JDK9 の JMC & JFR のプレビュー

Java SE Advanced

日本オラクル株式会社
Java Global Business Unit
マスター・プリンシパル・セールスコンサルタント

宇野 浩司

Java Day Tokyo 2017
2017年5月17日

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

本日の内容

JFR と JMC のおさらい

JDK9 の Java Flight Recorder

Java Mission Control 6.0

JFR と JMC のおさらい

JDK9 の Java Flight Recorder

Java Mission Control 6.0

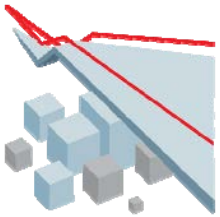
Java Day ORACLE

JMC と JFR

- **Java Mission Control と Java Flight Recorder**
- **ライセンス - Java SE Advanced (有償)に含まれている**
  - WebLogic Suite と WebLogic Enterprise にも含まれている。
- **機能自身は、ダウンロードしたバイナリにすでに含まれている (2013年9月 7u40から, 初期設定では機能的にOffになっている)**
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
- **開発時は無償**
  - http://www.oracle.com/technetwork/java/javase/terms/license/index.html

Java SE Advanced

サポート窓口



延長サポート

社内Java利用管理ツール

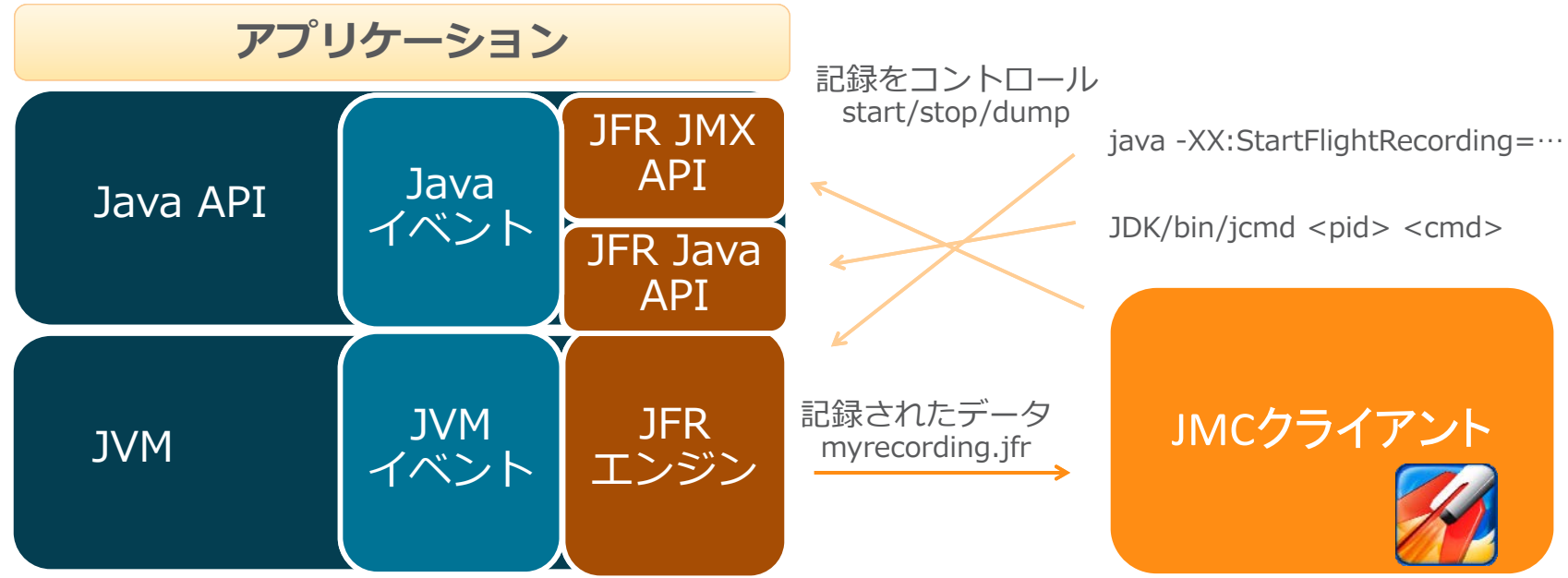
プロファイラー (JMC & JFR)

