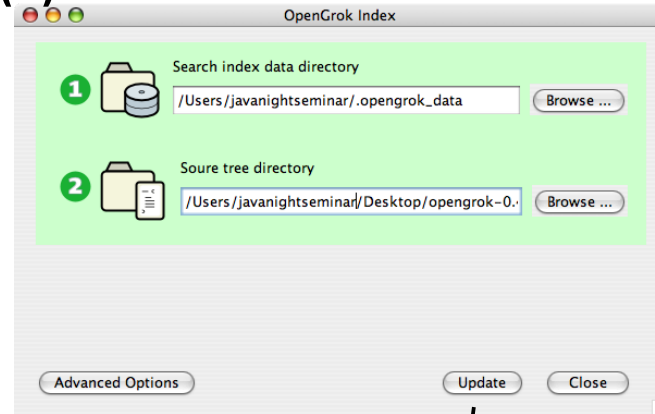
- 必要ファイルをダウンロード
 - > opengrok-0.4.tar.gz ... OpenGrok 本体とソース
 - > <http://www.opensolaris.org/os/project/opengrok>
 - > Exuberant ctags (<http://ctags.sourceforge.net>)
 - > Java5.0 以上
 - > Tomcat か GlassFish
 - > Subversion 1.3.0 (Subversion を使う場合のみ)
 - > Mercurial (Mercurial を使う場合のみ)

OpenGrok インデックスの作成

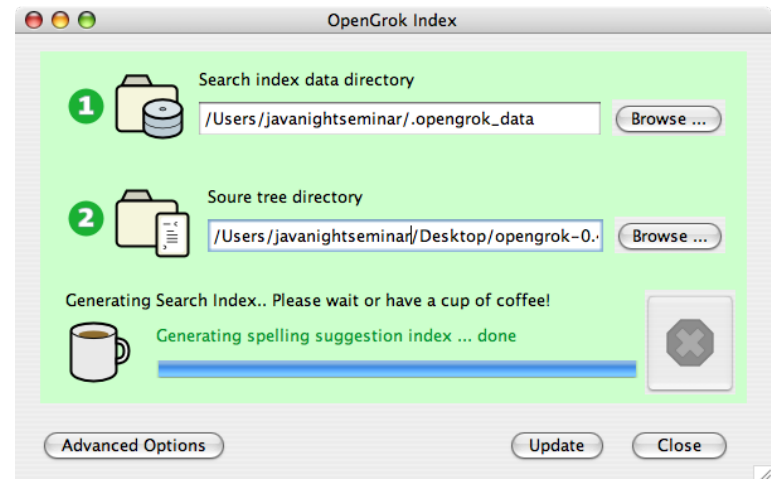
(1) 起動して ..



(2) ソースとインデックス格納先を選択

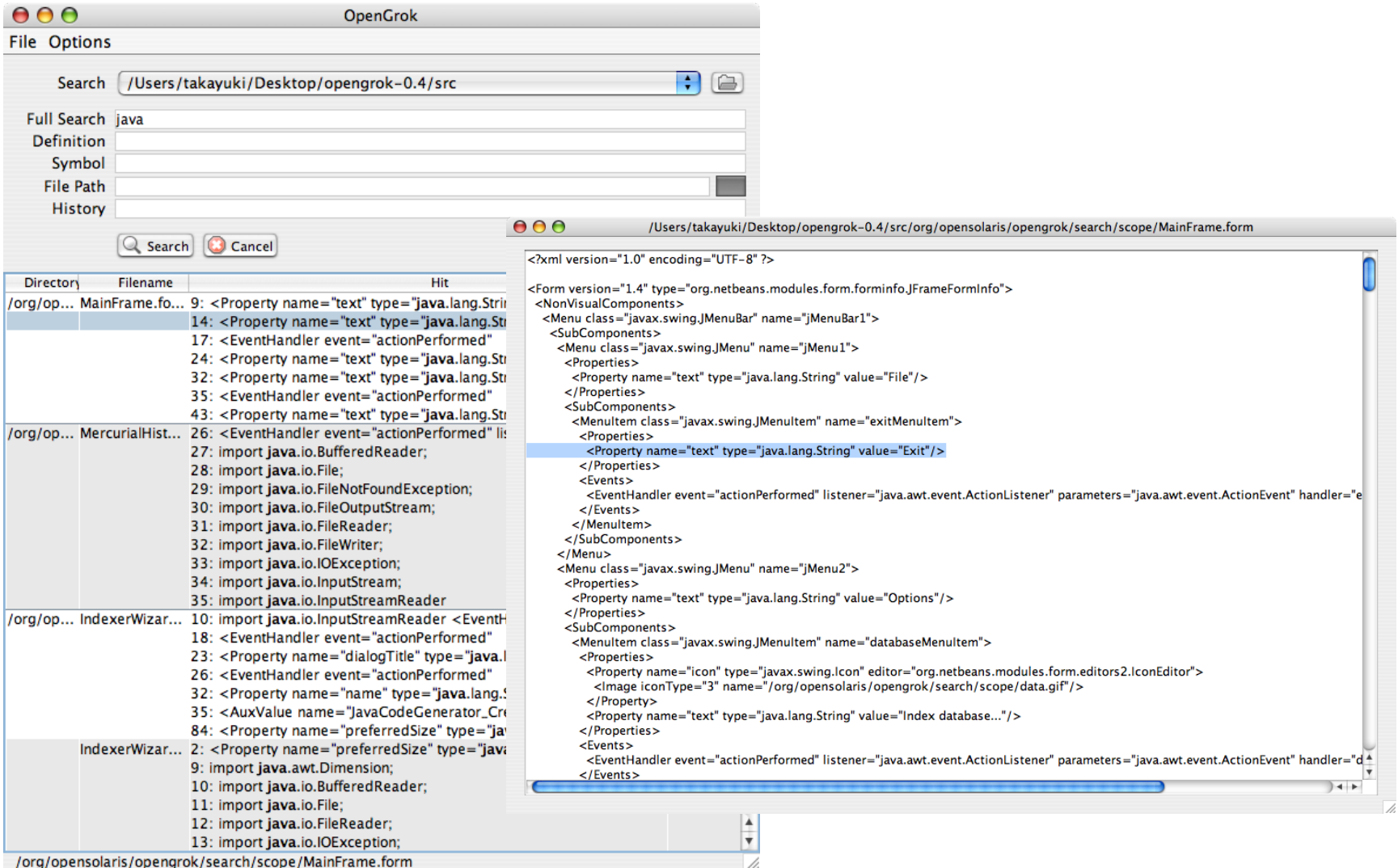


(3) Update して完了



(opengrok.jar を実行)

OpenGrok ソースの検索 (Swing)



The screenshot shows the OpenGrok application interface. The top window displays search options and a list of search results. The bottom window shows the XML source code for the selected file.

