# 【MicroProfile開発ガイド】
# MicroProfile Config

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 Config概要
  - MicroProfile Configとは
  - 利用シナリオ
  - フィーチャーの有効化

- MicroProfile Configの使用方法
  - 構成ソース（ConfigSources）とordinal
  - カスタム構成ソース（Custom ConfigSources）
  - 値の取得方法
  - 静的注入と動的注入
  - 標準装備コンバーターとpriority
  - カスタム・コンバーター

- 参考リンク

# MicroProfile Config概要

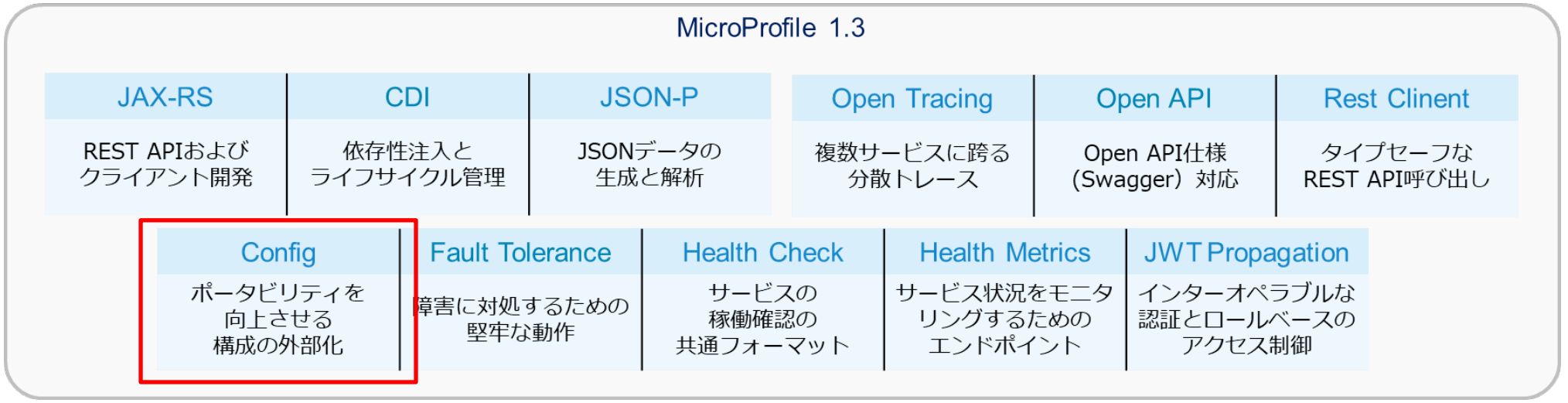
# MicroProfile Configとは

多くのアプリケーションは、稼動環境に応じて構成の変更を行う必要があります。

MicroProfile Configを利用することで、プロパティ・ファイル、環境変数、Javaのシステム・プロパティなど、アプリケーションの外部から構成情報を取得することができます。

特にクラウド環境では、構成情報が動的に変更されるケースがありますが、MicroProfile Configはこれをアプリケーションの再起動なしに取得することが可能です。

