【MicroProfile開発ガイド】
MicroProfile OpenTracing

2018/06

日本アイ・ビー・エム システムズ・エンジニアリング株式会社

Disclaimer

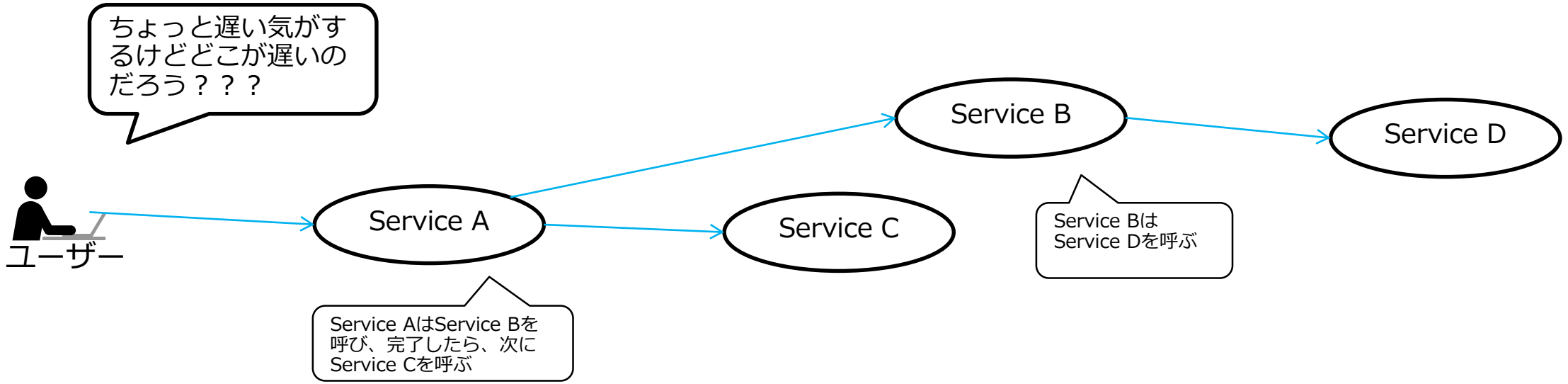
- この資料は日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム　システムズ・エンジニアリング株式会社の正式なレビューを受けておりません。
- 当資料は、資料内で説明されている製品の仕様を保証するものではありません。
- 資料の内容には正確を期するよう注意しておりますが、この資料の内容は2018年6月現在の情報であり、製品の新しいリリース、PTFなどによって動作、仕様が変わる可能性があるのでご注意下さい。
- 今後国内で提供されるリリース情報は、対応する発表レターなどでご確認ください。
- IBM、IBMロゴおよびibm.comは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点でのIBMの商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。
- 当資料をコピー等で複製することは、日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム　システムズ・エンジニアリング株式会社の承諾なしではできません。
- 当資料に記載された製品名または会社名はそれぞれの各社の商標または登録商標です。
- JavaおよびすべてのJava関連の商標およびロゴはOracleやその関連会社の米国およびその他の国における商標または登録商標です。
- Microsoft, WindowsおよびWindowsロゴは、Microsoft Corporationの米国およびその他の国における商標です。
- Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。
- UNIXはThe Open Groupの米国およびその他の国における登録商標です。

目次

MicroProfile OpenTracing概要

分散トレースとは（1/3）

マイクロサービス・アーキテクチャーを採用したアプリケーションでは、1つの要求を処理するために複数のサービスが連携して稼働します。このとき、複雑に連携している各サービスが個別に出力するトレースでは、障害発生時の原因究明やパフォーマンス分析が困難です。