OpenGrok Search Results:

Directory	Filename	Hit
/org/op...	MainFrame.fo...	9: <Property name="text" type="java.lang.Stri 14: <Property name="text" type="java.lang.St 17: <EventHandler event="actionPerformed" 24: <Property name="text" type="java.lang.St 32: <Property name="text" type="java.lang.St 35: <EventHandler event="actionPerformed" 43: <Property name="text" type="java.lang.St
/org/op...	MercurialHist...	26: <EventHandler event="actionPerformed" li 27: import java.io.BufferedReader; 28: import java.io.File; 29: import java.io.FileNotFoundException; 30: import java.io.OutputStream; 31: import java.io.FileReader; 32: import java.io.FileWriter; 33: import java.io.IOException; 34: import java.io.InputStream; 35: import java.io.InputStreamReader
/org/op...	IndexerWizar...	10: import java.io.InputStreamReader <Event 18: <EventHandler event="actionPerformed" 23: <Property name="dialogTitle" type="java.l 26: <EventHandler event="actionPerformed" 32: <Property name="name" type="java.lang.! 35: <AuxValue name="JavaCodeGenerator_Cri 84: <Property name="preferredSize" type="ja
	IndexerWizar...	2: <Property name="preferredSize" type="javi 9: import java.awt.Dimension; 10: import java.io.BufferedReader; 11: import java.io.File; 12: import java.io.FileReader; 13: import java.io.IOException;

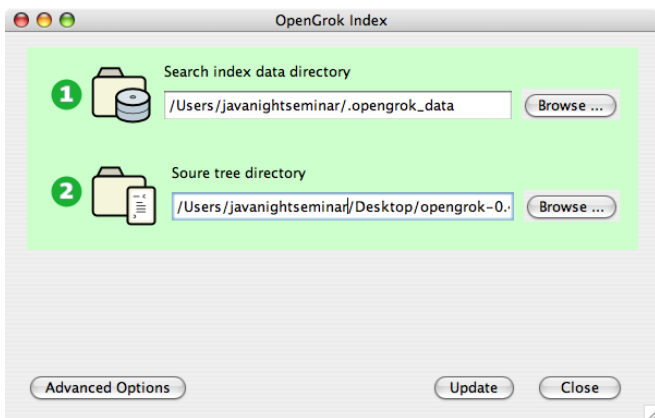
XML Source Code (MainFrame.form):

```

<?xml version="1.0" encoding="UTF-8" ?>
<Form version="1.4" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
<NonVisualComponents>
  <Menu class="javax.swing.JMenuBar" name="jMenuBar1">
    <SubComponents>
      <Menu class="javax.swing.JMenu" name="jMenu1">
        <Properties>
          <Property name="text" type="java.lang.String" value="File"/>
        </Properties>
        <SubComponents>
          <MenuItem class="javax.swing.JMenuItem" name="exitMenuItem">
            <Properties>
              <Property name="text" type="java.lang.String" value="Exit"/>
            </Properties>
            <Events>
              <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="d
            </Events>
          </MenuItem>
        </SubComponents>
      </Menu>
      <Menu class="javax.swing.JMenu" name="jMenu2">
        <Properties>
          <Property name="text" type="java.lang.String" value="Options"/>
        </Properties>
        <SubComponents>
          <MenuItem class="javax.swing.JMenuItem" name="databaseMenuItem">
            <Properties>
              <Property name="icon" type="javax.swing.Icon" editor="org.netbeans.modules.form.editors2.IconEditor">
                <Image iconType="3" name="/org/opensolaris/opengrok/search/scope/data.gif"/>
              </Property>
              <Property name="text" type="java.lang.String" value="Index database..."/>
            </Properties>
            <Events>
              <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="d
            </Events>
          </MenuItem>
        </SubComponents>
      </Menu>
    </SubComponents>
  </Menu>

```

OpenGrok ソースの検索準備 (Web)



配布ファイル source.war の中にある WEB-INF/web.xml に
インデックスを作成したディレクトリ (DATA_ROOT) と
ソースコードのディレクトリ (SRC_ROOT) を記述してデプロイ

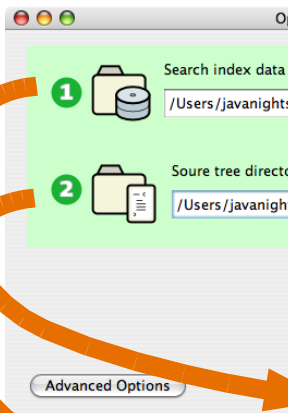
OpenGrok ソースの検索準備 (Web)

```

<web-app>
  <display-name>OpenGrok</display-name>
  <description>A wicked fast source browser</description>

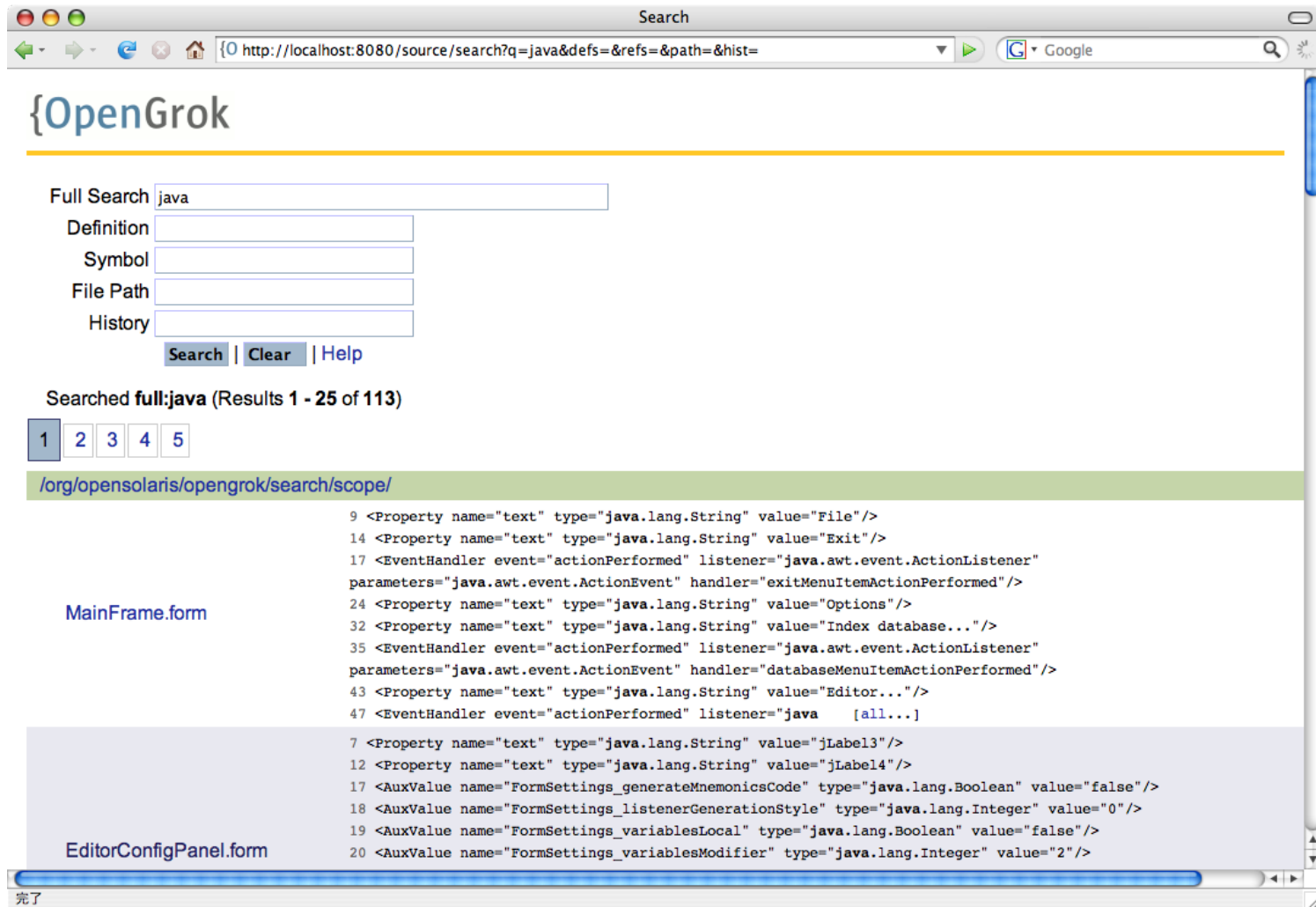
  <context-param>
    <param-name>DATA_ROOT</param-name>
    <param-value>/directory/containing/data</param-value>
  </context-param>
  <context-param>
    <param-name>SRC_ROOT</param-name>
    <param-value>/directory/containing/src</param-value>
  </context-param>

