WAS Liberty 最新情報セミナー 2015

～ 次世代ランタイム Libertyで変わる Java EE システム～

1. Libertyの概要と適用エリア

© 2015 IBM Corporation

1

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2015年7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。

© 2015 IBM Corporation

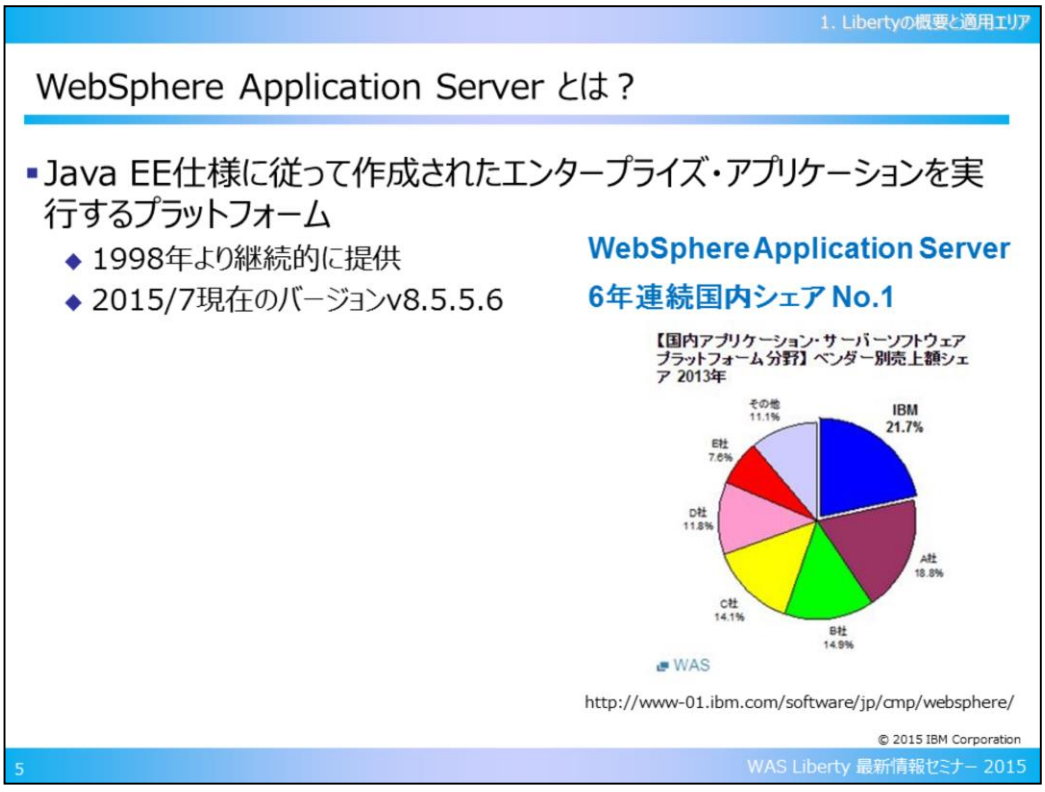
目次

- Libertyとは？
- Liberty登場の背景とアプリケーション・サーバーに求められる要件の変化
- Libertyの特長
- Libertyの適用エリア