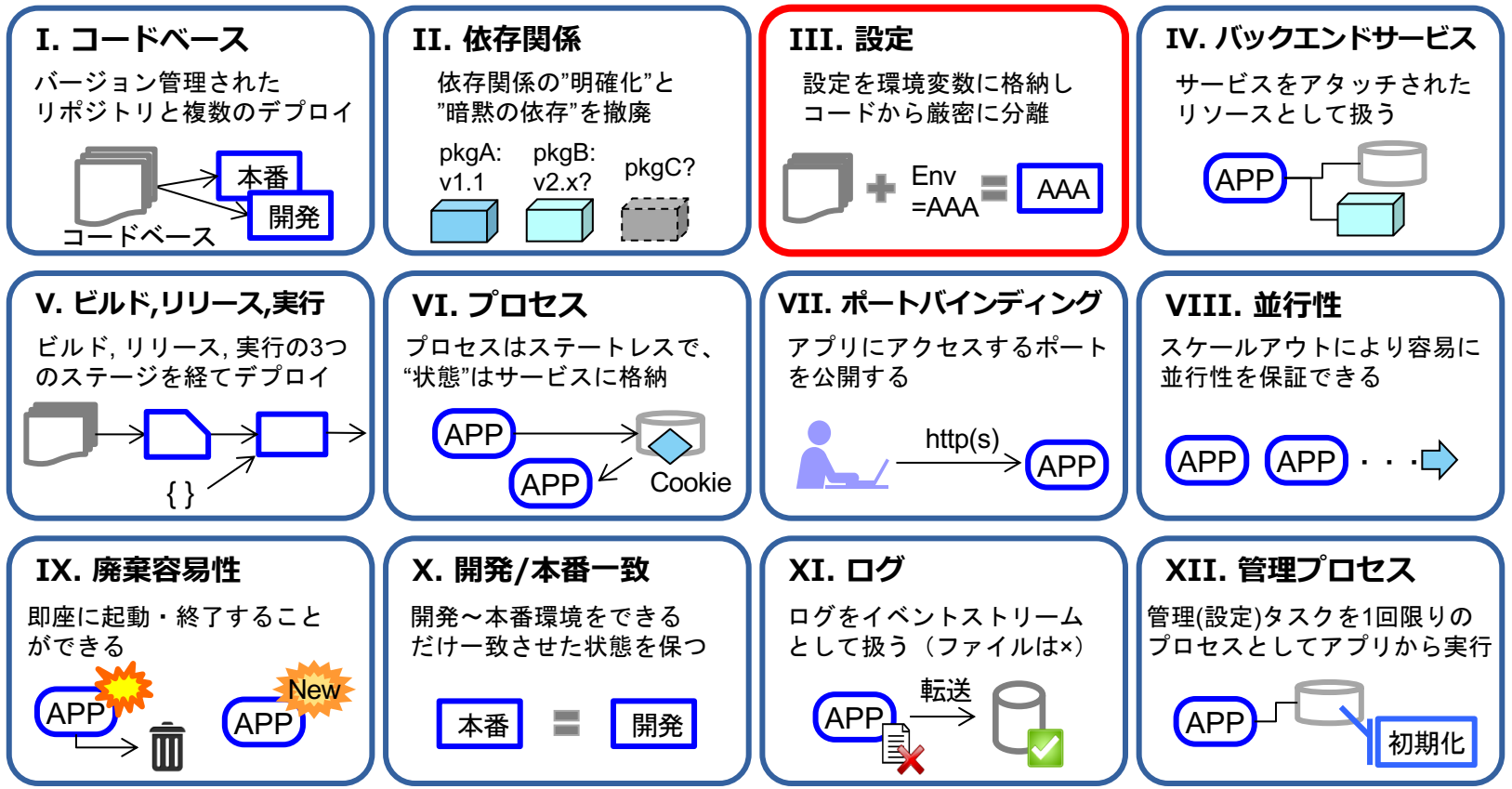
WAS Libertyでは、MicroProfile Config 1.2.1仕様に準拠した、mpConfig-1.2フィーチャーが提供されています。



# 利用シナリオ

クラウド・ネイティブ・アプリケーション開発のベストプラクティスであるThe Twelve -Factor App （およびBeyond the Twelve-Factor App）では、構成情報をコードから分離することが望ましいとされています。

