

マイグレーション

当セッションの目的

- WAS V9 traditionalへの移行に当たって必要な項目の全体像を把握する

Agenda

1. WASのマイグレーションパス
2. WAS V9 traditionalへの移行
3. アプリケーションの移行
4. 実行環境の移行

1. WASのマイグレーションパス

WASマイグレーションのきっかけ

- システム更改
- 使用ソフトウェアの陳腐化
 - ◆ WASのサポート終了
 - ◆ SSL技術の進歩
- 新しい環境への対応
 - ◆ 新ブラウザ・クライアントテクノロジー
 - ◆ モバイル
- 新しいアプリケーションアーキテクチャーの採用
 - ◆ マイクロサービスアーキテクチャー
 - ◆ HTML5アプリケーション
 - ◆ モバイル連携
 - ◆ クラウド環境との統合

WAS 7.0/8.0通常サポート終了と JDKサポート終了

- WAS 7.0 / 8.0は2018年4月に通常サポートが終了します
- WAS 8.5については, JDK 6は2018年4月以降, JDK 7は2019年9月以降は新規の修正は作成されません

2016	2017	2018	2019	2020
WAS V7.0		2018/4		2019/9
JDK 6				
WAS V8.0				
JDK 6				
WAS V8.5				
JDK 6				
JDK 7				
JDK 8				

セキュリティ：SSL/TSL技術の陳腐化

- 2011
 - ◆ BEAST (CVE-2011-3389) SSL,TLSのCBCモードの処理の初期化ベクトル決定に関する問題で生ずる脆弱性
- 2012
 - ◆ CRIME (CVE-2012-4929) TLS圧縮機能に存在する脆弱性
- 2013
 - ◆ LuckyThirteen (CVE-2013-0169)
SSL,TLS,DTLSのCBCモードの処理に存在する脆弱性
- 2014
 - ◆ POODLE (CVE-2014-3566)
SSL3.0のCBCモードの脆弱性
 - ◆ POODLE bites TLS (CVE-2014-8730)
POODLE同様の攻撃がTLS1.0でも利用できる
- 2015
 - ◆ FREAK (CVE-2015-0204 / CVE-2015-0138) 過去の暗号輸出規制時のグレードが使われる問題
 - ◆ Bar Mitzvah Attack (CVE-2015-2808) RC4自身の脆弱性
 - ◆ Logjam (CVE-2015-400)
Diffie-Hellman鍵交換プロトコルを使用したTLS接続で512ビットの輸出グレード暗号にダウングレードされる
- 2016
 - ◆ SLOTH (CVE-2015-7575 / CVE-2016-0201) TLS,MD5のHash衝突の脆弱性
 - ◆ DROWN (CVE-2016-0800) SSL v2でRSAベースの証明書を使っていると、同一証明書用いたTLS通信が解読される問題
- 2017
 - ◆ 多くのブラウザでSHA-1で署名された証明書が無効化



新しい環境への対応：新ブラウザモバイル

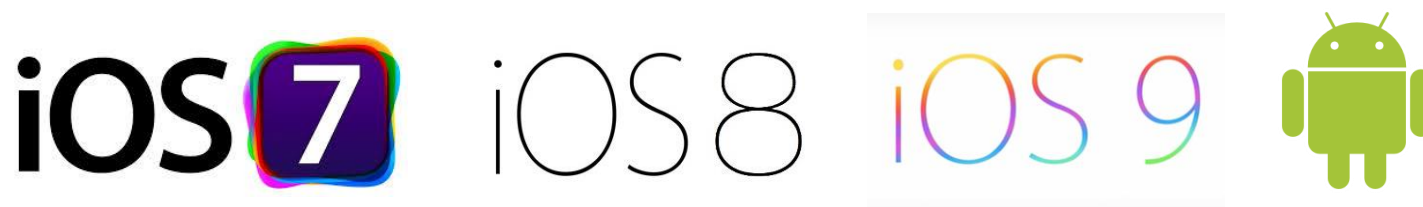
- 新しいブラウザ



- 進歩の速いクライアントサイドの技術



- 毎年変わるモバイルOS

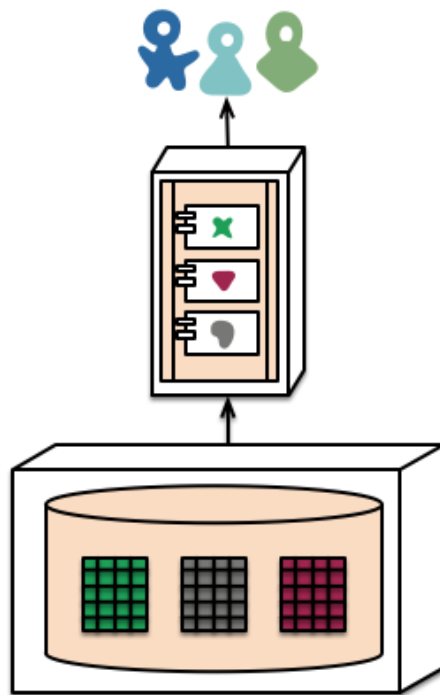


新しいアプリケーション・アーキテクチャー

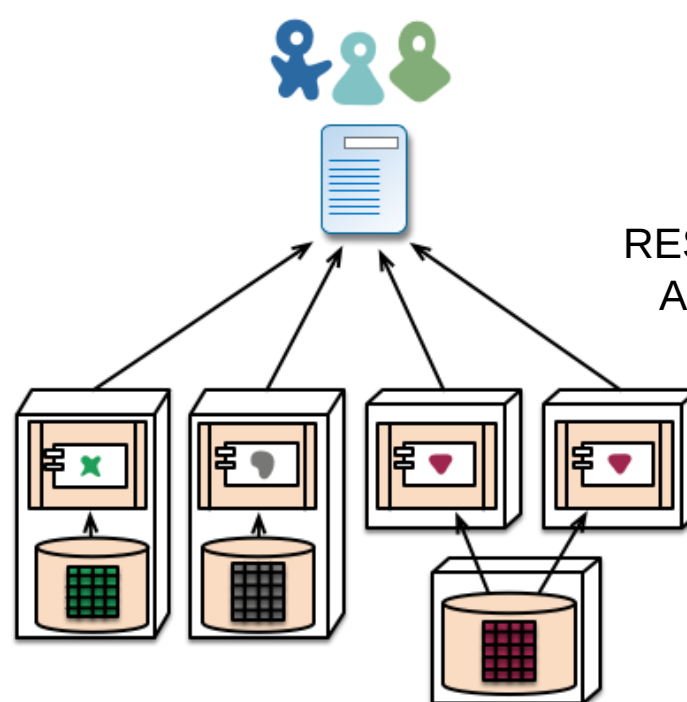
- 鍵となるのはアプリケーション機能のAPI化
 - ◆ アプリケーション機能を人間が操作するブラウザからだけでなく外部のプログラムから利用可能にする
- 呼び出し方式
 - ◆ SOAPによるWebサービス
 - ◆ RESTfulなWebサービス ← **おすすめ**
 - ◆ 独自プロトコル