© 2015 IBM Corporation

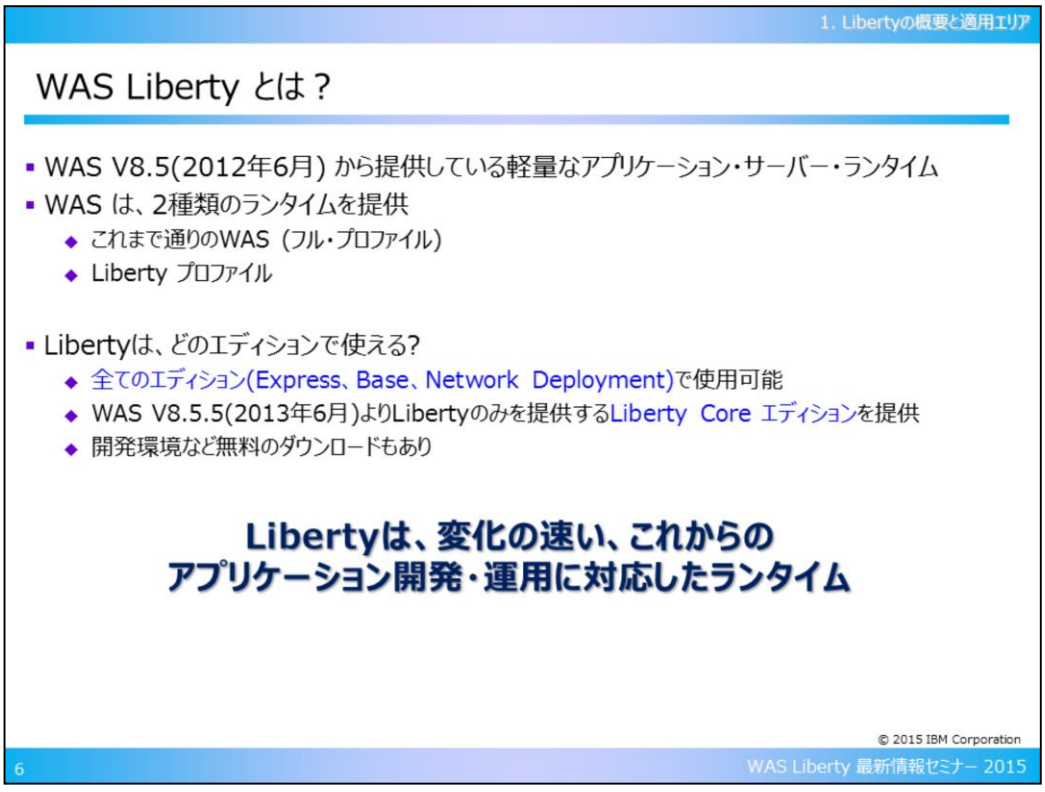
Libertyとは？

© 2015 IBM Corporation



WebSphere Application Server は、Java Enterprise Edition(Java EE)の使用に従って実装された、エンタープライズ向けアプリケーションを稼動させるランタイムです。IBMは、1998年より一貫して自社で開発を継続しています。2015年7月6日現在の最新のバージョンは、WAS V8.5.5.6です。

市場シェアも高く、2014年7月にIDC Japan株式会社が発表した「国内アプリケーション・サーバーソフトウェアプラットフォーム分野」で2013年国内シェア1位を獲得しています。国内シェアは2008年から6年連続で1位を獲得しています。



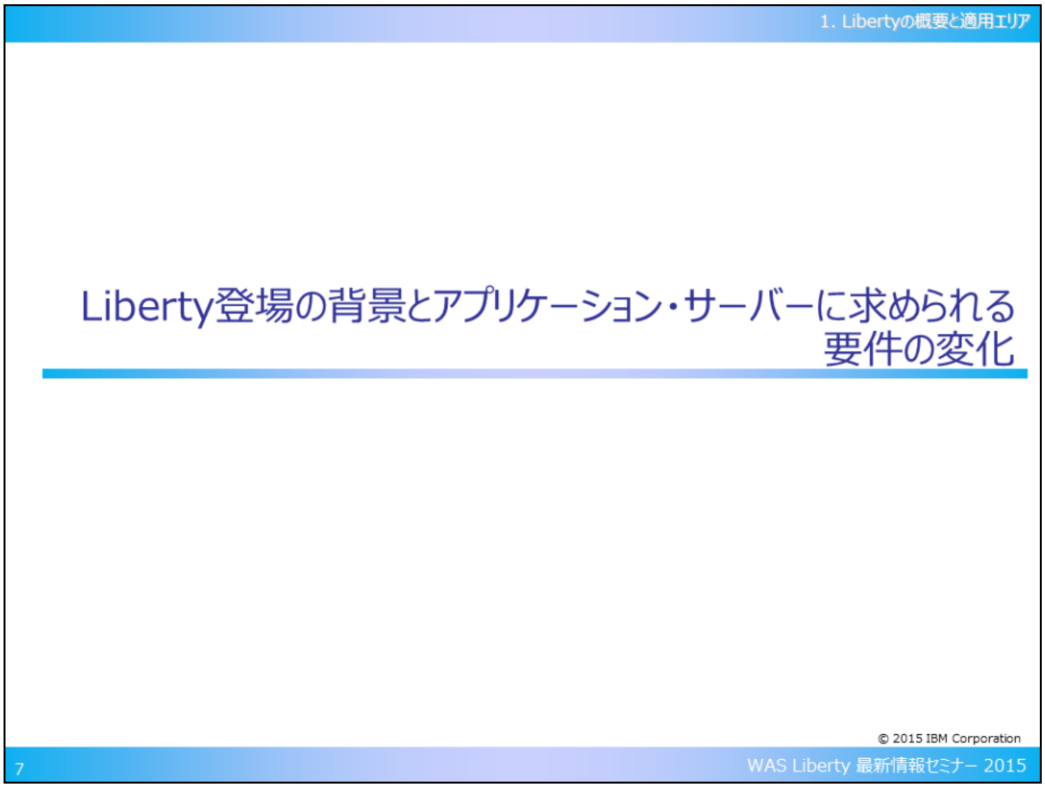
WAS Libertyは、WAS V8.5（2012年6月）より提供を開始した軽量なアプリケーション・サーバー・ランタイムです。WAS Libertyという新しい製品を出したわけではなく、WASに2種類のランタイムが含まれるようになりました。1つは、これまで通りのWAS（フル・プロファイル）と、もう1つが、Libertyプロファイルです。