機能の概要

- **JFR (Java Flight Recorder)**
  - Oracle の Java SE 実行環境に内蔵されている**プロファイラ**
  - Java実行環境内にすでに収集されているデータを**効率よく記録する仕組み**

- **JMC (Java Mission Control)**
  - クライアントアプリケーション
  - JFRが記録したデータの視覚化
  - JMXコンソール (リアルタイム監視)
  - …

JFR と JMC の構成



イベントとして入手できる主な情報

- **Java アプリケーション**
  - 例外, スレッド, ファイル I/O, ネットワーク I/O など
- **Java VM**
  - メモリの割当て, クラスのロード, JIT, GC, メソッドのサンプリング など
- **オペレーティングシステム**
  - メモリやCPUの利用情報 など

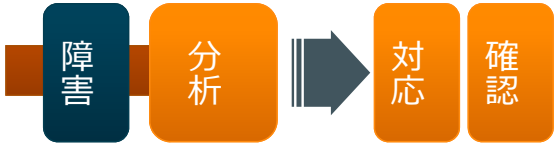
アプリケーションから JVM内部およびOSまで 幅広く情報を記録

プロファイラの継続利用による障害の対応（例）

プロファイラを利用しない場合の対応

障害発生時に確認された
ログを利用した一次対応

プロファイラを正常時から継続利用した場合の対応



障害発生時にさまざまな情報が
正常な状態から取得済み

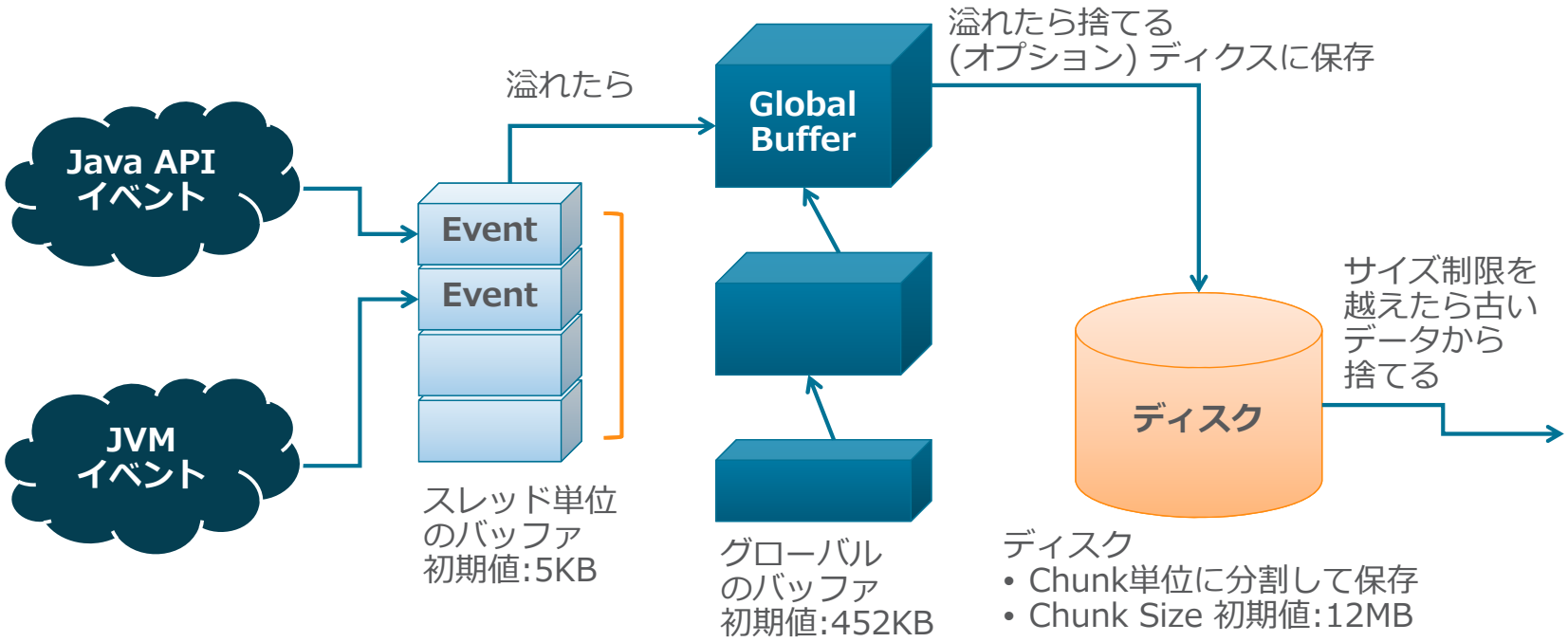
解決までの時間を短縮！

課題: 負荷が高い, 記録用のツールをリモートから常時接続

JFR によるプロファイリングの特徴

- **アプリケーションの変更は不要**
  - JavaVMの設定を有効にするのみ
- **低負荷 (初期設定は1～2%のオーバヘッド) - 運用時にも利用できる**
  - 診断情報生成や記録のためにアプリケーションへのコードの追加や変更はしない
    - HotSpot VMがすでに生成および収集しているデータを活用
  - すべての振る舞いを記録しない
    - 条件と一致した場合にイベントとして記録
    - イベント発生条件(しきい値や期間)の変更は可能
- **アプリケーション・ローカルに記録**
  - 記録時にリモートに他のツールを接続し続ける必要なし
  - 記録用のファイルの圧縮やサイズ制限可能
    - compress(gzip), maxage, maxsize