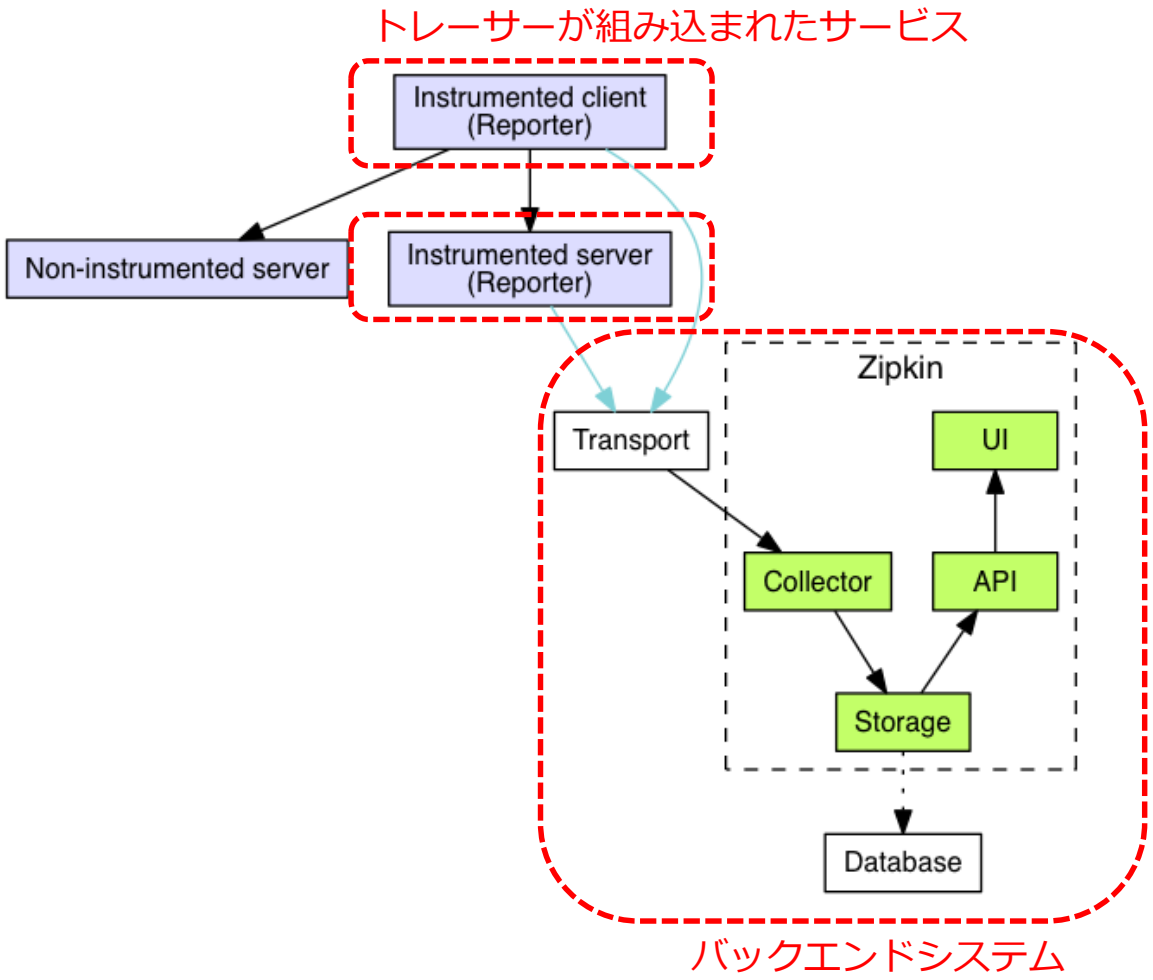
このような課題を解決するため、マイクロサービス間の連携状況を簡単にトレースできるようにするための仕組みが「分散トレース」です。

分散トレースとは（2/3）

右の図は代表的な分散トレース・システムであるZipkinのアーキテクチャーです。
分散トレース・システムは以下の2つから成り立ちます。

- トレース・データを送信するためのトレーサーが組み込まれたアプリケーション（サービス）
- トレース・データを集約し、ユーザーへ分析のインターフェースを提供するバックエンド・システム