```



インデック
ソースコー

OpenGrok ソースの検索 (web)



Search

http://localhost:8080/source/search?q=java&defs=&refs=&path=&hist=

{OpenGrok}

Full Search

Definition

Symbol

File Path

History

[Search](#) | [Clear](#) | [Help](#)

Searched full:java (Results 1 - 25 of 113)

1 2 3 4 5

</org/opensolaris/opengrok/search/scope/>

MainFrame.form

```

9 <Property name="text" type="java.lang.String" value="File"/>
14 <Property name="text" type="java.lang.String" value="Exit"/>
17 <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent" handler="exitMenuItemActionPerformed"/>
24 <Property name="text" type="java.lang.String" value="Options"/>
32 <Property name="text" type="java.lang.String" value="Index database..."/>
35 <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent" handler="databaseMenuItemActionPerformed"/>
43 <Property name="text" type="java.lang.String" value="Editor..."/>
47 <EventHandler event="actionPerformed" listener="java [all...]
```

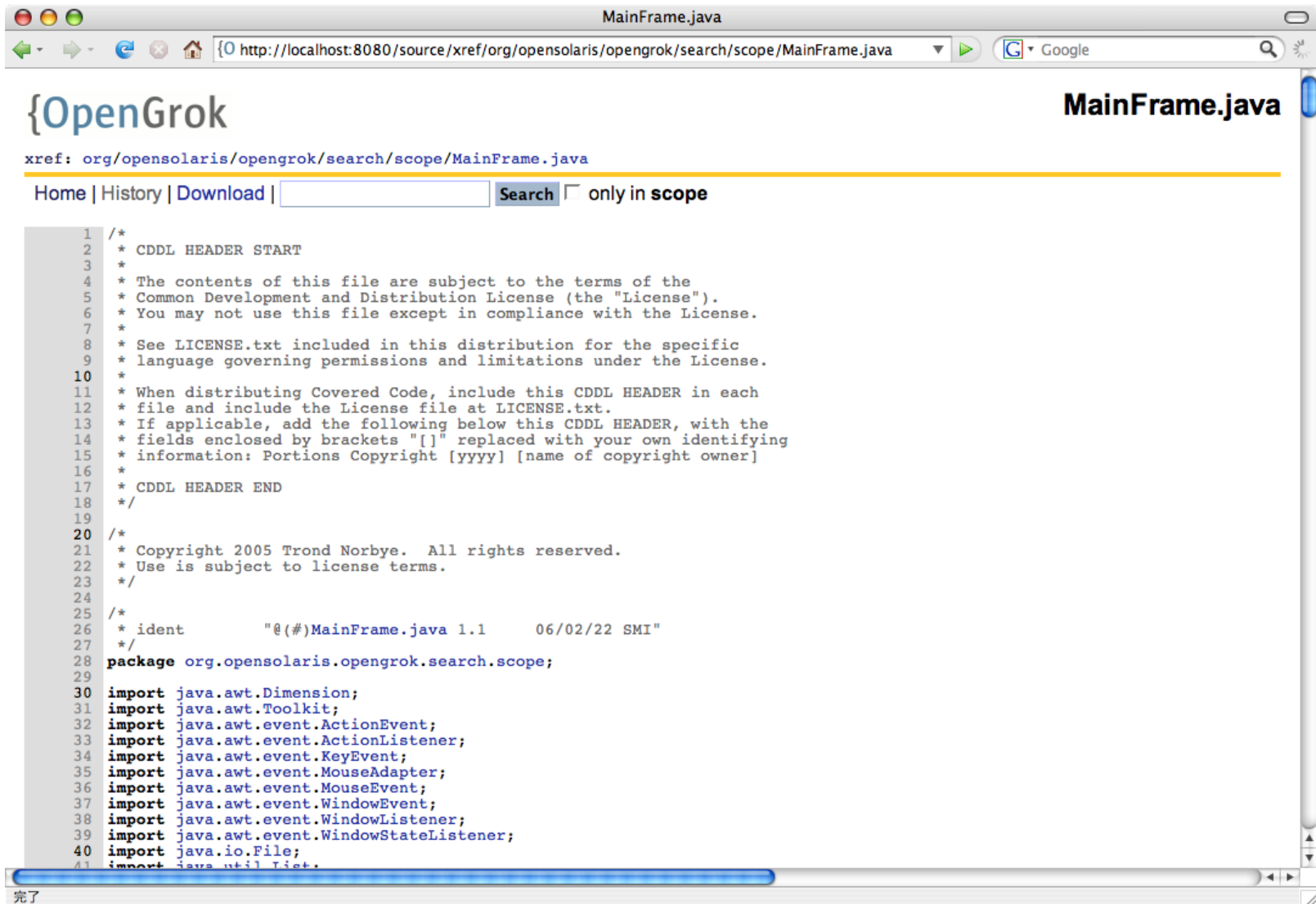
EditorConfigPanel.form

```

7 <Property name="text" type="java.lang.String" value="jLabel3"/>
12 <Property name="text" type="java.lang.String" value="jLabel4"/>
17 <AuxValue name="FormSettings_generateMnemonicsCode" type="java.lang.Boolean" value="false"/>
18 <AuxValue name="FormSettings_listenerGenerationStyle" type="java.lang.Integer" value="0"/>
19 <AuxValue name="FormSettings_variablesLocal" type="java.lang.Boolean" value="false"/>
20 <AuxValue name="FormSettings_variablesModifier" type="java.lang.Integer" value="2"/>
```