MicroProfile ConfigはJsonファイルやDBなど異なるソースから同じ方法で構成情報を取得できるため、取得方法の標準化ができます。

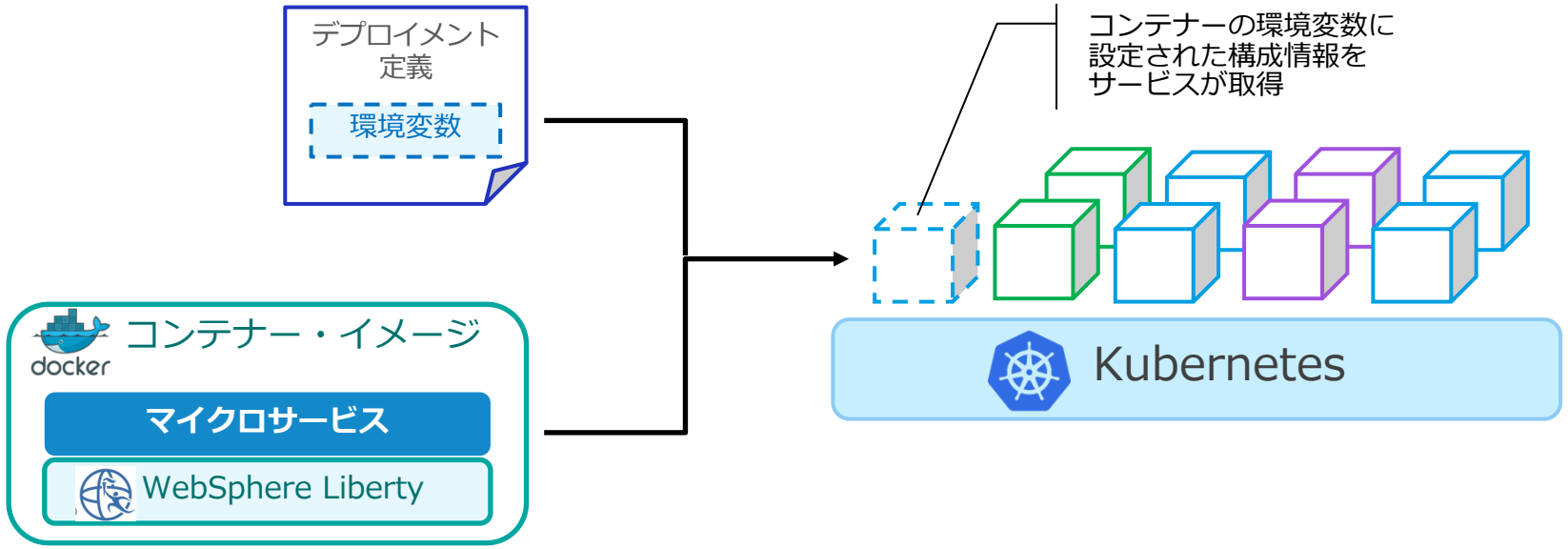


# 利用シナリオ

- コンテナー環境における利用例

  マイクロサービスでは実行環境としてコンテナーの利用が一般的になりつつありますが、コンテナーの特徴であるポータビリティーを確保するためには環境に依存する構成情報をコンテナーから外部化することが重要です。

  接続先のサービスやデータベースへの接続情報などの構成情報はMicroProfile ConfigのAPIを使用して取得するようにサービスを実装しておき、それらの値はKubernetesのデプロイメント定義上の環境変数として外部化しておく、といった利用シナリオが想定されます。



# フィーチャーの有効化

WAS Liberty上でMicroProfile Configの機能を有効化するためには、該当サーバーのserver.xmlにmpConfig-1.2フィーチャーを設定します。

```
<featureManager>
    <feature>mpConfig-1.2</feature>
</featureManager>
```

以下にリストされたフィーチャーを設定することで有効化（暗黙的ロード）することも可能です。

- microProfile-1.3
- mpMetrics-1.1
- mpOpenAPI-1.0

mpConfig-1.2フィーチャーを設定することで有効化されるフィーチャーはありません。

# MicroProfile Configの使用方法

# 構成ソース（ConfigSources）とordinal

デフォルトでは、以下の 3 つの構成ソース（ConfigSources）から構成情報を取得できます。構成ソースの優先順位はordinal値で指定され、値の大きいものが優先されます。

| ConfigSources | ordinal値 |
|---|---|
| Java のシステム･プロパティ | 400 |
| 環境変数 | 300 |
| クラス・パス内の全ての META-INF/microprofile-config.properties ファイル | 100 |