Microservices Architectureアプリケーション

- 単一の（モノリシックな）アプリとして実装するのではなく、複数のサービスの統合として実装する
 - ◆ 変更の影響範囲を極小化
 - ◆ 機能の再利用を促進



monolith - single database

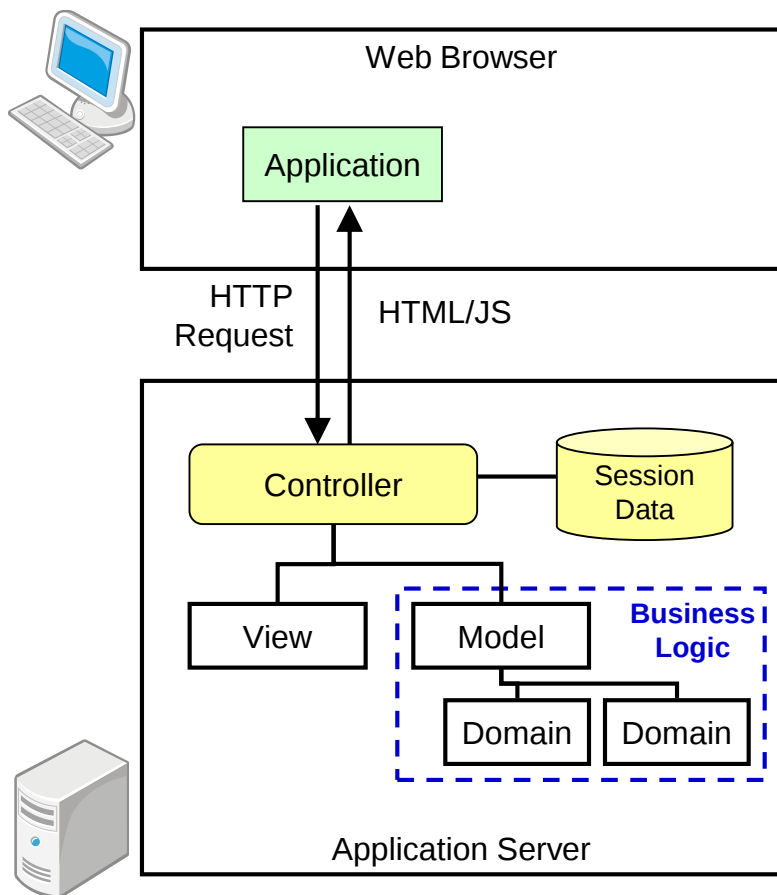


RESTを中心とした
APIによる連携

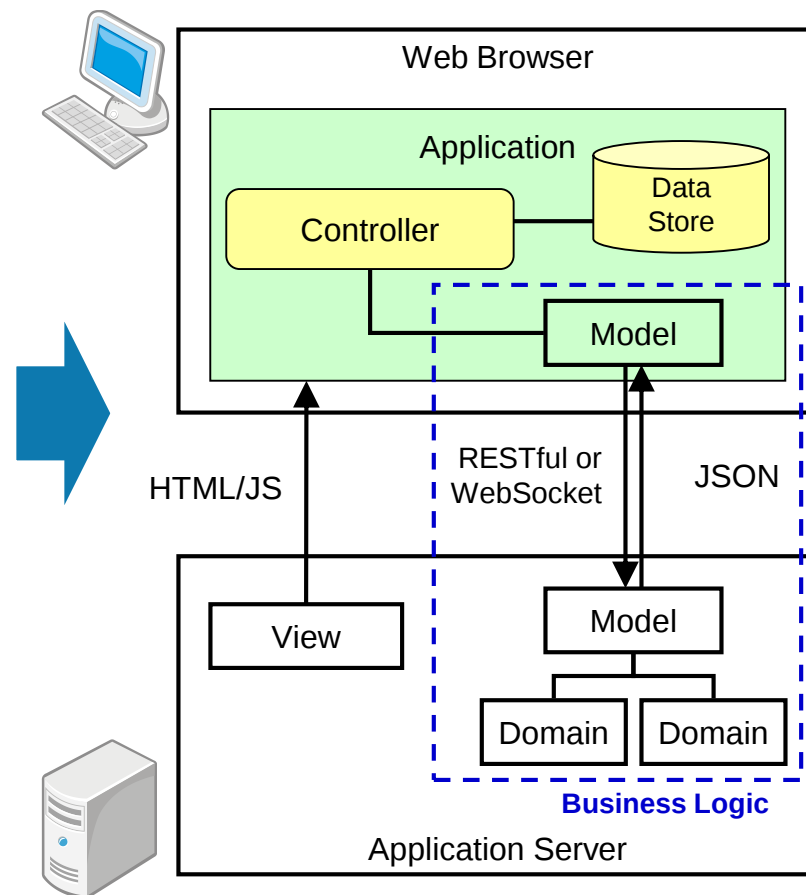
microservices - application databases

HTML5によるアプリケーションのモダン化

従来のWebアプリケーション



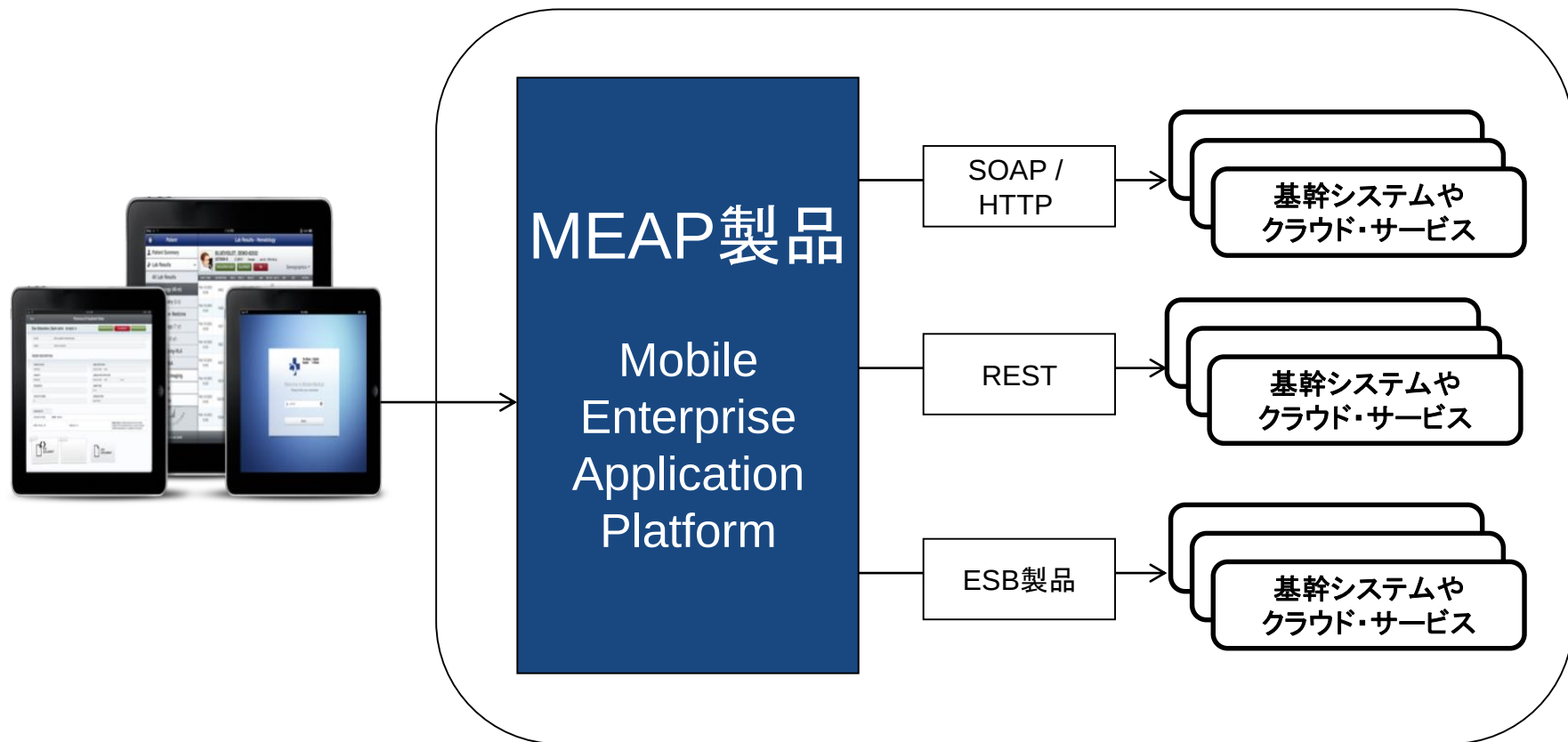
Single Page Application (SPA)



サーバーはクライアントアプリにサービス（API）を提供

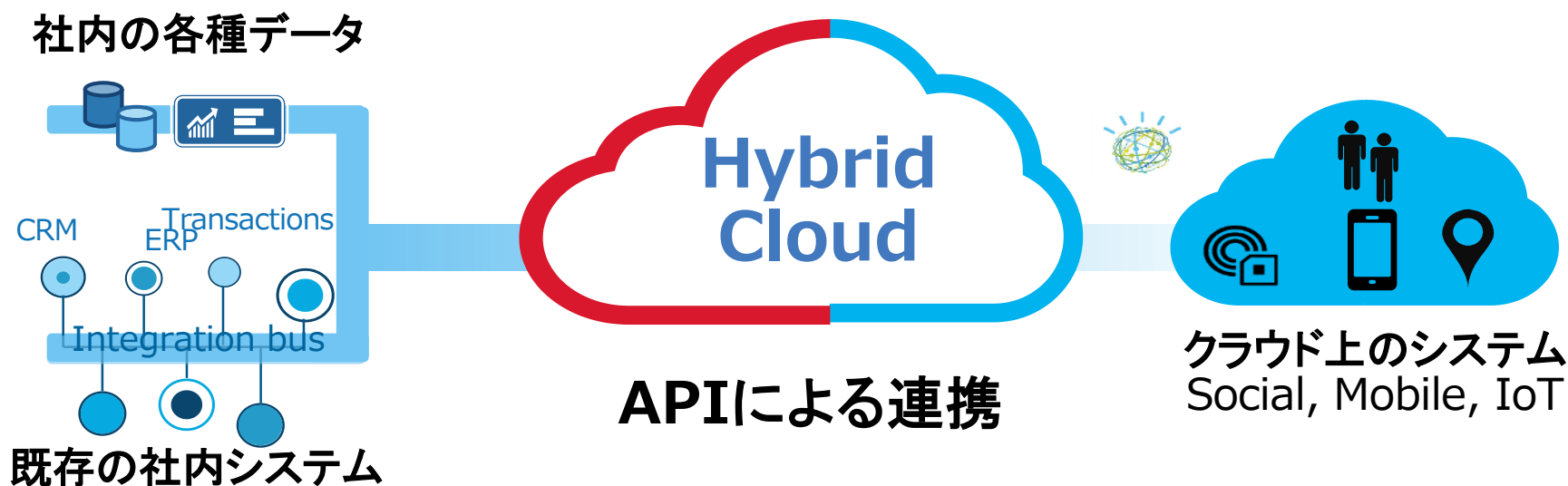
モバイルアプリケーションへの対応

- モバイルアプリケーションは（多くはMEAP経由で）バックエンドシステムをサービスとして利用



クラウドとの連携

- クラウドの活用は必須だが
全てをクラウドに出すことはできない
 - ◆ 社内システムと社外クラウドを連携させる
Hybrid Cloudへの挑戦



WAS V9 ～18年の歴史と進化～

□ WAS V9登場！！

Java EE 7

WAS V9

WAS V8.5.5.6 2016

WAS V8.5.5 2015

Java EE 6 WAS V8.5 2014

WAS V8.0 2013

2012

Java EE 5

WAS V7 FP 2011

WAS V7

2010

WAS V6.1
Feature Pack

2009

J2EE 1.4

WAS V6.1

2008

2007

WAS V6.0

2006

WAS V5.1

2005

J2EE 1.3

WAS V5.0

2004

WAS V4.0

2003

J2EE 1.2

WAS V3.5

2002

WAS V3.0

2001

WAS V2.0

2000

1999

WAS V3.0

EJB

WAS V4.0

J2EE 1.2

Webサービスサポート

動的キャッシュ

リソースアナライザー

Java 2

マルチOSサポート

WAS V5.1

JDK1.4

JSF

PME

最新のWS*

WAS V6.1

JDK 5

SIPサポート

開発ツール(AST)