フライトレコーダ内部のイベント記録の流れ



JMC & JFR デモ

JDK8, JMC 5.5

JFR と JMC のおさらい

JDK9 の Java Flight Recorder

Java Mission Control 6.0

Java Day ORACLE

JDK9 の JFR

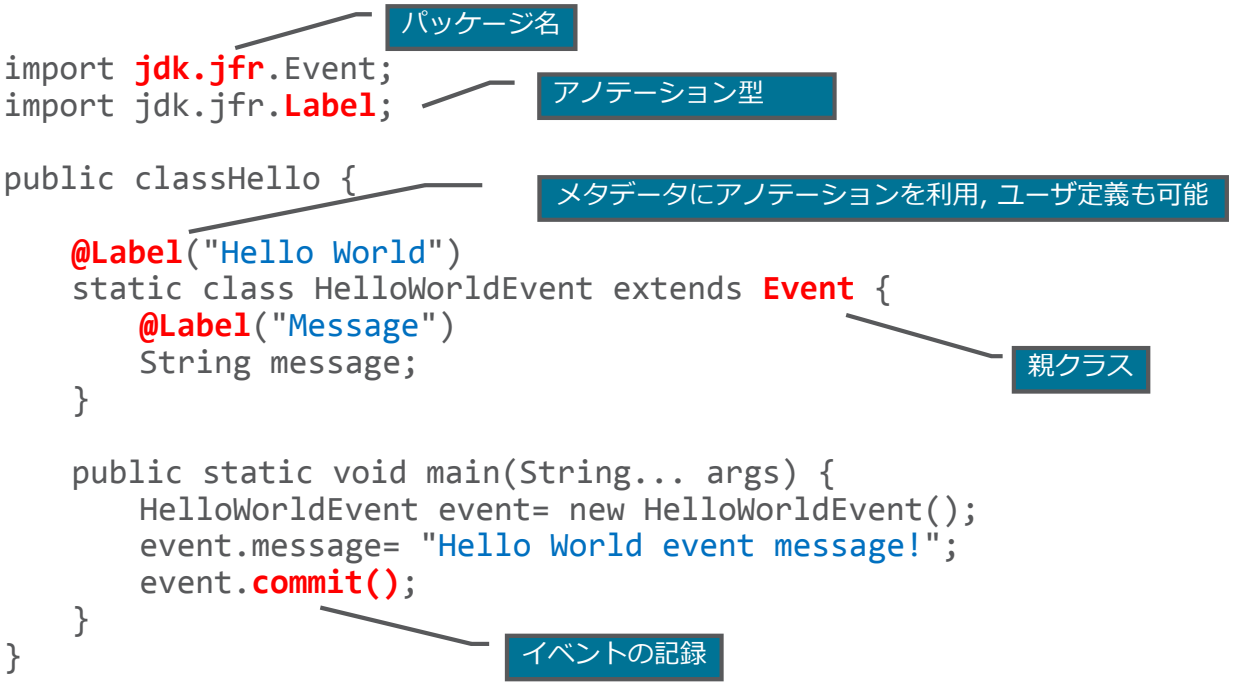
新機能の概要

- **パフォーマンスの継続的な改善**
- **異常終了時のディクスへのデータの出力**
  - クラッシュやOOM (Out Of Memory) 時など
- **API のサポート**
- **イベントの追加**
- **整理されたコマンドラインオプション**

サポートされるAPI

- **以前の試験的なAPIとは非互換**
  - oracle.jrockit.* から jdk.jfr.* に移動
- **モジュール jdk.jfr**
  - 小さなプロファイルで動作
    - JMXなしでもAPIからコントロール可能
  - 機能
    - カスタムイベントの追加
    - Flight Recorderのコントロール
    - 記録の解析
- **JMX - FlightRecorderMXBean**
    - com.oracle.jrockitからjdk.management.jfrへ
    - jdk.jfrとは別モジュール

カスタムイベント用のAPI




```
import jdk.jfr.Event;
import jdk.jfr.Label;

public classHello {

    @Label("Hello World")
    static class HelloWorldEvent extends Event {
        @Label("Message")
        String message;
    }

    public static void main(String... args) {
        HelloWorldEvent event= new HelloWorldEvent();
        event.message= "Hello World event message!";
        event.commit();
    }
}
```

フライト記録を構文解析 (例)

```
public class ParserExample {

    public static void main(String[] args) throws IOException{

        longmaxNanos= Long.MIN_VALUE;
        RecordedEvent maxEvent= null;

        for(RecordedEvent event : RecordingFile.readAllEvents(Paths.get(args[0]))) {
            if(event.getEventType().getName().equals("se.hirt.jfr4junit.TestEvent")) {
                long nanos= event.getDuration();
                if(nanos> maxNanos) {
                    maxNanos= nanos;
                    maxEvent= event;