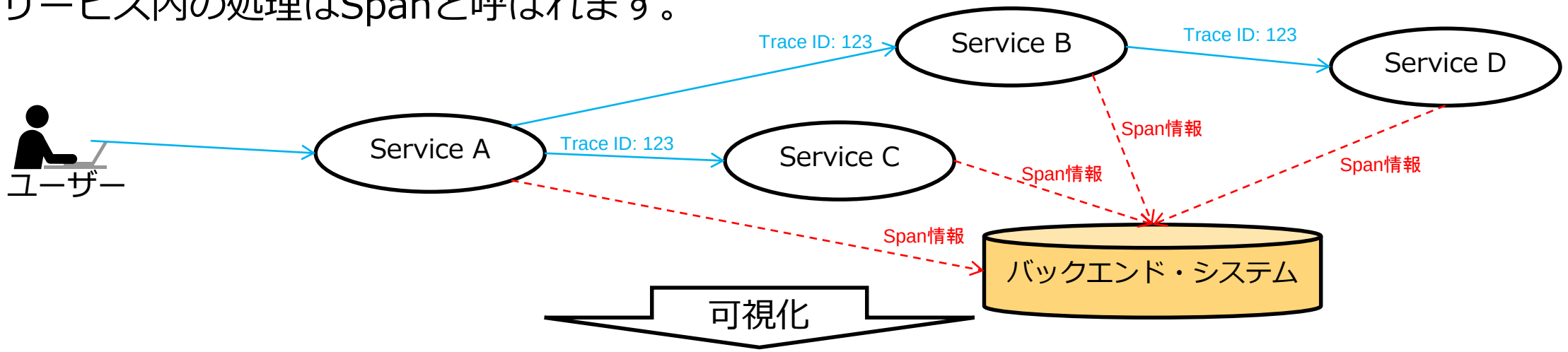
トレーサーは上流のサービスで要求に相関ID（トレースID）を割り当て、これを下位のサービスに伝搬します。そして、トレーサーは各サービスから相関ID付きのトレース・レコードをバックエンド・システムに記録します。

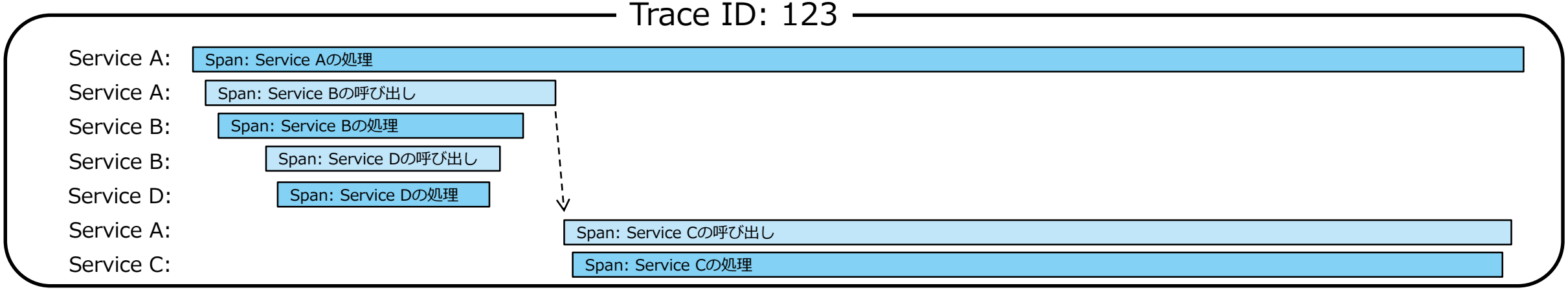


分散トレースとは（3/3）

ある要求の開始から終了までの処理の全体はTraceと呼ばれます。このTraceには、分散トレーシング・システムで一意となるIDが割り当てられます。

あるサービス内の処理はSpanと呼ばれます。





OpenTracingとは

アプリケーションに分散トレースのための計測機能を組み込む（instrument、計装する）ための、様々な言語の、ベンダーに中立なAPI仕様をまとめ、その仕様を実装したライブラリを提供しているのがOpenTracingプロジェクトです。

OpenTracing APIを使うことで、アプリケーション・コードと分散トレースの実装を分離することができます。すなわち、OpenTracing Tracerの実装を切り替えるというわずかな変更で、異なるバックエンド・システムにトレース・データを送信することができます。