最新のWS*

WAS V6.0

J2EE 1.4

HA機能拡張

SOA対応

新Messaging Engine

最新のWS*

WAS V7

Java EE 5 / JDK 6

柔軟な管理

ランタイム・プロビジョニング

コードとFixの集中管理

Java高速化(参照圧縮)

WAS V8.0

Java EE 6 / JDK 6

バッチ実行環境

ログ・トレース高速化

集中インストレーション管理

WAS V8.5.5

Liberty Core Edition提供

WXSをBASE/NDに同梱

WebサーバーPluginでの

インテリジェント管理

WAS V8.5

Libertyプロファイル

インテリジェント管理

JavaSE 7

WAS V8.5.5.6

WAS Liberty

Java EE 7対応

WAS V9 登場！！

Java EE 7, Java 8対応

API機能強化

API Connectを同梱

クラウド対応強化

WASで提供される二つのランタイム

バージョン	提供されるWASランタイム	
WAS V8.0		WAS Java EE6 完全対応
WAS V8.5	WAS Libertyプロファイル Servlet/JSPなど基本機能	WAS Fullプロファイル Java EE6 完全対応
WAS V8.5.5	WAS Libertyプロファイル Java EE6 Web Profile対応	WAS Fullプロファイル Java EE6 完全対応
WAS V8.5.5.6	WAS Libertyプロファイル Java EE7 完全対応	WAS Fullプロファイル Java EE6 完全対応
WAS V9.0	WAS Liberty Java EE7 完全対応	WAS traditional Java EE7 完全対応

用途に応じたランタイムの選択

- WAS traditional

- ◆ **既存資産の活用**を目的としたランタイム
 - ◆ 旧WASでおこなっていた**従来の運用を継続**したいお客様
- ◆ WAS Libertyで対応していない**旧API**を使用しているアプリケーションの実行環境として
 - ◆ JAX-RPCやEntity Bean, CommonJなど

- WAS Liberty

- ◆ モダンなアプリケーション開発・サーバー運用に対応した**新時代のランタイム**
 - ◆ 軽量さを活かしたAgile開発やCD（継続的デリバリー）
 - ◆ ツールによる運用の自動化・Platform as a Code
- ◆ **クラウド環境**での使用やリソースの限定されたIoT環境での使用

WAS traditional の Java SE / EEのサポート

WAS V6.1	J2EE 1.4 Servlet 2.4/JSP 2.0 EJB 2.1				J2SE 5.0
WAS V7.0	J2EE 1.4 Servlet 2.4/JSP 2.0 EJB 2.1	Java EE 5 Servlet 2.5/JSP 2.1 EJB 3.0			Java SE 6
WAS V8.0	J2EE 1.4 Servlet 2.4/JSP 2.0 EJB 2.1	Java EE 5 Servlet 2.5/JSP 2.1 EJB 3.0	Java EE 6 Servlet 3.0/JSP 2.2 EJB3.1 / JAX-RS1.0		Java SE 6
WAS V8.5	J2EE 1.4 Servlet 2.4/JSP 2.0 EJB 2.1	Java EE 5 Servlet 2.5/JSP 2.1 EJB 3.0	Java EE 6 Servlet 3.0/JSP 2.2 EJB3.1 / JAX-RS1.0		Java SE 6/7/8 (*1)
WAS V9.0		Java EE 5 Servlet 2.5/JSP 2.1 EJB 3.0	Java EE 6 Servlet 3.0/JSP 2.2 EJB3.1 / JAX-RS1.0	Java EE 7 Servlet 3.1/JSP 2.3 EJB3.2 / JAX-RS2.0	Java SE 8

(*1) Java SE 8のサポートはWAS V8.5.5.9以降

参考：WAS Liberty の Java SE / EEのサポート

WAS V8.5

Java EE 6
Web Profile(*1)
Servlet 3.0/JSP 2.2
JAX-RS1.0

Java EE 7 (*2)
Servlet 3.1/JSP 2.3
EJB3.2 / JAX-RS2.0

Java SE
6/7/8(*3)

WAS V9.0

Java EE 6
Web Profile
Servlet 3.0/JSP 2.2
JAX-RS1.0

Java EE 7
Servlet 3.1/JSP 2.3
EJB3.2 / JAX-RS2.0

Java SE 8

(*1) Java EE 6 Web ProfileのサポートはWAS V8.5.5以降

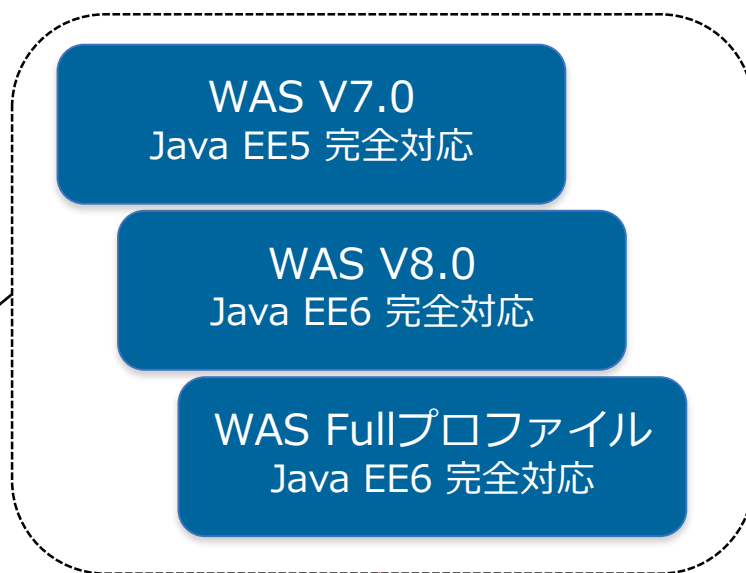
(*2) Java EE 7のサポートはWAS V8.5.5.6以降

(*3) Java SE 8のサポートはWAS V8.5.5.2以降

二つのマイグレーションパス

- WAS V7.0/8.0, WAS V8.5 Fullプロファイルから
 - ◆ WAS V9.0 traditional へ移行
 - ◆ WAS V9.0 Liberty へ移行

「マイグレーションガイド
・ WAS Liberty編」でカバー



この資料でカバー

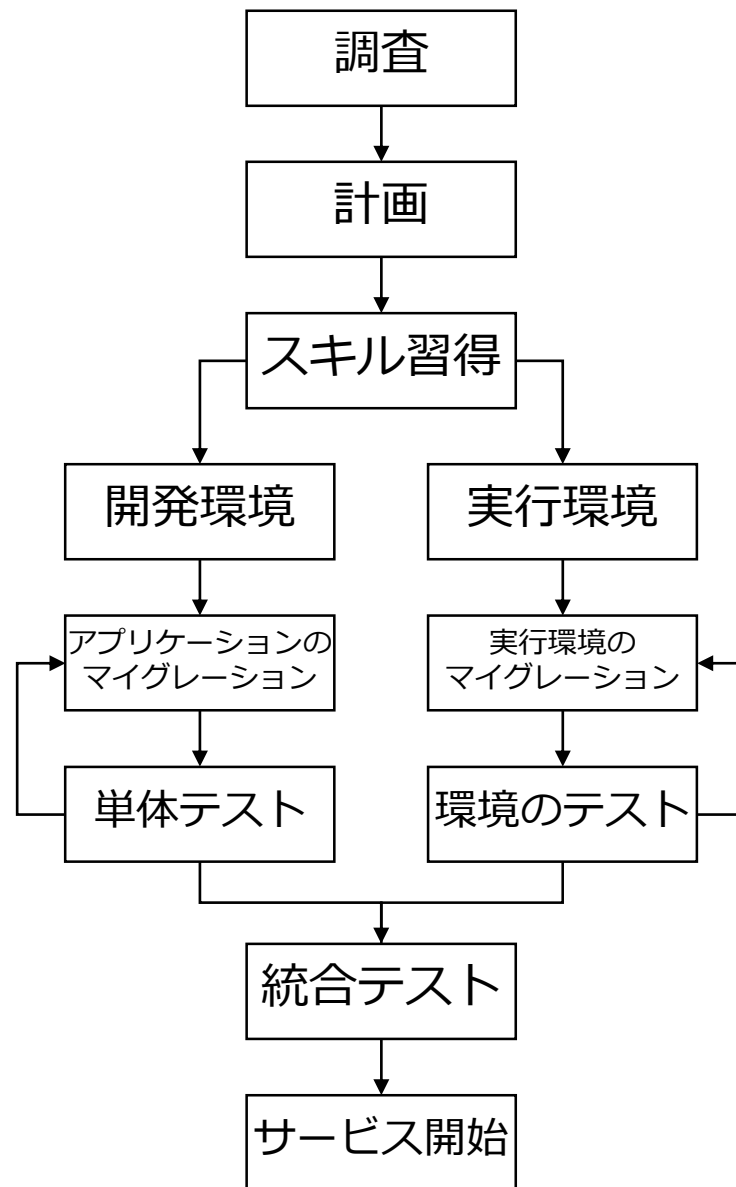
WAS V9.0 Liberty
Java EE7 完全対応

WAS V9.0 traditional
Java EE7 完全対応

2. WAS V9 traditionalへの移行

マイグレーションのロードマップ

- 調査
- 計画
- スキルの習得
- 開発環境
 - ◆ 開発環境の更新
 - ◆ アプリケーションのマイグレーション
 - ◆ 単体テスト
- 実行環境
 - ◆ 実行環境のマイグレーション
 - ◆ 実行環境のテスト
- 統合テスト
- サービス開始



WAS V9.0のエディション構成

WAS Family Edition

WAS ND



ミッション・クリティカルなアプリ向けに、可用性、高いパフォーマンス、高度な運用管理機能を提供。

WXSの全機能が利用可能

WAS for z/OS



z/OSのシスプレックスの機能を活用して、高いセキュリティ、高信頼性、優れたリソース活用を実現

WXS z/OS クライアント機能が利用可能

WAS (Base)