                }
            }
        }

        System.out.printf("Longest running test was: %s for %ds¥n",
                maxEvent.getValue("displayName"), maxNanos/ 1_000_000);
        System.out.println("Event was:¥n"+ maxEvent);
    }
}
```

JFRのコントロールとフライト記録の構文解析 (例)

```
public class RecordAndConsume {

    public static void main(String[] args) throwsIOException{

        Path path= Paths.get(args[0]);

        try(Recording recording= new Recording()) {
            recording.setName("Fibonacci Recording");
            recording.start();
            recording.enable(FibonacciEvent.class);
            for(intn= 0; n< 50; n++) {
                FibonacciEvent event= new FibonacciEvent();
                event.number= n;
                event.begin();
                event.value= Fibonacci.fibonacciIterative(n);
                event.commit();
            }
            recording.stop();
            recording.dump(path);

            for(RecordedEvent event: RecordingFile.readAllEvents(path)) {
                intnumber= event.getValue("number");
                longvalue= event.getValue("value");
                System.out.printf("fibonacci(%d) = %d (time: %dns)¥n", number, value,
                  event.getDuration());
            }
        }
    }
}
```

- 記録の開始、
- イベントの記録
- ファイルへのダンプ

新しく追加されるイベント

- より詳細はセーフポイントの情報
- より詳細なコードキャッシュの情報
- PLAB (promotion local allocation buffers) イベント
- 詳細なインライン情報のためのコンパイラ・イベント
- よりよい視覚化のためのG1特有情報のイベント
- モジュール・イベント
- ネイティブライブラリ・イベント

など

コマンドラインから利用する場合の変更点

ハードディスクに保存しない場合 (In-memory example)