なお、OpenTracingが標準化の対象としているのはAPIであり、以下は対象ではありません。

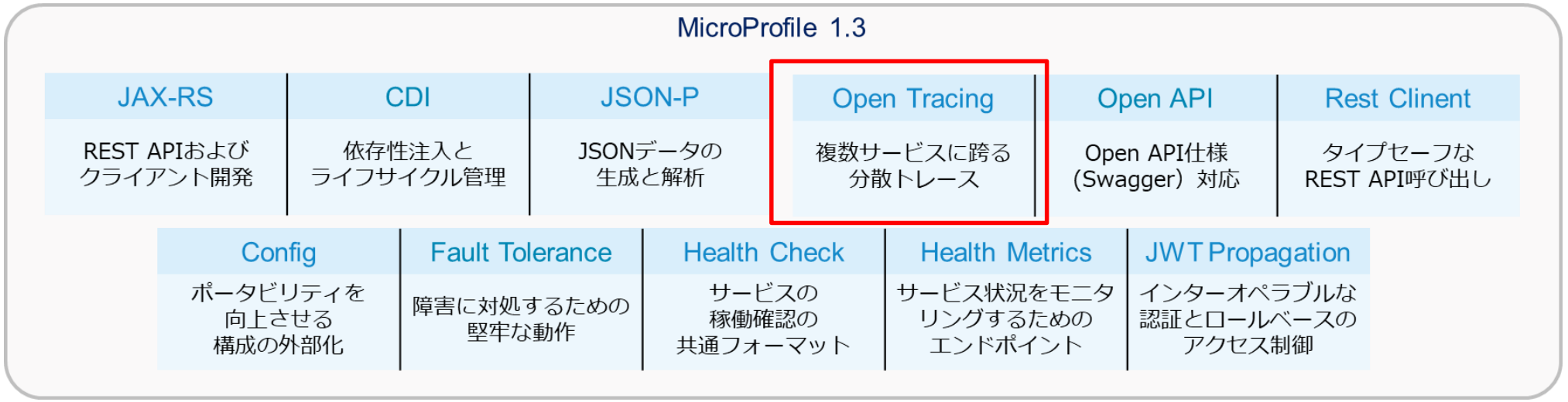
- サービス間で相関IDを伝搬するためのプロトコルとデータのフォーマット
- トレーサーが計測した相関ID付きのトレース・レコードを、バックエンド・システムに送信するプロトコルとデータのフォーマット

これらの標準化はOpenCensusという別のプロジェクトで検討されています。

MicroProfile OpenTracingとは

MicroProfile OpenTracing（MP OpenTracing）プロジェクトはJAX-RSアプリケーションをトレースするための仕様とAPIを策定しています。このプロジェクトはOpenTracingプロジェクトのいわばラッパーであり、MP OpenTracing 1.0仕様ではOpenTracing Java実装のバージョン0.30.0を使用しています。

LibertyのMP OpenTracingフィーチャーは、JAX-RSアプリケーションを対象として、アプリケーションに分散トレースのためのコードを追加することなく、OpenTracing準拠のトレーサーを組み込み、バックエンド・システムにトレース・データを送信することを可能にします。



フィーチャーの利用方法

ユーザー・フィーチャーの導入

MP OpenTracingフィーチャーを利用するには、 MP OpenTracingフィーチャーに加えて、バックエンド・システムに対応したio.opentracing.Tracerの実装を提供するユーザー・フィーチャーが必要です。

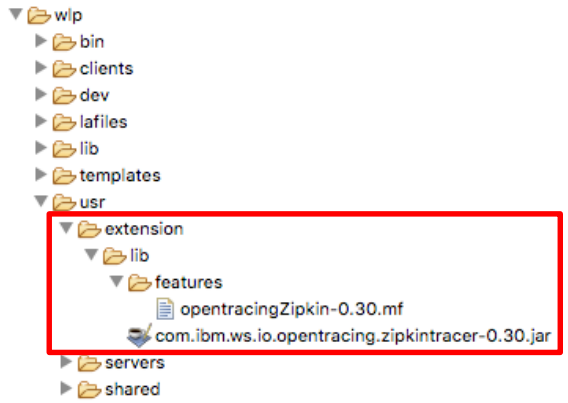
Zipkinに対応したユーザー・フィーチャーのサンプル（opentracingZipkin）が提供されており、このフィーチャーを利用できます。

1. ユーザー・フィーチャーのzipファイルをダウンロードします。
   - ユーザーフィーチャーのソースコード（GitHub）
     https://github.com/WASdev/sample.opentracing.zipkintracer
   - ビルドされたユーザー・フィーチャーのzipファイル（Maven Central）
     http://central.maven.org/maven2/net/wasdev/wlp/tracer/liberty-opentracing-zipkintracer/1.0/liberty-opentracing-zipkintracer-1.0-sample.zip

```
wget https://repo1.maven.org/maven2/net/wasdev/wlp/tracer/liberty-opentracing-zipkintracer/1.0/liberty-opentracing-zipkintracer-1.0-sample.zip -O /tmp/tracer.zip
```