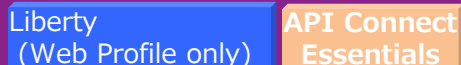


API Connect Essentials

Web層のクラスタリングと、セッション・フェイルオーバー機能の提供により、ある程度の規模の環境において、高いセキュリティと高パフォーマンスを提供するトランザクション・エンジン。

WXSのHTTP セッション・キャッシングと動的キャッシュが利用可能

WAS Liberty Core



軽量で低コストのLibertyプロファイル・ベースの製品。Java EEの全機能が不要なWebアプリケーションの稼動環境を迅速に構築。



Expressエディション Hypervisorエディションの廃止

- Expressエディション
 - ◆ 32bit版のみの提供だったExpressエディションは役割を終えV9からは提供されません
 - ◆ Java EE Webプロファイルで稼働するアプリケーションを使用している場合は、Liberty Coreへの移行を検討して下さい
- Hypervisorエディション
 - ◆ VMwareの仮想マシンイメージとして提供されていたWAS ND版
 - ◆ PureApplicationなどの最新の構築製品への移行が必要

各バージョンでのエディション構成

WAS 6.x	WAS 7.0/8.0/8.5	WAS 8.5.5	WAS 9.0
Express (Base) Network Deployment	Express (Base) Network Deployment Hypervisor Edition	Express Liberty Core (Base) Network Deployment Hypervisor Edition	Liberty Core (Base) Network Deployment

全プラットフォームの64bit対応

- 32bit版のJDKを使用したWASは、
全プラットフォームで提供されなくなりました
 - ◆ 64bit版のみが提供されます
- Javaのアプリケーションはbit数には依存しません
 - ◆ 基本的に移行に当たって、JDKのbit数の変更によるアプリケーションの書き換えは必要ありません
- JVMから呼び出されているライブラリは影響をうけます
 - ◆ JNIを使用してNativeのライブラリを呼び出している場合は、64bit環境で再コンパイルが必要
- パフォーマンスチューニングなどは再実施が必要
 - ◆ Javaヒープサイズの制限が大幅に緩和
 - ◆ 4Gを超えるヒープサイズではGCポリシーの検討が必要

関連ソフトウェアの更新

- ベンダーからのサポート状況や、組み合わせのサポート可否などの条件により、WASと同時に使用しているS/Wなどについても、更新が必要な場合もある
 - ◆ プラットフォーム（OS, H/W）
 - ◆ データベース
 - ◆ メッセージング製品 等
- WASとの連携がサポートされているS/Wについては、以下のURLで確認可能
 - ◆ <http://www.ibm.com/support/docview.wss?uid=swg27006921>

例) WASとの組み合わせがサポートされている製品のバージョン（製品出荷時点）

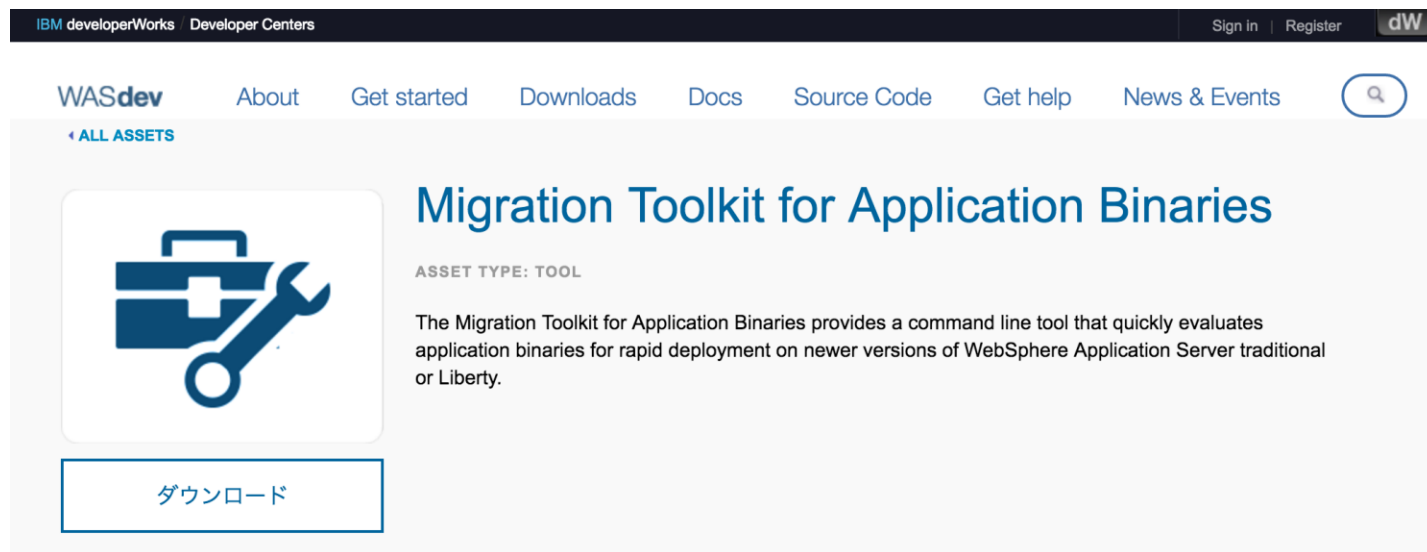
	WAS 6.1	WAS 7.0	WAS 8.0	WAS 8.5	WAS 9.0
Windows	2000 /2003/2008	2003/2008	2003R2 /2008/2008 R2	2008/2008 R2	2012/2012 R2
AIX	5.2/5.3/6.1	5.3/6.1	6.1/7.1	6.1/7.1	7.1/7.2
Solaris	9/10	9/10	10	10	11
Apache	2.0/2.2	2.0/2.2	2.2	2.2	2.2/2.4
DB2	8.1/8.2/9/9.5	8.1/8.2/9/9.5	9.1/9.5/9.7	9.1/9.5/9.7/10.1	9.7/10.1/10.5/11.1
Oracle	9i/10g	10g/11g	10g/11g	10g/11g	12c

3. アプリケーションの更新

Migration Toolkit for Application Binaries

- アプリケーションのEAR/WARファイルを分析し、使用されているAPIや機能を調査するツール
- アプリケーションがWAS V9 Liberty/traditionalなどで実行することができるか分析します
- 移行計画段階の簡易調査ツールとして利用できます

https://developer.ibm.com/wasdev/downloads/#asset/tools-Migration_Toolkit_for_Application_Binaries



Migration Toolkit for Application Binaries

■ 評価

```
java -jar binaryAppScanner.jar <WAR/EAR> --evaluate
```

- ◆ アプリケーションで使用されているAPIの一覧を表示

■ 分析

```
java -jar binaryAppScanner.jar <WAR/EAR> --analyze
```

```
--sourceAppServer=[was61|was70|was80|was855]
```

```
--sourceJava=[ibm5|ibm6|ibm7|oracle5|oracle6|oracle7]
```

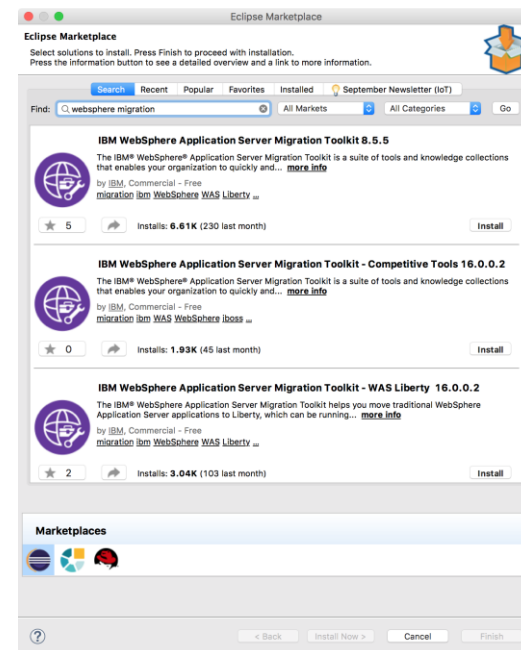
- ◆ アプリケーションで使用されているAPIのうち移行にあたって注意が必要なものを出力

■ デフォルトでは結果はHTMLファイルとして出力

- ◆ JSONやテキスト形式の出力もできます

IBM WAS Migration Toolkit

- Javaソースコード・構成ファイルを静的に解析して修正が必要な箇所・注意が必要な箇所を指摘
- Eclipseに組み込んで使用するツール
 - ◆ Eclipse Marketplaceからダウンロード導入可能
- 三種類の機能を提供
 - ◆ **WAS Migration Toolkit**
旧バージョンのWASからWAS V9 traditionalへ
 - ◆ **WAS Migration Toolkit WAS Liberty**
旧バージョンのWASからWAS V9 Libertyへ
 - ◆ **WAS Migration Toolkit Competitive Tools**
他社のアプリケーションサーバーからWASへ
- 無償で利用することができます



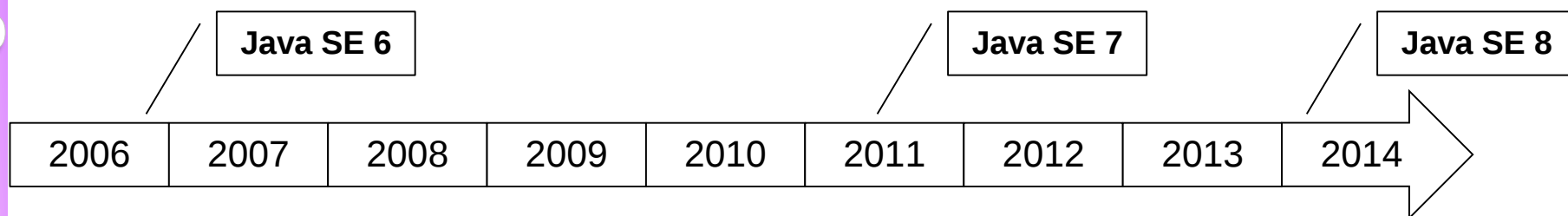
http://www.ibm.com/developerworks/jp/websphere/library/was/was_migration_toolkit/