完了

OpenGrok ソースの閲覧 (web)



Browser window title: MainFrame.java

Address bar: http://localhost:8080/source/xref/org/opensolaris/opengrok/search/scope/MainFrame.java

Page title: {OpenGrok} MainFrame.java

Breadcrumb: xref: org/opensolaris/opengrok/search/scope/MainFrame.java

Navigation: Home | History | Download | Search only in scope

```

1  /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * See LICENSE.txt included in this distribution for the specific
9  * language governing permissions and limitations under the License.
10 *
11 * When distributing Covered Code, include this CDDL HEADER in each
12 * file and include the License file at LICENSE.txt.
13 * If applicable, add the following below this CDDL HEADER, with the
14 * fields enclosed by brackets "[]" replaced with your own identifying
15 * information: Portions Copyright [yyyy] [name of copyright owner]
16 *
17 * CDDL HEADER END
18 */
19
20 /*
21 * Copyright 2005 Trond Norbye. All rights reserved.
22 * Use is subject to license terms.
23 */
24
25 /*
26 * ident      "@(#)MainFrame.java 1.1      06/02/22 SMI"
27 */
28 package org.opensolaris.opengrok.search.scope;
29
30 import java.awt.Dimension;
31 import java.awt.Toolkit;
32 import java.awt.event.ActionEvent;
33 import java.awt.event.ActionListener;
34 import java.awt.event.KeyEvent;
35 import java.awt.event.MouseAdapter;
36 import java.awt.event.MouseEvent;
37 import java.awt.event.WindowEvent;
38 import java.awt.event.WindowListener;
39 import java.awt.event.WindowStateListener;
40 import java.io.File;
41 import java.util.List;

```

完了

ドリル4:まとめ

- いきなり細かく読み始める前に全体を把握しておく
- 目的に適したツールを使う（例：チェックツール、閲覧専用ソフト）

ドリル 5:

速さは、力



Q. チューニングで大事な事

1. トライ & エラー
2. コメントを削ってソースを軽くする
3. なるべくインスタンスを再利用するようにコードを書き換える
4. 統計データを取得する

統計を取って狙いを定める

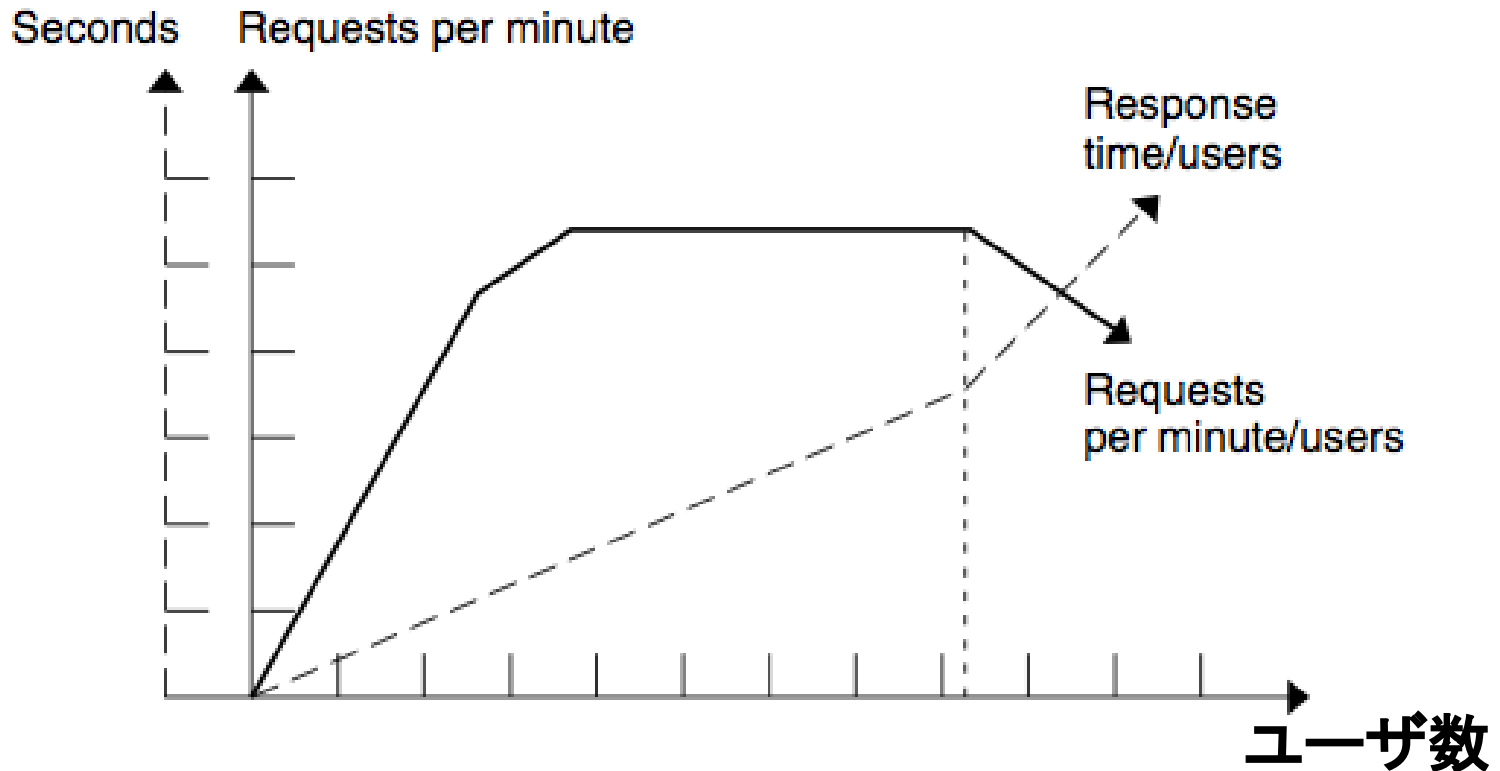
- むやみなトライ & エラーは時間の無駄
- 統計を取って効果のありそうなところを探す
 - > 重い処理
 - > 無駄な処理
 - > 頻繁に呼び出される処理
 - > I/O 待ちや他システムの連携待ち