2. Libertyのusrディレクトリーにzipファイルを展開します。

```
unzip /tmp/tracer.zip -d /opt/ibm/wlp/usr
```



（参考）Mavenでのユーザー・フィーチャーのダウンロード

ユーザー・フィーチャーのダウンロードと展開をMavenビルド時に実施する場合は、download-maven-pluginを利用できます。以下にpom.xmlの記述例を示します。

```
            <plugin>
                <groupId>com.googlecode.maven-download-plugin</groupId>
                <artifactId>download-maven-plugin</artifactId>
                <version>1.4.0</version>
                <executions>
                    <execution>
                        <id>install-tracer</id>
                        <phase>prepare-package</phase>
                        <goals>
                            <goal>wget</goal>
                        </goals>
                        <configuration>
                            <url>https://repo1.maven.org/maven2/net/wasdev/wlp/tracer/liberty-opentracing-
zipkintracer/1.0/liberty-opentracing-zipkintracer-1.0-sample.zip</url>
                            <unpack>true</unpack>
                            <outputDirectory>${project.build.directory}/liberty/wlp/usr</outputDirectory>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
```

（参考）Dockerでのユーザー・フィーチャーのダウンロード

LibertyをDockerコンテナーで稼働させる場合、Dockerfileに下記を追加してユーザー・フィーチャーのダウンロードと展開を行います。

```
RUN wget https://repo1.maven.org/maven2/net/wasdev/wlp/tracer/liberty-opentracing-zipkintracer/1.0/liberty-opentracing-zipkintracer-1.0-sample.zip -O /tmp/tracer.zip && ¥
  unzip /tmp/tracer.zip -d /opt/ibm/wlp/usr && ¥
  rm /tmp/tracer.zip
```

フィーチャーの有効化（1/2）

server.xmlでmpOpenTracing-1.0フィーチャーと、 opentracingZipkinユーザー・フィーチャーを有効化します。
mpOpenTracing-1.0フィーチャーが含まれるMicroProfile-1.3を設定することも可能です。

```
<featureManager>
    <feature>mpOpenTracing-1.0</feature>
    <feature>usr:opentracingZipkin-0.30</feature>
</featureManager>

<opentracingZipkin host="localhost" port="9411" />
```

mpOpenTracing-1.0フィーチャーを有効化することにより、以下のフィーチャーも暗黙的ロードにより自動的に有効化されます。

- servlet-3.1
- cdi-1.2
- json-1.0
- jaxrsClient-2.0
- jaxrs-2.0
- opentracing-1.0
- jndi-1.0

※opentracing-1.0はフィーチャーはMicroProfile OpenTracing仕様が確定する前にリリースされており、Liberty v17.0.0.4から使用可能です。mpOpenTracing-1.0フィーチャーは確定したMicroProfile OpenTracing仕様に準拠しており、v18.0.0.1から使用可能です。後述する@Tracedアノテーションなどによる高度なトレースの取得にはmpOpenTracing-1.0フィーチャーが必要です。

フィーチャーの有効化（1/2）

opentracingZipkinユーザー・フィーチャーに指定可能な主なオプションとそのデフォルト値は以下の通りです。

オプション	デフォルト値	意味
host	zipkin	Zipkinサーバーのホスト名またはIPアドレス
port	9411	Zipkinサーバーのポート番号

※その他のオプションについては、以下のリンク先のmetatype.propertiesを確認して下さい。
https://github.com/WASdev/sample.opentracing.zipkintracer/blob/master/src/main/resources/OSGI-INF/i10n/metatype.properties

フィーチャーの動作

フィーチャーの動作の概要

基本的な利用方法では、アプリケーション・コードの変更は不要です。

MP OpenTracingフィーチャーは以下のことを行います。

- JAX-RSのインバウンド・リクエストから自動的にSpanContextを抽出する
- JAX-RSのインバウンド・リクエストについて自動的にSpanを開始し、リクエストが終了したらSpanを終了する
- JAX-RSのアウトバウンド・リクエストについて自動的にSpanContextを注入する
- JAX-RSのアウトバウンド・リクエストについて自動的にSpanを開始し、リクエストが終了したらSpanを終了する