WAS V9 traditionalへの移行にあたって 必要な主な手順

- Java SE 8への対応
- 非推奨・除去された機能を使用している場合は書き換え
- Java EE仕様については基本的な上位互換が取られている
 - ◆ WAS V7.0以降からであれば、原則としてそのまま移行できる
 - ◆ 非推奨となった機能，削除された機能を使用している場合は移行を検討
 - JAX-RPC： JAX-WSを使用して書き換え
 - EJB Entity Bean： JPAを使用して書き換え
 - JSF 1.2 Sun参照実装： 別の実装を使用するかJSF 2.xに移行

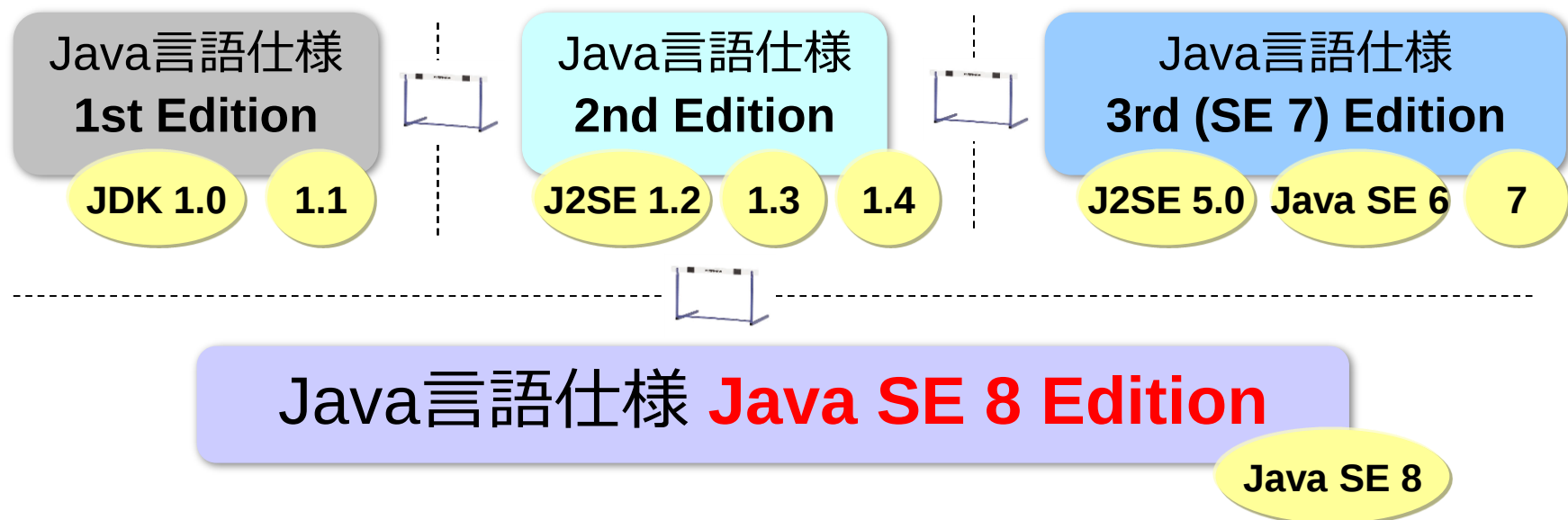
Java SE 8への移行

- WAS V9はLiberty/traditionalともに**Java SE 8のみ**が提供される
 - ◆ WAS Libertyで自前のJRE実行環境を別途用意する場合は、Java SE 6/7の使用も可能（2017年6月/2019年6月版まで）
- Java SE 8への対応が必要
 - ◆ Java SE 6は登場から10年以上が経過しておりトレンドから脱落
 - ◆ Java SE 8も2年以上にわたって使用されており実績も十分



Java Standard Editionの三度目の大変革

- Java SE 8では、言語仕様レベルの大きな変更が行われた
 - ◆ Project Lambda
- 過去二回（1.1→1.2および1.4→5.0）に匹敵するあるいは、それ以上のインパクトのある大変革

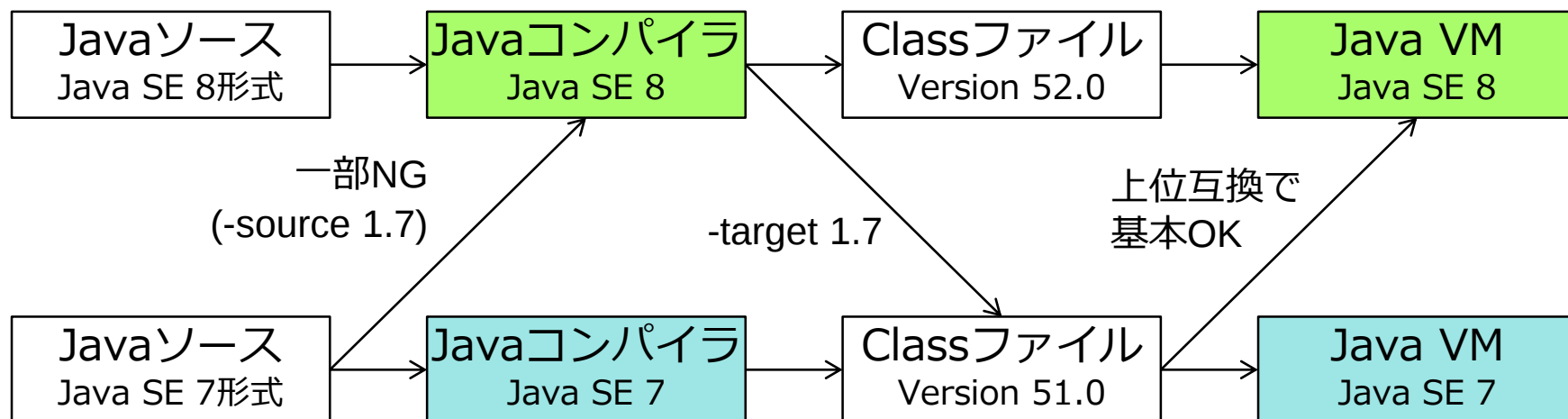


Java SE 8への移行

- バイナリ互換
 - Java SE 7/6用にコンパイルされたクラスファイルは基本的にそのまま稼働します
 - Javaクラスを直接操作する処理, クラス・インターフェースの構造に前提をおいている処理の中に, Java SE 8環境では動かないものがあります
 - 例: CDIコンテナ・コンパイルインターセプタなど
- ソースコード互換
 - Java SE 7以前の環境用に書かれたJavaソースファイルは, 一部の記述を修正する必要があることがあります
 - J2SE 5.0で導入されたGenericsを利用していないコードは, 使用するよう書き換えることを強く推奨します
 - Java SE 8の新機能の多くがGenericsの使用を前提としています

ソース互換とバイナリ互換

- ソース上位互換
 - ◆ Java SE 8のコンパイラで旧バージョンの形式のJavaソースをコンパイル
- バイナリ上位互換
 - ◆ 旧バージョンのコンパイラでコンパイルしたClassファイルをJava SE 8のJava VMで実行
 - ◆ -target 1.7等でコンパイルしたClassファイルをJava SE 8で実行



Java SE 8新機能 : Project Lambda

- Lambda式
- 型推論
- Method Reference
- Default Method
- Stream API



(引数) -> { 処理 }

オブジェクトとして「変数に代入」したり
「メソッドの引数にわたす」ことができる「コード片」

- Java SE 8の多くの機能の中核として利用されている

Java SE 8 : Generics（総称型）への対応

- 汎用的なオブジェクトを扱うクラスにおいて、扱うオブジェクトの型をパラメーターとして指定できるように
 - ◆ コンパイル時点での型の安全性の保証
 - ◆ 型キャストの排除
- コレクション・フレームワークをはじめとして多くの標準APIが総称型を使用したものに変更
 - ◆ 型を指定しないで使用するとコンパイル時に警告が出るように
- Java SE 8の新機能の多くがGenericsの使用を前提に
 - ◆ Genericsを利用していないと、型推論が使えないため、利用できない新機能が多数
 - ◆ 単なる警告ではなく実害があるようになったのでGenericsの使用を強く推奨

Genericsを使用するように変更したコードの例

```
private HashMap hosts; // 型は指定しない
public void init() throws IOException {
    hosts = new HashMap();
    hosts.put("localhost", InetAddress.getLocalHost());
    hosts.put("publicdns", "8.8.8.8"); // エラーとならない
}
public InetAddress getAddress(String hostname) {
    return (InetAddress)hosts.get(hostname); // キャストが必要
}
```

潜在的なバグ

ClassCastExceptionが
発生する可能性



```
private HashMap<String, InetAddress> hosts; // 型を指定する
public void init() throws IOException {
    hosts = new HashMap<String, InetAddress>();
    hosts.put("localhost", InetAddress.getLocalHost());
    hosts.put("publicdns", "8.8.8.8"); // コンパイルエラーとなる
}
public InetAddress getAddress(String hostname) {
    return hosts.get(hostname); // キャストは不要
}
```