レイテンシとスループット

- チューニング目標は注意深く設定する
 - > レイテンシ (平均応答時間)
 - > 例 . 平均応答 700ms 以内
 - > スループット (単位時間あたりの処理量)
 - > 例 . 1 分間あたり平均 1300 トランザクション

レイテンシとスループットの関係

- ユーザ数の増加とともにレイテンシは増加
- あるユーザ数を境にスループットは低下



プロファイリング : NetBeans Profiler

The screenshot displays the NetBeans IDE Profiler interface. The main window shows a list of Hot Spots with columns for Method, Self time [%], Self time, and Invocations. The 'Hot Spots - Method' table is as follows:

Hot Spots - Method	Self time [%]	Self time	Invocations
com.sun.org.apache.commons.logging.impl.Jdk14Logger.isDe	20.8%	1.12 ms	1
org.apache.jsp.index_jsp_jspService (javax.servlet.http.Htt...	11.2%	0.608 ms	1
org.apache.coyote.Response.setContentType (String)	2.3%	0.125 ms	1
sun.nio.cs.UTF_8\$Decoder.decodeArrayLoop (java.nio.Byte...	1.7%	0.093 ms	1
java.lang.ClassLoader.findLoadedClass (String)	1.7%	0.090 ms	15
java.lang.ThreadLocal\$ThreadLocalMap.getAfterMiss (Threa...	1.6%	0.086 ms	1
org.apache.jasper.runtime.JspWriterImpl.write (String, int, int)	1.4%	0.077 ms	21
java.nio.charset.CharsetEncoder.isLegalReplacement (byte[])	1.4%	0.076 ms	1
java.nio.charset.CharsetDecoder.decode (java.nio.ByteBuffer...	1.2%	0.066 ms	1
org.apache.jasper.runtime.JspWriterImpl.<init> (javax.servle...	1.2%	0.065 ms	1
java.util.HashMap.put (Object, Object)	1.2%	0.063 ms	7
org.apache.catalina.loader.WebappClassLoader.loadClass (S...	1%	0.055 ms	5
org.apache.jasper.runtime.PageContextImpl._initialize (java...	1%	0.054 ms	1
java.util.HashMap.<init> (int, float)	1%	0.054 ms	1
org.apache.coyote.tomcat5.CoyoteResponseFacade.setConten	0.9%	0.048 ms	1
sun.nio.cs.UTF_8\$Decoder.decodeLoop (java.nio.ByteBuffer...	0.9%	0.048 ms	1
org.apache.coyote.tomcat5.OutputBuffer.checkConverter ()	0.9%	0.048 ms	2
org.apache.coyote.tomcat5.CoyoteResponse.setContentType	0.9%	0.047 ms	1
org.apache.jasper.runtime.JspFactoryImpl.internalGetPageCo	0.8%	0.046 ms	1
java.util.HashMap.get (Object)	0.8%	0.044 ms	8
java.lang.ThreadLocal\$ThreadLocalMap.get (ThreadLocal)	0.8%	0.044 ms	2
org.apache.coyote.tomcat5.OutputBuffer.setConverter ()	0.8%	0.044 ms	1
java.lang.ThreadLocal.get ()	0.8%	0.041 ms	2
org.apache.jasper.servlet.JasperLoader.loadClass (String, bo...	0.7%	0.039 ms	5
org.apache.jasper.runtime.PageContextImpl.release ()	0.7%	0.038 ms	1
java.nio.ByteBuffer.<init> (int, int, int, int, byte[], int)	0.7%	0.037 ms	2
org.apache.coyote.tomcat5.CoyoteResponse.isAppCommittee	0.7%	0.036 ms	3
com.sun.org.apache.commons.logging.impl.Jdk14Logger.isTr	0.7%	0.036 ms	15
sun.nio.cs.StreamEncoder\$CharsetSE.<init> (java.io.Output...	0.6%	0.035 ms	1
org.apache.tomcat.util.buf.MessageBytes.equalsIgnoreCase...	0.6%	0.034 ms	2
org.apache.jasper.runtime.JspWriterImpl.write (String)	0.6%	0.034 ms	21

The right-hand side of the interface shows the 'DrillDown' panel with a 'Scope: Overall' section. It contains a pie chart and a 'Method category distribution' section:

- Web container: 0.0%
- Overall(Self): 99.10%

The 'Method category distribution' section shows a bar chart for 'Web container (100.0%)'.

プロファイリング : NetBeans Profiler

The screenshot displays the NetBeans IDE Profiler interface. The main window shows the 'Call Tree - Method' view, which is a hierarchical tree of method calls. The 'Hot Spots - Method' view is also visible, showing a list of methods with their self-time percentages and invocation counts.

Call Tree - Method

Method	Time [%]	Time	Invocations
All threads		5.54 ms (100%)	1
httpSSLWorkerThread-8080-1		5.54 ms (100%)	1
org.apache.jsp.index_jsp._jspService (javax.servlet.http.HttpServletRequest, javax.servlet.http.Http...		5.55 ms (100%)	1
org.apache.jasper.runtime.JspFactoryImpl.releasePageContext (javax.servlet.jsp.PageContext)		2.58 ms (46.6%)	1
org.apache.jasper.runtime.JspFactoryImpl.internalReleasePageContext (javax.servlet.jsp.Pa...		2.56 ms (46.3%)	1
org.apache.jasper.runtime.PageContextImpl.release ()		2.52 ms (45.6%)	1
Self time		0.023 ms (0.4%)	1
java.util.LinkedList.addFirst (Object)		0.010 ms (0.2%)	1
java.lang.ThreadLocal.get ()		0.003 ms (0.1%)	1
Self time		0.016 ms (0.3%)	1
org.apache.jasper.runtime.JspFactoryImpl.getPageContext (javax.servlet.Servlet, javax.servle...		0.937 ms (16.9%)	1
Self time		0.608 ms (11%)	1
java.lang.ClassLoader.loadClassInternal (String)		0.607 ms (10.9%)	5
org.apache.coyote.tomcat5.CoyoteResponseFacade.setContentType (String)		0.397 ms (7.2%)	1