これらの機能はJAX-RSのエンドポイントにFilterとして自動的に組み込まれます。

※基本的なJAX-RSのエンドポイントのトレース以外を行いたい場合は、@Tracedアノテーションを使い、トレースすべきメソッドを指定したり、OpenTracing Tracerインスタンスをインジェクションしてインスタンスに直接アクセスすることも可能です。次節で紹介します。

インバウンド・リクエストの処理

JAX-RSのエンドポイントにリクエストが到着すると、フィーチャーによってリクエストからSpanContextが抽出され、このSpanContextへ子としての参照を持つ、新しいSpanが作成されます。

この新しいSpanのオペレーション名は以下となります。

```
<HTTP method>:<package name>.<class name>.<method name>
```

このSpanには以下のタグが付与されます。タグはZipkinコンソールでの検索などに利用できます。

タグ	値の例	備考
Tags.SPAN_KIND	server	OpenTracing APIのTagsクラスで定数SPAN_KIND_SERVERとして定義
Tags.HTTP_METHOD	GET	
Tags.HTTP_URL	http://localhost:9081/factorial/2	
Tags.HTTP_STATUS	200	
Tags.ERROR	true	サーバーエラー（5XX）の場合に付与される

アウトバウンド・リクエストの処理

JAX-RS（javax.ws.rs.client.Client）によってリクエストが送られる場合、アクティブなSpanに子としての参照を持つ、新しいSpanが作成され、アウトバウンド・リクエストにはそのSpanContextが注入されます。アウトバウンド・リクエストの終了でSpanも終了します。

この新しいSpanのオペレーション名は以下になります。

```
<HTTP method>
```

このSpanには以下のタグが付与されます。タグはZipkinコンソールでの検索などに利用できます。

タグ	値の例	備考
Tags.SPAN_KIND	client	OpenTracing APIのTagsクラスで定数SPAN_KIND_CLIENTとして定義
Tags.HTTP_METHOD	GET	
Tags.HTTP_URL	http://localhost:9080/calc/2/3	
Tags.HTTP_STATUS	200	
Tags.ERROR	true	クライアントエラー（4XX, 5XX）の場合に付与される

Zipkin

Zipkinの起動

ZipkinサーバーはMicroservice Builder FabricをIBM Cloud Private環境にデプロイすることで導入できるほか、複数の方法で導入ができます。Zipkinのサイトを確認して下さい。

- https://zipkin.io/pages/quickstart.html

稼働確認のためには、jarファイルをダウンロードし、ローカルで実行する方法が簡単です。

```
curl -sSL https://zipkin.io/quickstart.sh | bash -s
java -jar zipkin.jar
```