Java SE : バージョン間の非互換への対応

- 仕様変更により一部にコンパイルが通らないコードや、コンパイルが正常に通った場合も、APIの挙動が変わることがあります
- JDK 6.0からJDK 7.0への移行
 - ◆ Exceptionをcatch節の中でRe-throwする際の言語仕様変更など軽微なもののみ
 - ◆ 非互換の詳細については、以下の文章を参照
<http://www.oracle.com/technetwork/java/javase/compatibility-417013.html>
- JDK 7.0からJDK 8.0への移行
 - ◆ J2SE 1.2で非推奨となっていたThread.stop()が無効化された、1024ビット未満のRSA鍵が拒否されるなどの動作変更がいくつかある
 - ◆ Classファイルの構造についてある種の前提をおいているコードは動かない
 - 「インターフェースはメソッド実装を持たない」
「同じアノテーションは重複しない」 など
 - CDIコンテナなどが動かなくなった例がある
 - ◆ 非互換の詳細については、以下の文書を参照
<http://www.oracle.com/technetwork/java/javase/overview/8-compatibility-guide-2156366.html>
<http://www.oracle.com/technetwork/jp/java/javase/overview/8-compatibility-guide-2156366-ja.html>
- その他、内部仕様に依存するアプリケーションの挙動が変わる場合も

参考：

WAS V8.5.5 FullプロファイルのJDKを8にあげる

- WAS V8.5.5 Fixpack 9以降（WAS V8.5.5.9以上）を適用
- 「Java SDK 8.0 for full profile and Liberty」を導入
 - ◆ いずれもサポートサイトよりダウンロード可能
 - ◆ IBM Installation Managerで適用・導入する
- managesdkコマンドで使用するJava SDKを切り替え
 - ◆ 管理コマンドが使用するJava SDKを切り替える
managesdk -setCommandDefault -sdkName 1.8_64
 - ◆ プロファイルが使用するJava SDKを切り替える
managesdk -enableProfileAll -sdkName 1.8_64
 - 32bit環境では「1.8_32」を使用する
 - ◆ 特定のアプリケーションサーバーの使用するJDKを
管理コンソールから変更することも可能

非推奨および 削除となった機能・安定化された機能（１）

■ 除去された機能（Removed features）

- ◆ そのバージョンから使用できなくなった機能
- ◆ 使用している場合は、必ず移行が必要

■ 非推奨となった機能（Deprecated features）

- ◆ 将来のバージョンで削除が予定される機能
- ◆ 基本的には、3年間、または、メジャー・リリースが2つ上がるまでは（そのどちらかの長い期間）サポートされる。
 - たとえばV6.0で非推奨となった機能については、V6.0およびV6.1の間はサポートするが、V7.0以降は製品から削除される可能性がある
- ◆ 2リリースよりも早く削除されることも（その場合は、Knowledge Centerに記述）

■ 安定化された機能（Stabilized features）

- ◆ 将来のバージョンでの削除は予定されていないが、代替機能があるため機能改善や新機能の追加は行われないもの。
- ◆ ただちに移行する必要はないが、代替機能の検討はおこなう

非推奨および 削除となった機能・安定化された機能（2）

- V9.0までの間に非推奨および削除となった機能の詳細についてはKnowledge Centerの以下の章を参照
“非推奨のフィーチャー、安定化されたフィーチャー、および除去されたフィーチャー”
http://www.ibm.com/support/knowledgecenter/ja/SSAW57_9.0.0/com.ibm.websphere.nd.multiplatform.doc/ae/rmig_depfeat.html
- V7.0から「安定化された機能」のカテゴリが追加されたため、以前は「非推奨」となっていた機能の一部が「安定化」されたものもある
 - ◆ 一度「安定化」されたものが、再度「非推奨」になることもあるので注意

WAS V9 traditionalで除去された機能（１）

- JavaServer Faces (JSF) 1.2 Sun リファレンス実装
- Edgeコンポーネント
 - ◆ Load Balancer for IPv4 （Legacy Load Balancer）
 - ◆ DMZ Secure Proxy Server for z/OS
- Communications Enabled Applications (CEA) , Service Component Architecture (SCA) の実行環境
- Web 2.0 and Mobile Toolkit
 - ◆ Dojo Toolkit, WebDAV 拡張機能, マップ変換などが削除
 - ◆ Webメッセージング・サービスは残っているが, 非推奨となっているため可能な限り非同期サブレットなどに移行

WAS V9 traditionalで除去された機能（２）

- WAS V7.0以前の環境への集中インストール
◆ バージョン混合セルでの集中インストールの使用は、
WAS V8.0/8.5/9.0に限定されます
- スクリプトによるWebサーバープラグインの構成
◆ pctツールを使用して構成
- HP-UX環境における
WebSphere カスタマイズ・ツールボックスのGUI画面
◆ コマンドラインからCUI環境を利用する
- IBM i 用リモート・インストール・ツール
- マイグレーションツールの一部のコマンド
◆ convertScriptCompatibility コマンド
◆ WebSphereConnectJDBCdriverConversion コマンド

WAS V9で非推奨となった主な機能

- Java EE仕様で非推奨となったもの
 - ◆ EJB Entity Bean
 - ◆ JAX-RPC (Java API for XML-based RPC)
 - ◆ JAX-R (Java API for XML Registries)
 - ◆ Java EE Application Deployment
- IBM独自機能
 - ◆ CommonJ / 非同期Bean
- サービス統合バス (SIBus)
 - ◆ MQ連携が非推奨に

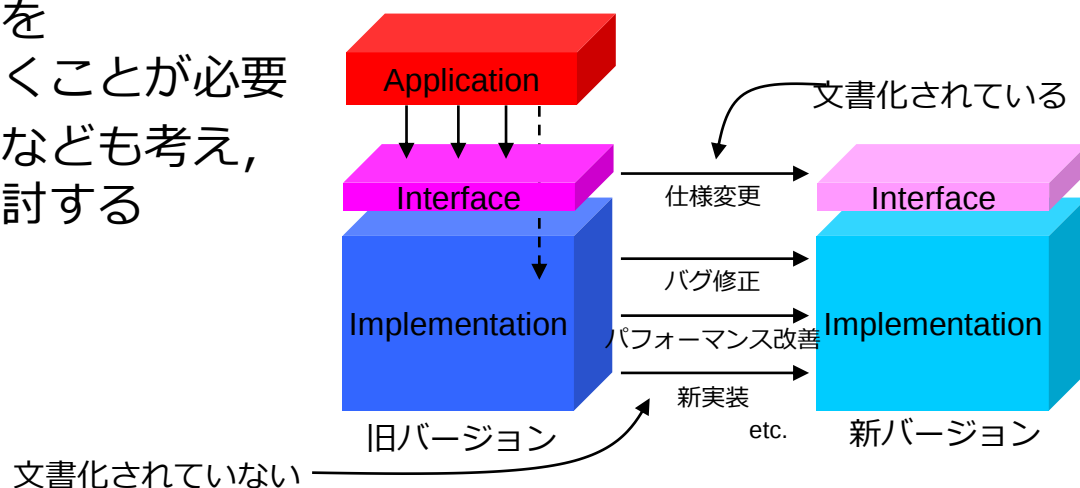
その他、 移行に当たって対処が必要な例（過去の例）

- サーバーAPIの仕様外の挙動の違い（バグが治ったことによる副作用など）
 - ◆ JSPでnullオブジェクトを出力したとき
 - ◆ CookieがないときのHttpServletRequestでgetCookies()の戻り値
 - ◆ JDBCで、executeQueryによる結果を返さないSQLの実行
 - ◆ RequestDispatcherでforwardからreturnした後の出力
 - ◆ includeされたServlet/JSPでのaddHeader/setHeaderしたとき
 - ◆ ユーザスレッドを作成し、スレッド間でDB接続を共有している
 - ◆ <jsp:useBean>のclass属性でJavaBean仕様に反するクラスを指定
 - ◆ etc.
- 非公開スเปックへの依存
 - ◆ WASがセットするCookieの文字列への依存
 - ◆ HttpSession#getIdで取得される文字列の内容への依存
 - ◆ サーバーのローカルディレクトリ構造への依存
 - ◆ WebSphere内部クラスの使用
 - ◆ etc.

これらはWAS V7.0以降からの移行では発生しませんが
同種の問題が発生する可能性があります

単体テスト・統合テストの重要性

- 前ページのような問題は、事前のコードレビューで発見することが難しい
 - ◆ 「仕様の変更」は文書化されているが、
「仕様外の動作の変更」はほとんど文書化されていない
 - ◆ 開発環境のコードチェック機能でも発見できないケースが多い
 - ◆ 問題が発生するパターンが無数に考えられ、事前検証で網羅することが不可能
- テストを実行して発見することが最良の手段
 - ◆ 最低限必要な修正を加えたら、まずは新環境でテストを行い問題の洗い出しを行う
 - ◆ 十分な期間のテストをスケジュールしておくことが必要
 - ◆ 運用後のパッチ適用なども考え、テストの自動化も検討する