上記のordinal値はデフォルト値となり、構成ソース内でconfig_ordinalプロパティを設定することで上書きできます。

同じプロパティを持つ複数の構成ソースに同一のordinal値を設定してしまった場合、構成ソースのNameの値（getName()が返す値）の文字列ソート順で先にくる方の構成ソースが優先されます。

# カスタム構成ソース（Custom ConfigSources）（1/4）

デフォルトで使用可能な構成ソースの他に、任意の構成ソースを追加することができます。

カスタム構成ソースを作成するためには、org.eclipse.microprofile.config.spi.ConfigSourceインターフェースを実装します。

その後、リソースファイル（/META-INF/services/org.eclipse.microprofile.config.spi.ConfigSource）に作成したクラスを、完全修飾クラス名で登録します。

# カスタム構成ソース（Custom ConfigSources）（2/4）

- ConfigSource用インターフェース

| クラス |
| --- |
| org.eclipse.microprofile.config.spi.ConfigSource |

| メソッド | 説明 |
| --- | --- |
| String getName() | 構成ソースの名前を返します。 |
| int getOrdinal() | 構成ソースのordinal値を返します。 |
| Map<String, String> getProperties() | 構成ソースのプロパティを返します。 |
| Set<String> getPropertyNames() | 構成ソースのすべてのプロパティ名を返します。 |
| String getValue(String propertyName) | 指定されたプロパティの値を返します。 |

# カスタム構成ソース（Custom ConfigSources）（3/4）

以下はカスタム構成ソースのサンプルです。

```
public class CustomConfigSource implements ConfigSource {
        @Override
        public int getOrdinal() {
                return 112;
        }
        @Override
        public Set<String> getPropertyNames() {
                return readPropertyNames();
        }
        @Override
        public Map<String, String> getProperties() {
                return readPropertiesFromDb();
        }
        @Override
        public String getValue(String key) {
                return readPropertyFromDb(key);
        }
        @Override
        public String getName() {
                return "customDbConfig";
        }
}
```

# カスタム構成ソース（Custom ConfigSources）（4/4）

以下はorg.eclipse.microprofile.config.spi.ConfigSourceファイルのサンプルです。
完全修飾クラス名（fully-qualified class name）で登録を行います。

```
io.openliberty.guides.config.CustomConfigSource
```

# 値の取得方法（1/2）

MicroProfile Configではいくつか値を取得する方法があります。
これにはプログラムから値を取得するほか、アノテーションを用いた方法があります。

1. プログラムから値を取得するサンプル。

```
public class ConfigUsageSample {
        public void useTheConfig() {
                // Configインスタンスを取得します
                Config config = ConfigProvider.getConfig();
                String serverUrl = config.getValue("acme.myprj.some.url", String.class);
                callToServer(serverUrl);
        }
}
```

※org.eclipse.microprofile.config.Configインターフェースの API については以下リンク先を参照ください。
https://openliberty.io/docs/ref/microprofile/1.3/#package=org/eclipse/microprofile/config/package-frame.html&class=org/eclipse/microprofile/config/Config.html

# 値の取得方法（2/2）

2. アノテーションを用いて値を取得するサンプル。

```
@Inject
@ConfigProperty(name = "port_number", defaultValue="100")
private int portNumber;

public int getPortNumber() {
          return portNumber;
}
```

上のサンプルのようにアノテーションを用いて値を取得する場合、アプリケーション起動時に構成ソースから取得された値が注入され、その後構成ソース上の値が変更されても動的に再読み込みされることはありません。