ZipkinのJavaプロセスを起動後、以下のURLでZipkinコンソールにアクセスできます。

- http://localhost:9411/zipkin/

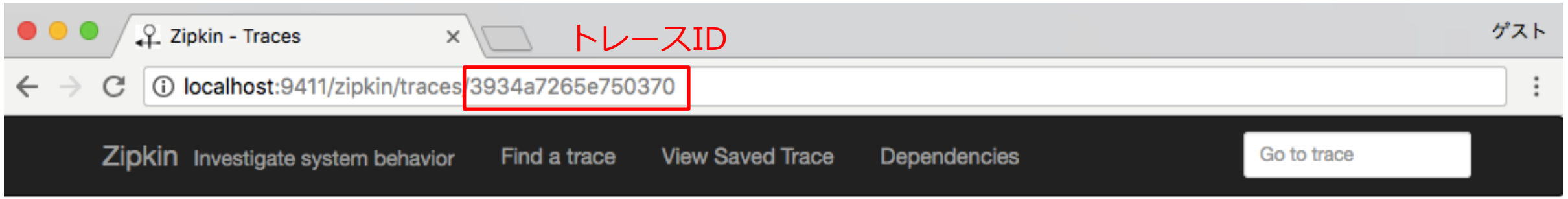
（参考）

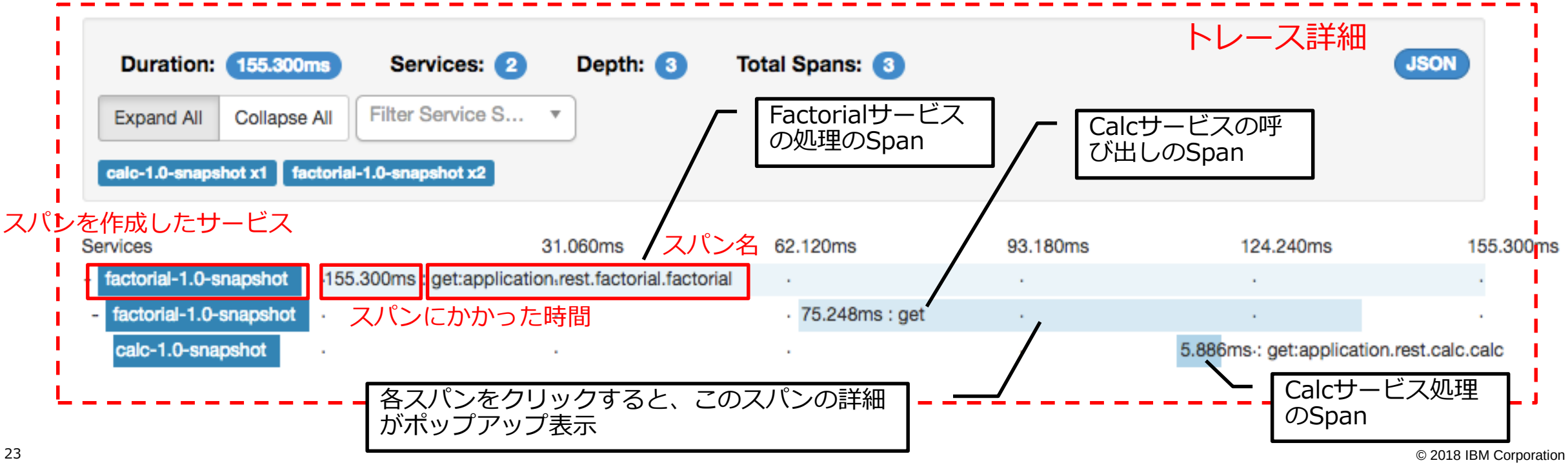
IBM Cloud Private への Microservice Builder ファブリックのインストール

https://www.ibm.com/support/knowledgecenter/ja/SS5PWC/installing_fabric_task.html

Zipkinコンソール（トップ画面）

Zipkinコンソール（トレース詳細）





高度な利用方法

@Tracedアノテーション（1/2）

デフォルトでは、JAX-RSのエンドポイントにおいて自動的にSpanが作成されますが、任意のメソッドの開始／終了でSpanを開始／終了したい場合、@Tracedアノテーションを使用します。

- @Tracedアノテーションをクラスに付与した場合、アノテーションはそのクラスの全てのメソッドに適用されます。
- @Tracedアノテーションをクラスとメソッドの両方に付与した場合は、メソッドのアノーテションが優先されます。

```
. . .
import org.eclipse.microprofile.opentracing.Traced;
. . .
@ApplicationScoped
public class CalcClient {
. . .
    @Traced // このメソッドでトレースを有効にする
    public long multiply(long p1, long p2) {
. . .
    }
```

@Tracedアノテーション（2/2）

@Tracedアノテーションでは2つの引数が使用可能です。

引数	値	デフォルト値	説明
value	trueまたはfalse	true	@Tracedアノテーションがクラスレベルで指定された場合に、特定のメソッドでSpanの作成を無効にしたい場合@Traced(false)を指定します。 特定の JAX-RSエンドポイントに対して、@Traced(false)を使用してスパン作成を無効にすることもできます。この場合上流の SpanContextも抽出されません。
operationName	Spanのオペレーション名	""	Spanのオペレーション名を指定します。指定されなかった場合（デフォルトの""である場合）はデフォルトのオペレーション名が使用されます。 メソッドがJAX-RSのエンドポイントではない場合、デフォルトのオペレーション名は<package name>.<class name>.<method name>となります。 クラスに対してopetationNameが指定された場合は、メソッドのアノテーションで上書きしない限り、全てのメソッドでこのSpan名が使用されます。