Hot Spots - Method

Method	Self time [%]	Self time	Invocations
java.lang.String.indexOf (String, int)		0.018 ms (0.3%)	1
java.lang.String.indexOf (int)		0.018 ms (0.3%)	17
org.apache.jasper.runtime.JspWriterImpl.ensureOpen ()		0.017 ms (0.3%)	25
org.apache.jasper.runtime.PageContextImpl.setAttribute (String, Object, int)		0.017 ms (0.3%)	1
java.lang.ThreadLocal\$ThreadLocalMap.resize ()		0.017 ms (0.3%)	1
sun.nio.cs.UTF_8\$Encoder.<init> (java.nio.charset.Charset)		0.016 ms (0.3%)	1
org.apache.jasper.runtime.JspWriterImpl.flushBuffer ()		0.016 ms (0.3%)	1
sun.nio.cs.StreamEncoder\$CharsetSE.<init> (java.io.OutputStream, Object, java.nio.charset.CharsetEnco...		0.016 ms (0.3%)	1
java.nio.HeapCharBuffer.<init> (int, int)		0.016 ms (0.3%)	1
sun.nio.cs.StreamEncoder.write (char[], int, int)		0.016 ms (0.3%)	1
java.util.HashSet.add (Object)		0.016 ms (0.3%)	5
java.lang.String.indexOf (int, int)		0.016 ms (0.3%)	17
org.apache.jasper.runtime.JspFactoryImpl.releasePageContext (javax.servlet.jsp.PageContext)		0.016 ms (0.3%)	1

Basic Telemetry

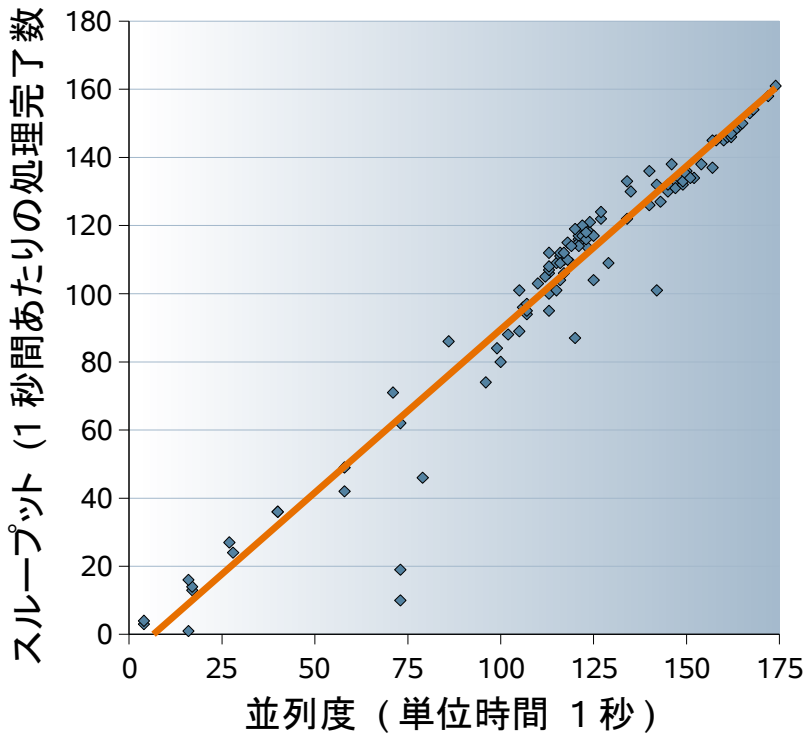
- Instrumented: 41,650 Methods
- Filter: Profile All Classes
- Threads: 42
- Total Memory: 46,878,720 B
- Used Memory: 23,263,512 B

JMeter でスループットを計測する

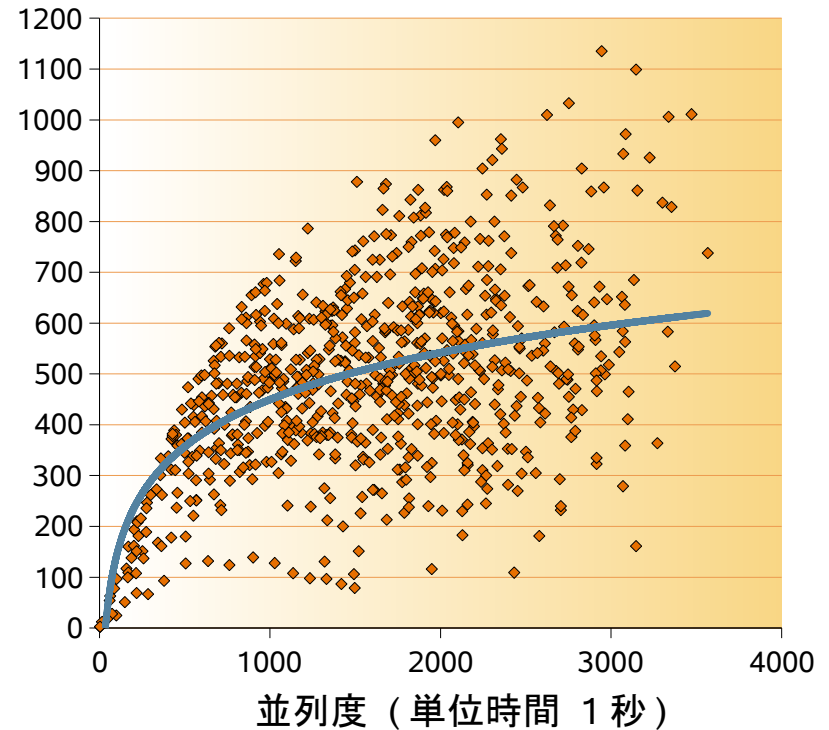
- 負荷生成 & レスポンスログ取得
 - ログを加工してデータを分析



ほとんどボトルネックがないケース



ボトルネックが現れ始めたケース



JMeter でスループットを計測する

- 詳しくは ...
 - > http://blogs.sun.com/okazaki/entry/sdc_performance_tuning_seminar
 - > http://blogs.sun.com/okazaki/entry/throughput_analysis_2
 - > http://blogs.sun.com/okazaki/entry/throughput_analysis_3

DTrace を使う

- ホワイトペーパーを参照
 - > <http://sdc.sun.co.jp/solaris/topics/hotspot.html>

ホワイトペーパー

Java HotSpot™ Virtual Machine での動的トレースのサポート

はじめに

Java™ Platform, Standard Edition 6 (コード名: Mustang) では、Java HotSpot™ Virtual Machine 内での動的トレース (DTrace) のサポートが導入されています。Dynamic Tracing Framework は Solaris™ 10 オペレーティングシステムの一部で、オペレーティングシステムカーネルとユーザープロセスを動的に変更し、プロンプトと呼ばれる特定の対象がイベントでデータを記録することで、パフォーマンスメトリックを収集します。そしてプロンプトは、プロバイダと呼ばれる特別なカーネルモジュールを含めて使用可能になります。Mustang リリースに含まれているプロバイダとプロンプトにより、DTrace で Java プログラミング言語で書かれているアプリケーションのパフォーマンスデータを収集することができます。