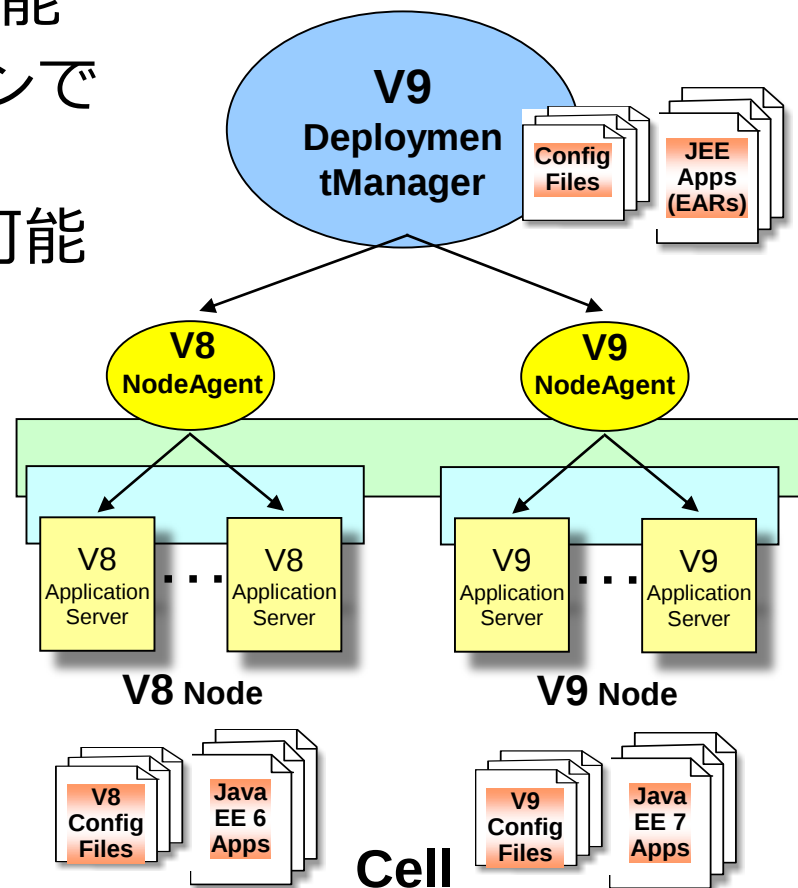
4. 実行環境の更新

WAS V7.0/8.0/8.5からWAS V9.0 traditional

- 基本的に管理機能の上位互換が取られている
 - ◆ トポロジー：Deployment Manager/Node Agent/AppServer
- サーバートポロジーや運用スクリプトなどはほとんどそのまま使用することができる
- 既存環境からの構成は、マイグレーションツールで移行可能
- 移行の過程で、セル内に異なるバージョンが共存することも可能
 - ◆ ノード内は同一のバージョン
 - ◆ Deployment Managerのバージョンが最新である必要がある

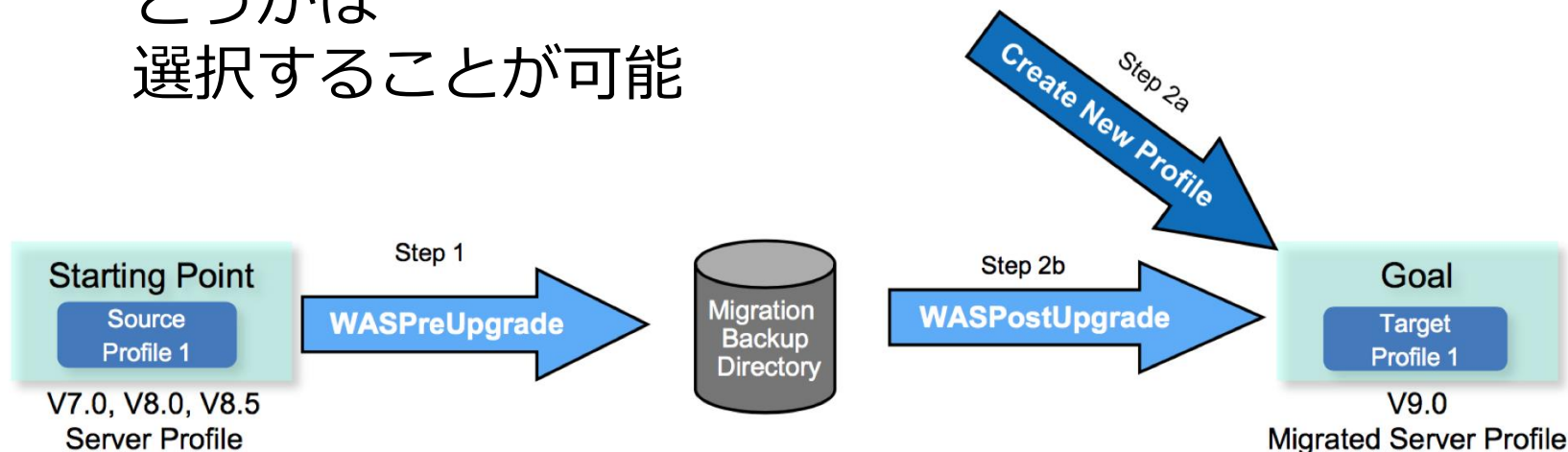
異なるバージョンの混在

- 1つのCell内で
V8.5, V8.0, V7.0とV9.0の
ノードを混在することが可能
- DMはより新しいバージョンで
なければならない
- 旧ノードは異なるOSでも可能



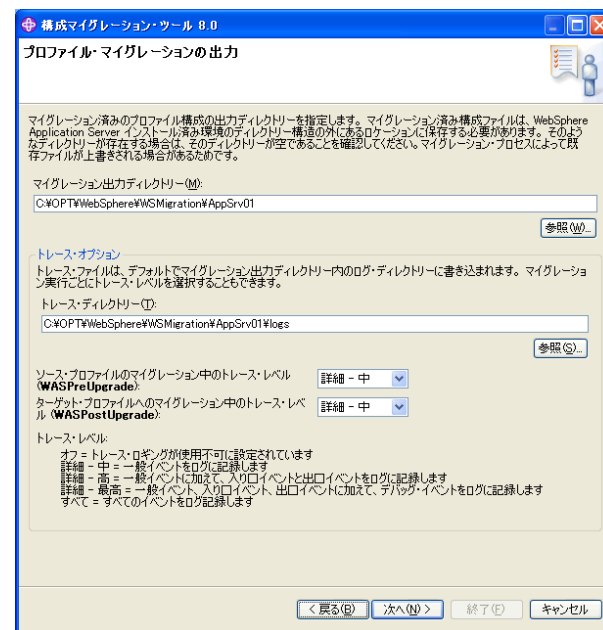
ツールによる構成のマイグレーション

- WASPreUpgrade/WASPostUpgradeによるマイグレーションが可能
 - ◆ Step 1 既存環境から構成情報をバックアップ
 - ◆ Step 2a 新規に作成したV9環境に
 - ◆ Step 2b バックアップされた旧環境の情報を反映
- 旧環境は削除されていないので、切り戻すことも可能
- 導入されているアプリケーションを含めて移行するかどうかは
選択することが可能



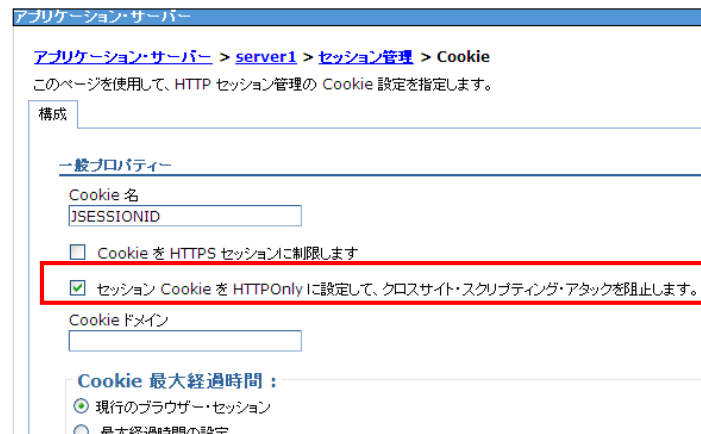
WASPreUpgrade / WASPostUpgrade

- コマンドラインツール
 - ◆ WASPreUpgrade
 - ◆ WASPostUpgrade
- GUIツール
 - ◆ 構成マイグレーションツール
 - WAS Customization Toolboxから起動
 - コマンドラインツールのフロントエンド・プログラム
 - 指定されたパラメーターでWASPreUpgradeとWASPostUpgradeが実行される



ハードニングによる影響

- WAS V8.0から、セキュリティを向上させるためのハードニングとして、いくつかのパラメーターのデフォルト値が変更されています
 - ◆ この影響でアプリケーションの動作に影響が出る可能性があります
- WASの利用するCookieへのHttpOnlyオプションのデフォルトの付加
 - ◆ WASがセッション維持に使用しているCookie, 認証情報を維持しているCookieに、デフォルトでHttpOnlyオプションがつくようになりました
 - JSESSIONIDやLTPAなど
 - ◆ HttpOnlyが付加されたCookieは、JavaScriptからのアクセスが制限されるほか、ブラウザによってはWebページに埋め込まれたJava Appletやフラッシュなどからも使用出来なくなります
 - アプリケーションがこれらの用途でCookieにアクセスしている場合、動作の不良が発生する可能性があります
 - ◆ 問題が発生した場合は、管理コンソールから設定を外して下さい



Load Balancer for IPv4の除去

- 従来提供されていたカーネルエクステンションベースの Load Balancer for IPv4 (LLB : Legacy LoadBalancer) は削除された
 - ◆ WAS V6.1で非推奨となり、その後安定化されたが
対応プラットフォームがWAS V9.0のサポートからはずれたために削除
- WAS V6.0から同梱されているLoad Balancer for IPv4 and IPv6 (ULB : User space LoadBalancer) へ移行
- LLBとULBの機能差に注意が必要
 - ◆ LLBではできた「自サーバーへのDispatch」がULBではできない
 - ◆ LoadBalancerに、いわゆる「Sorry Server」を兼任させていた場合、
トポロジの変更が必要になるケースがある
 - ◆ Windows環境では、LLBよりもULBのほうがパフォーマンスが悪い
パフォーマンス要件が満たせない場合はAIX/Linux環境を利用する