- **JDK 8**
  - -XX:+UnlockCommercialFeatures
    -XX:+FlightRecorder
    -XX:FlightRecorderOptions=defaultrecording=true,
    dumponexit=true, dumponexitpath=/home/hirt/myrecording.jfr
- **JDK 9**
  - -XX:+UnlockCommercialFeatures
    -XX:StartFlightRecording=dumponexit=true,
    filename=/home/hirt/myrecording.jfr

Java Day ORACLE

コマンドラインから利用する場合の変更点

ハードディスクに保存する場合 (Disk Repository Example)

- **JDK 8**
  - -XX:+UnlockCommercialFeatures
    -XX:+FlightRecorder
    -XX:FlightRecorderOptions=defaultrecording=true,
    dumponexit=true,
    dumponexitpath=/home/hirt/myrecording.jfr,disk=true,
    maxsize=500M,maxage=20m
- **JDK 9**
  - -XX:+UnlockCommercialFeatures
    -XX:StartFlightRecording=dumponexit=true,
    filename=/home/hirt/myrecording.jfr,
    maxsize=500M,maxage=20m

JCMD

JCMD を使用する場合は今までと同じ

例)

> jcmd 4711 VM.unlock_commercial_features

> jcmd 4711 JFR.start duration=2m,filename=/home/hirt/myrecording.jfr 29

JFR と JMC のおさらい

JDK9 の Java Flight Recorder

Java Mission Control 6.0

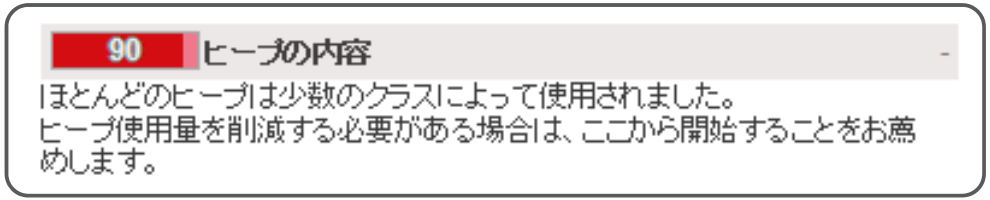
Java Mission Control 6

新機能の概要

- **メジャー・バージョンアップ**
  - JDK9にのみ提供
  - JDK 7/8 および JDK 9 JFR ファイルフォーマットのサポート
  - JDK 7/8 および JDK 9 への接続をサポート
- **フライト記録の自動分析**
- **新しいユーザインタフェース**

フライト記録の自動分析

- **定義されたルールに基づきデータを分析**
  - Java APIを利用した拡張もサポート
- **結果の一覧表示**
  - 潜在的な問題点の指摘
- **個々のページへのリンク**



フライト記録の自動分析 -コンポーネント

- **JMC以外でも利用可能なPOJOコンポーネント**
  - パーサ
  - ルール・エンジン

- **ルール・エンジンはJMC以外でも利用**
  - Oracle内部のPoCクラウドサービス
    - JFRのアップロード、JSON形式の戻り値
  - Oracle内部の分析Webアプリケーション (API テスト, rules テストなど)
  - Oracle Enterprise Managerの記録の自動分析で使用

フライト記録の自動分析 - Java APIを利用した拡張

- **API が提供する機能**
  - フィルタ
  - アグリゲータ

- **簡単にルールの開発が可能**
  - サポートの対象ではないがJMC6のドキュメントには明記
  - 標準的なJavaサービス・ローダ機構を使用
    - ボイラープレートを生成するPDEプラグイン(試験的)
  - 多くの支持があれば将来正式にサポート対象に

ルール用インタフェース - IRule

```
public interface IRule {

    RunnableFuture<Result> evaluate(IItemCollection items,
        IPreferenceValueProvider valueProvider);

    Collection<TypedPreference<?>> getConfigurationAttributes();

    getId();

    getName();

    getTopic();

}
```