Mustang リリースには、hotspot と hotspot_jni という 2 つの組み込み型の DTrace プロバイダが含まれています。これらのプロバイダにより実行されるすべてのプロンプトは、User-level Statically Defined Tracing (USD) プロンプトで、Java HotSpot Virtual Machine プロセスの PID によりアクセスされます。

hotspot プロバイダには、次の Java HotSpot Virtual Machine サブシステムに関連するプロンプトが含まれます。

- VM ライフサイクルプロンプト: VM の初期化および終了
- スレッドライフサイクルプロンプト: スレッド起動および停止イベント用
- クラスローディングプロンプト: クラスのロードおよびロード解除動作
- ガベージコレクションプロンプト: システム全体のガベージおよびメモリーブローコレクション用
- メソッドコンパイルプロンプト: どのメソッドがどのコンパイラによりコンパイルされているかを示す
- モニタープロンプト: すべての待機および通知イベントに加え、競合モニターエントリーおよび終了イベント用
- アプリケーションプロンプト: スレッド実行、メソッドエントリー/メソッドリターン、およびオブジェクト割り当ての詳細な調査

すべての hotspot プロンプトは VM ライブラリ (libjvm.so) から発生するため、VM を組み込むプログラムからも提供されます。

hotspot_jni プロバイダは Java™ Native Interface (JNI) に関連するプロンプトを含み、すべての JNI メソッドのエントリーおよびリターンポイントに位置しています。

また、DTrace jstack アクションは、Java メソッド名およびネイティブ関数名の両方を含む混合モードスタックトレースを出力します。

DTrace を使用したアプリケーションの調査と Mustang Java Hotspot Virtual Machine

この節では、Mustang Java HotSpot Virtual Machine と Solaris 10 Dynamic Tracing Framework の対話を示す。2 つのサンプルアプリケーションを示します。第 1 の例である JavaDemo は Mustang リリース

¹ この機能は、Java 2 Platform Standard Edition 5.0 Update Release 3 からサポートされています。

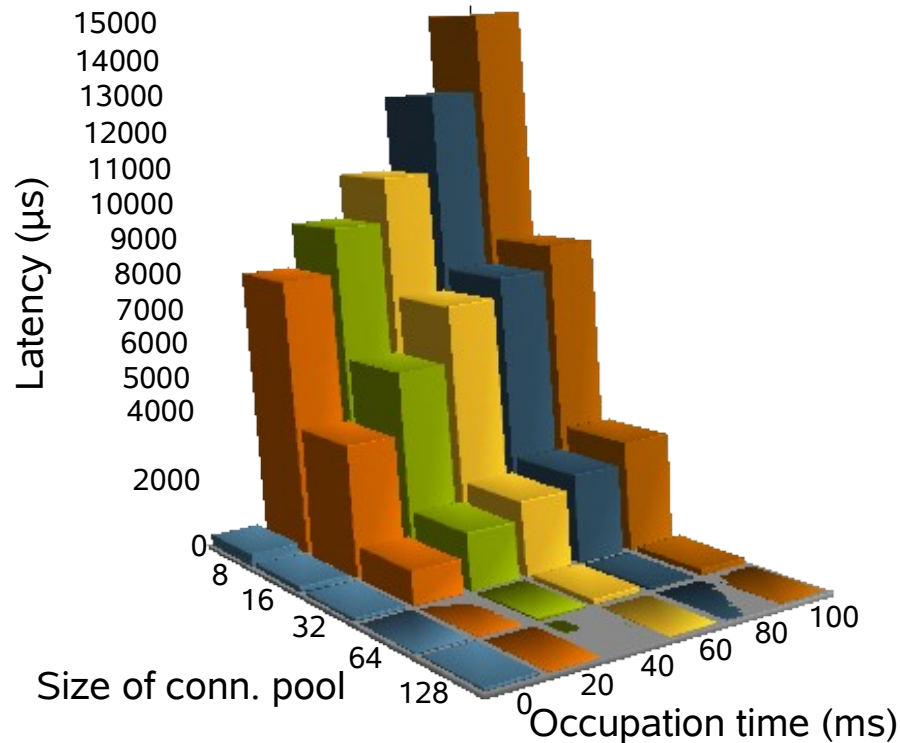
DTrace 使用例

- アプリケーションサーバの JDBC コネクションプールの大きさと待ち時間

```
#!/usr/sbin/dtrace -s
dvm$1:::monitor-wait {
    self->my[tid, copyinstr(arg0)] = timestamp;
}
dvm$1:::monitor-waited {
    @my[tid, copyinstr(arg0)]
        = sum(timestamp - self->my[tid, copyinstr(arg0)]);
}
pid$1:::exit:entry {
    printa(@my); exit(0);
}
```

DTrace で AppServer をプロファイル

- 自分の欲しい情報を望み通りに取得可能
- システムパフォーマンスに大きな影響を与えない
 - > DTrace 自身が過負荷なプロファイルを自動停止



ドリル5:まとめ

- 統計情報を上手に活用する
- ツールを活用する
- チューニング目標をクリアにする

ドリル 6:

“人間は時間的なプレッシャーを
いかにかけられても、
速くは考えられない”