Edge Component／LLBとULBの違い

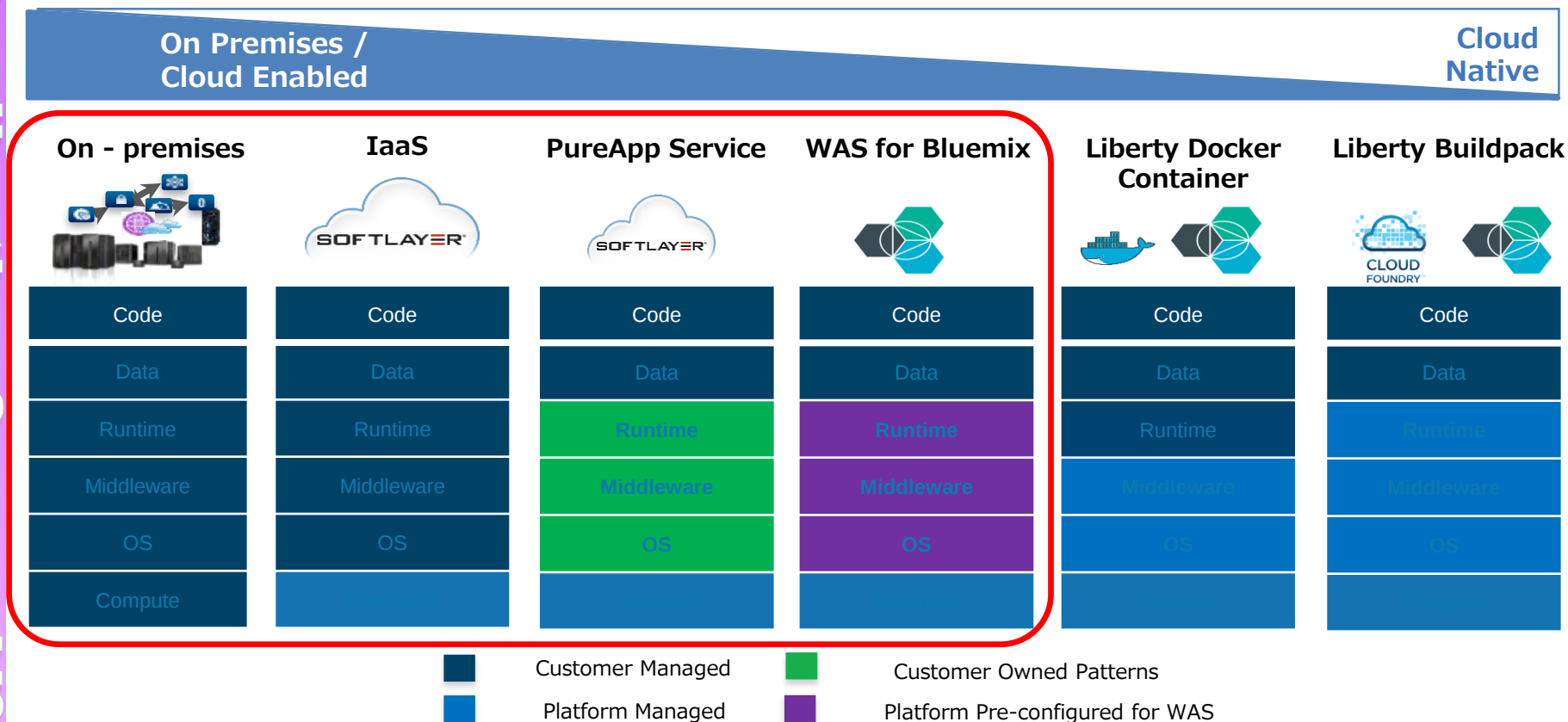
		LLB	ULB
①	転送方式	MAC、NAT、KCBR、CP+CBR	MAC転送、カプセル化転送
②	区切り文字	:	@
③	アフィニティー方式	IP Sticky、Cookie	IP Sticky
④	ルール	10種類	2種類
⑤	dscontrol server downコマンド	使用可能	使用不可
⑥	同一筐体内への転送	可能	不可
⑦	テイクオーバー時の接続/アフィニティー情報の引き継ぎ	引き継ぎ可能	引き継ぎ不可
⑧	goActiveスクリプトによるクラスター・アドレスの設定	必須	自動
⑨	serverDownスクリプトの挙動	従来通り	変更
⑩	インストールディレクトリ	<EDGE_ROOT>/lb	<EDGE_ROOT>/ulb
⑪	ワイルドカード・クラスター/ポート	使用可能	使用不可
⑫	ICMPの転送	しない	する

管理スクリプト

- コマンドライン管理ツールwsadminでは二つのスクリプト言語が使用可能
 - ◆ Jacl
 - ◆ Jython
- Jaclスクリプトは非推奨に
- 新規のスクリプト開発はJythonでおこなう

移行にあたってのクラウドの活用

- WAS traditionalもさまざまなクラウドで稼動可能
 - ◆ IaaS (SoftLayer, AWS, Azure)
 - ◆ PaaS (WAS for Bluemix, PureApplication)



WAS for Bluemix

- WASの導入された仮想マシンをクラウド上でサービスとして提供



WebSphere
Application Server

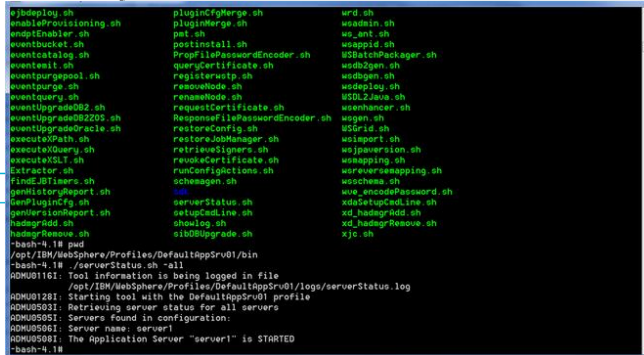
IBM

PUBLISH DATE
12/18/2015

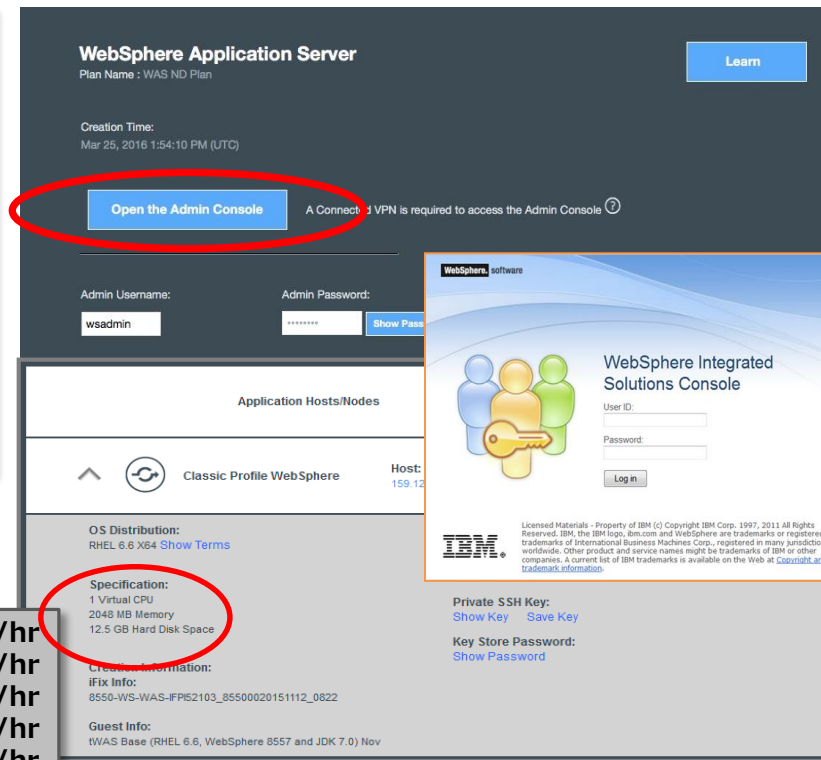
TYPE
Service

[VIEW DOCS](#)

Pick a plan		
Monthly prices shown are for country or region: United Kingdom		
Plan	Features	
WAS Liberty Core Plan	WebSphere Application Server Liberty Core. Red Hat Linux Guest.	£0.25 GBP/Hour
☒ WAS Base Plan	WebSphere Application Server Classic Base. Red Hat Linux Guest.	£0.32 GBP/Hour



S	- 2GB RAM,	1vCP,	12.5 GB disk	- WAS ND	- \$0.70/hr
M	- 4GB RAM,	2vCP,	25 GB disk	- WAS ND	- \$1.40/hr
L	- 8GB RAM,	4vCP,	50 GB disk	- WAS ND	- \$2.80/hr
XL	-16GB RAM,	8vCP,	100 GB disk	- WAS ND	- \$5.60/hr
XXL	-32GB RAM,	16vCP,	200 GB disk	- WAS ND	- \$11.20/hr



Variable VM sizes

5. まとめ

まとめ

- WAS traditionalは互換性を優先したランタイムなので既存のWAS環境からは最小限の負荷で移行が可能
- アプリケーションの移行はJava 8対応および廃止・非推奨となった機能の置き換え
- 実行環境の移行は既存環境の設計がそのまま利用可能

参照情報

- Knowledge Center 「マイグレーション、共存、および相互運用」
http://www.ibm.com/support/knowledgecenter/ja/SSAW57_9.0.0/com.ibm.websphere.nd.multiplatform.doc/ae/welc6topmigrating.html
- Knowledge Collection: Migration Planning and Resources
<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27008724>
- IBM WebSphere Application Server Migration Toolkit
<https://www.ibm.com/developerworks/websphere/downloads/migtoolkit/>
- RedBooks "WebSphere Application Server V8.5 Migration Guide"
<http://www.redbooks.ibm.com/abstracts/sg248048.html>

参考情報

- WAS V8.0 によるWebシステム基盤設計ワークショップ資料
https://www.ibm.com/developerworks/jp/websphere/library/was/was8_guide/
- WAS V9 Liberty基盤設計セミナー資料
https://www.ibm.com/developerworks/jp/websphere/library/was/liberty_v9_infradesign/
- WebSphere Application Server V9.0へのマイグレーションガイド - WAS traditional編 -
https://www.ibm.com/developerworks/jp/websphere/library/was/was9_twas_migration/

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客さま事例は、それらのお客さまがどのようにIBM製品を使用したか、またそれらのお客さまが達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客さまごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、WebSphere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。

Windowsは Microsoft Corporationの米国およびその他の国における商標です。

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。