ルール実装の例

```
public class EnvironmentVariableRule implements IRule {

     private static final TypedPreference<String> PREFERENCE_ENVIRONMENT_VARIABLE_NAME = new TypedPreference<>(
          "environmentVariable", "Environment Variable",
          "The name of the environment variable to check for a floating point score",
          UnitLookup.PLAIN_TEXT.getPersister(), "JFR_RULE_TEST");


     @Override
     public Collection<TypedPreference<?>> getConfigurationAttributes() {
          return Arrays.<TypedPreference<?>> asList(PREFERENCE_ENVIRONMENT_VARIABLE_NAME);
     }


     @Override
     public String getId() {
          return "se.hirt.envrule.EnvironmentVariableRule";
     }


     @Override
     public String getName() {
          return "Configurable Rule Demo";
     }

     @Override
     public String getTopic() {
          return JfrRuleTopics.ENVIRONMENT_VARIABLES_TOPIC;
     }

     .
     .
     .
```

Rule の例 (続き)

```
        .
        .
        .
        @Override
        public Result evaluate(IItemCollection items, IPreferenceValueProvider valueProvider) {

                String variableName = valueProvider.getPreferenceValue(PREFERENCE_ENVIRONMENT_VARIABLE_NAME);
                String environmentVariableValue = getEnvironmentVariable(variableName, items);

                if(environmentVariableValue == null) {
                        return new Result(this, 100, "Could not find the environment variable named "+ variableName);
                }

                doublescore = 0;

                try{
                        score = Double.parseDouble(environmentVariableValue);
                } catch(NumberFormatException e) {
                        return new Result(this, 100, "Could not parse the environment variable named "+ variableName);
                }

                return new Result(this, score, "The score in " + variableName+ " was "+ score);
        }

        private String getEnvironmentVariable(String variableName, IItemCollection items) {
                IItemCollection envItems= items.apply(ItemFilters.and(JfrFilters.ENVIRONMENT_VARIABLE,
                        ItemFilters.equals(JfrAttributes.ENVIRONMENT_KEY, variableName)));

                return envItems.getAggregate(JfrAggregators.first(JfrAttributes.ENVIRONMENT_VALUE));
        }
}
```

Rules の実行

```
public class RateFile {

    public static void main(String[] args) throwsIOException, CouldNotLoadRecordingException {

        IItemCollection events = JfrLoaderToolkit.loadEvents(new File(args[0]));

        for(IRule rule: RuleRegistry.getRules()) {

            Result result = rule.evaluate(events, IPreferenceValueProvider.DEFAULT_VALUES);

            if(result.getScore() > 50) {
                System.out.printf("(%.0f) [%s]: %s¥n", result.getScore(),
                  result.getRule().getId(),result.getShortDescription());
            }
        }

    }
}
```

新しいユーザインタフェース

- **ナビゲーション**
  - アウトライン・ビューのサポート
    - タブの中のタブはなくなった
  - レンジナビゲータの統合

- **先進的なフィルタ**
  - boolean 操作(or, and)でフィルタの構築が可能
  - 記録自体とは独立

新しいユーザインタフェース (続き)

- **分離されたスタックトレースの表示**
  - すべてのスレッドに対して共通
  - 構成を変更して旧ツリー方式の表示にも対応

- **Eclipse内で利用する場合は、JDK8以降のJVMが必要**
  - UI は、ストリーム、他のJDK8の言語仕様やJDK8で追加されたAPIを使用

新しいユーザインタフェース (続き)

- **GUI ビルダの廃止**
  - もし要望が多ければ復活させるかもしれない
- **ページのカスタム構成**
  - カスタムイベントのページの追加
  - 構成可能な既存ページ

JMC UI Demo

JMC 6 / JDK 9

まとめ

- **JDK 9 で JFR API をサポート**
  - レコーダのコントロール
  - カスタムイベントの生成
  - 記録の解析

- **JMC 6 の 新しい JFR 対応部分**
  - 記録の自動分析
    - カスタムルールを追加できるプラグイン
  - 新しいUI
  - JDK 7, 8 および 9 の記録フォーマットへの対応

参考情報

- **本会場の展示ブース**

- **Java Platform, Standard Edition トラブルシューティング・ガイド, リリース8 (2015年3月)**
  - https://docs.oracle.com/javase/jp/8/docs/technotes/guides/troubleshoot/

Q&A

Integrated Cloud

Applications & Platform Services

Java Day™

ORACLE


ORACLE