リスターの法則



速く答えを出すのに重要な事

1. 脳トレで鍛える
2. 練習や準備をたくさんしておく
3. コンピュータにやらせる
4. 他人に任せる
5. 超能力で時間をゆっくり進ませる

考える時間

実作業の時間

何も工夫していないとき



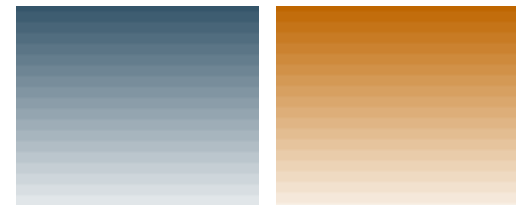
事前に準備をしておく



コンピュータにやらせる



他人に任せる
協力をあおぐ



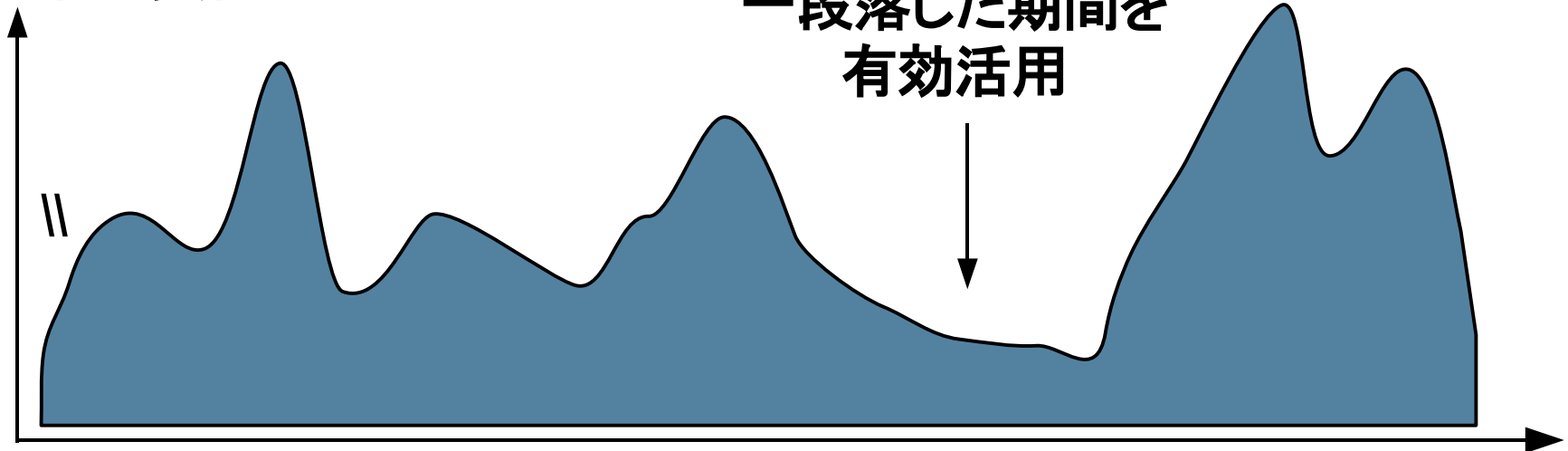
事前に準備をしておく

- ツールの使い方や機能制限を覚えておく
 - > 実際に使ってみる事が大事
- 情報を事前に手に入れておく
 - > 日々の情報収集
 - > 過去事例の調査
 - > 新技術動向の追跡

事前の準備

- 仕事が一段落したら情報を整理
 - > Wiki、メール、リンク集の整理
 - > 検索可能かつ、連想可能な状態に整理しておく事が大事

仕事の負荷



事前の準備：岡崎の場合

- ブログを読む
 - > RSS たくさん (30 サイト程度)
 - > 100~200 エントリ / 日
 - > なるべく自分もブログに残す
- メールを読む
 - > メーリングリストたくさん (100 程度)
 - > 200~300 通 / 日
 - > なるべく参加する

中身までしっかり読む
のは 2 割程度
最大 500 エントリ / 日
が限度

知らない技術が使われていたら

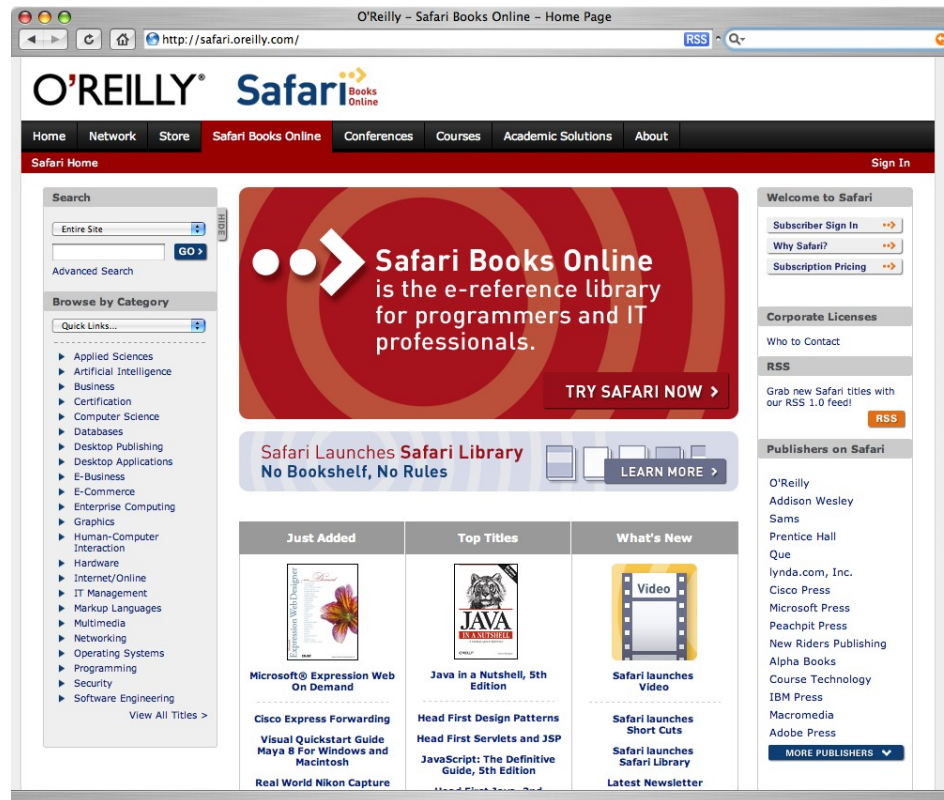
1. Google にきいてみる
2. サポート外と言って交渉
3. 知ったかぶり
4. 本屋にダッシュ
5. 知らない事なんて無い
6. 周りの人に聞きまくる

周りの人に聞きまくる

- トラブル時のように急ぎの場合はよく知っている人に聞くのがベスト
- 詳しい人が身の周りにいなくても、IT を活用する
 - > ブログ
 - > 掲示板やサポートフォーラム
 - > メーリングリスト

本屋にダッシュ

- 多少情報が古くても、情報がまとまっている事が多い
- オンラインで本が読める有料サービスも検討



“できません” シンドローム

- なぜか”できません”という回答だけは即答
- なぜできないのか？
 - > 何があったらできるのか、を考える
 - > 例：時間、予算、気力、制約が緩和される
- 忙しいからできない
 - > 優先度が低いからできない、と言い換え可能

ドリル6:まとめ

- トラブル時には事前の準備が大きな手助けになる
- IT をうまく使って労力を軽減する

参考資料

- ブログ
 - > <http://blogs.sun.com/okazaki>
- ツール関係
 - > <http://ja.netbeans.org>
 - > <http://pmd.sourceforge.net>
 - > <http://checkstyle.sourceforge.net>
 - > <http://findbugs.sourceforge.net>
 - > <http://www.opensolaris.org/os/project/opengrok>
- その他
 - > <http://sdc.sun.co.jp>