Tracerのインジェクション（1/2）

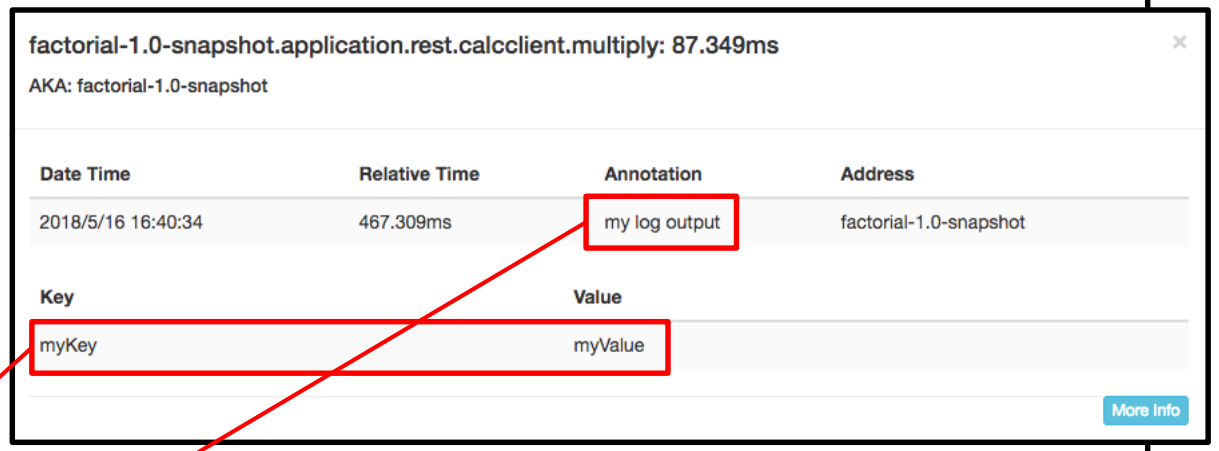
OpenTracing Tracerの構成済みインスタンスをインジェクションして、直接操作することも可能です。これによりビジネス・メソッド内でSpanを作成したり、Spanに情報を付加するなど、より複雑なトレース要件にも対応できます。

```
. . .
import io.opentracing.Tracer;
. . .
@ApplicationScoped
public class CalcClient {
. . .
    @Inject // Tracerインスタンスをインジェクション
    Tracer configuredTracer;
. . .
    @Traced
    public long multiply(long p1, long p2) {
. . .

    configuredTracer.activeSpan().setTag("myKey", "myValue"); // アクティブなSpanにタグをセット
    configuredTracer.activeSpan().log("my log output"); // アクティブなSpanにログをセット

. . .
    }
```

ZipkinコンソールのSpanの詳細から
セットしたタグやログは確認可能



Tracerのインジェクション（2/2）

Tracerインスタンスのインジェクションを行う場合、Tracerの実装クラスを提供するユーザー・フィーチャーのクラスにアクセスできるようにLibertyの設定を変更する必要があります。server.xmlでclassloaderのapiTypeVisibilityにthird-partyを追加します。

```
<webApplication name="factorial" location="${app.location}">
    <classloader apiTypeVisibility="api,ibm-api,spec,stable,third-party" />
</webApplication>
```

参考リンク

- 連載「Microservice Builder」第 4 回 Microservice Builder ファブリックの分散トレースを利用する
  https://www.ibm.com/developerworks/jp/websphere/library/icp/msb_introduction/4.html
- 連載「Microservice Builder」第 5 回 Microservice Builder ファブリックの分散トレースをもっと簡単に利用する
  https://www.ibm.com/developerworks/jp/websphere/library/icp/msb_introduction/5.html
- 分散トレースの有効化 - WebSphere Liberty Knowledge Center
  https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/twlp_dist_tracing.html
- Liberty 用の OpenTracing の構成 - Microservice Builder Knowledge Center
  https://www.ibm.com/support/knowledgecenter/ja/SS5PWC/configuring_opentracing_liberty.html
- IBM Cloud Private への Microservice Builder ファブリックのインストール - Microservice Builder Knowledge Center
  https://www.ibm.com/support/knowledgecenter/ja/SS5PWC/installing_fabric_task.html
- Open Tracing API 0.30.0 JavaDoc
  http://javadoc.io/doc/io.opentracing/opentracing-api/0.30.0
- MicroProfile OpenTracing JavaDoc
  https://openliberty.io/javadocs/microprofile-1.3-javadoc/org/eclipse/microprofile/opentracing/package-summary.html

End of File