いずれの構成済み構成ソースにもプロパティが存在しない場合は、defaultValueパラメーターを使用してデフォルト値を割り当てます。

※ConfigPropertyアノテーションの構文については以下リンク先を参照ください。

https://openliberty.io/docs/ref/microprofile/1.3/#package=org/eclipse/microprofile/config/inject/package-frame.html&class=org/eclipse/microprofile/config/inject/ConfigProperty.html

# 静的注入と動的注入

前頁のアノテーションによる値の取得方法は静的注入と呼ばれますが、構成ソースから値を動的に取得する動的注入も可能です。動的注入を行うためには、毎回動的に値をソースから読み込むように実装されたカスタム構成ソースの作成と、そのカスタム構成ソースに定義された値をjavax.inject.Provider型でインジェクションするという2点が必要になります。
javax.inject.Providerを使用した場合、get()メソッドの呼び出し時に構成ソースから値の取得が行われます。

以下は、動的注入（dynamic injection）にて、プロパティ"inventory_inMaintenance"を取得するサンプルです。

```
@Inject
@ConfigProperty(name = "inventory_inMaintenance")
private Provider<Boolean> inMaintenance;

public boolean isInMaintenance() {
          return inMaintenance.get();
}
```

# 標準装備コンバーターとPriority

構成ソースからプロパティを任意の型で取り出すためにコンバーターを使用できます。
以下の型については標準で用意されています。

- boolean / Boolean
- int / Integer
- long / Long
- float / Float
- double / Double
- Duration
- LocalTime
- LocalDate
- LocalDateTime
- OffsetDateTime
- OffsetTime
- Instant
- URL

コンバーターの優先順位は@Priorityアノテーションで制御されます。
標準装備コンバーターの優先順位は1です。

# カスタム・コンバーター（1/3）

org.eclipse.microprofile.config.spi.Converterを継承することで、カスタム・コンバーターを作成することができます。

作成したクラスは、/META-INF/services/org.eclipse.microprofile.config.spi.Converterに完全修飾クラス名で登録することで使用できます。

カスタム・コンバーターのデフォルトの優先順位（@Priority）は100です。

同一の型に変換するコンバーターが複数ある場合、優先度の高い(Priority値の高い)コンバーターが使用されます。

# カスタム・コンバーター（2/3）

- Converter用インターフェース

| クラス |
| --- |
| org.eclipse.microprofile.config.spi.Converter |

| メソッド | 説明 |
| --- | --- |
| T convert(String value) | Stringを指定された型に構成します。 |

# カスタム・コンバーター（3/3）

以下はカスタム・コンバーターのサンプルです。

```
package io.openliberty.guides.config;

import org.eclipse.microprofile.config.spi.Converter;
import io.openliberty.guides.config.Email;
import javax.annotation.Priority;

@Priority(200)
public class CustomEmailConverter implements Converter<Email> {

        @Override
        public Email convert(String value) {
                return new Email(value);
        }

}
```

以下はorg.eclipse.microprofile.config.spi.Converterファイルのサンプルです。

```
io.openliberty.guides.config.CustomEmailConverter
```

# 参考リンク

- 連載「Microservice Builder」第 2 回 MicroProfile Config を Kubernetes 環境で利用する
  https://www.ibm.com/developerworks/jp/websphere/library/icp/msb_introduction/2.html

- IBM Knowledge Center 「MicroProfile Config 1.2」
  https://www.ibm.com/support/knowledgecenter/ja/SSAW57_liberty/com.ibm.websphere.liberty.autogen.nd.doc/ae/rwlp_feature_mpConfig-1.2.html

- IBM Knowledge Center 「MicroProfile Config API」
  https://www.ibm.com/support/knowledgecenter/ja/SSAW57_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp_microprofile_overview.html

- Configuring microservices
  https://openliberty.io/guides/microprofile-config.html#enabling-dynamic-configuration-injection

# End of File