Libertyプロファイルは、WASがこれまで提供していた全てのエディション（Express、Base、Network Deployment）に含まれ、使用できます。また、2013年6月に、WAS v8.5.5が出荷したタイミングで、WAS Liberty Core エディションという新しいエディションが提供されました。Liberty Core エディションは、Liberty ランタイムのみ利用できます。また、開発者の方が利用するためのランタイムに関しては、以下リンク先より無料のダウンロードが可能です。

https://developer.ibm.com/wasdev/

2つのランタイムの違いの詳しい解説は、「2. Libertyのエディション&フル・プロファイルとの違い」を参照ください。

Libertyは、変化が早く継続的に改善が必要なアプリケーションの開発・運用に対応したランタイムです。



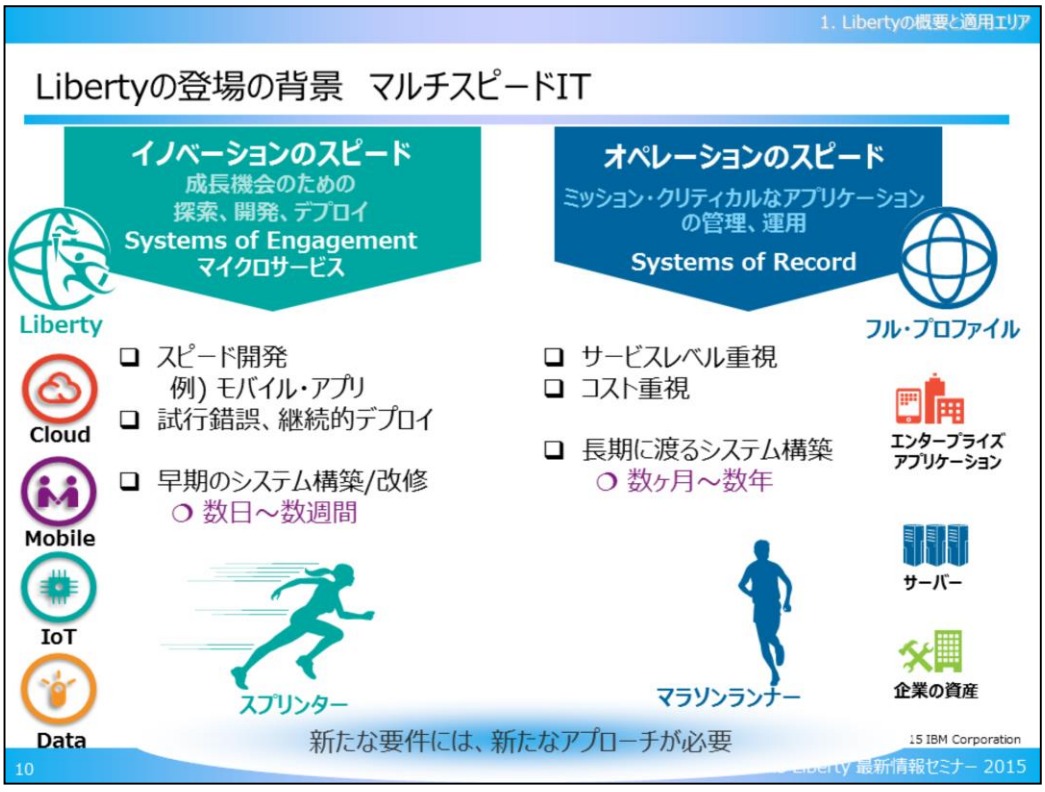
ここでは、Libertyが登場してきた背景について説明します。

これまで企業は、SoR(Systems of Record) と呼ばれる基幹システムに代表されるような、自社のビジネスに関係する情報を記録するシステムを作り、企業内の閉じたシステムと、記録されたデータでビジネスを行ってきました。10年程前から、Google、Twitter、Facebook、国内だとMixi、GREEなどのネット系企業が台頭してきました。これらの企業は、SoE(Systems of Engagement) と呼ばれるSNSなどに代表される人とかかわりあうWebのサービスを展開し、それまではデジタル化されていなかった個人の思考、嗜好、関係性が現在進行形でデジタル化しています。

更に、ここ5年程で、Internet of Thingsの世界が現実のものになってきています。モバイルの普及によりモバイル端末から位置情報の取得、Beaconなどを利用した様々なサービスも提供されはじめています。その他にも、センサーからの情報を利用した、スマートメーターや、Connected Car、スマート家電、スマートハウスなど様々なことをサービス化しようという試みが行われています。

このように個人に関連する情報、デバイスに関連する情報がデジタル化された世界で、企業だけがそれらの情報と無関係でい続けることは不可能です。

一般企業は、これまでいわゆるネット系企業が得意としていたSoEやIoTの領域に踏み込み、SoEやIoTのデータと自社内に持つデータをうまく組み合わせ、新しいビジネスを構築することが求められています。SoRの世界は、ある程度「かくあるべし」というビジネス要件がはっきりしていましたが、SoE、IoTの世界は、決まった要件はありません。ユーザーやシステムで産まれてくるデータにあわせて、常に変化し続けることが求められる世界です。



SoR、SoEのシステムを構築、運用することを考えるとどういう特徴があるでしょうか？

SoRのシステムは、基幹システムに代表されるようにビジネス要件の変化が少なく、いったん構築されると数年単位で利用されることが多く、マラソンランナーに例えることができます。SoRのシステムに一番求められるのは、品質です。IT部門や多くのSIerは、こうしたSoRシステムの構築・運用をこれまで行ってきています。WASでは、フル・プロファイルがこうしたシステムで利用されてきました。

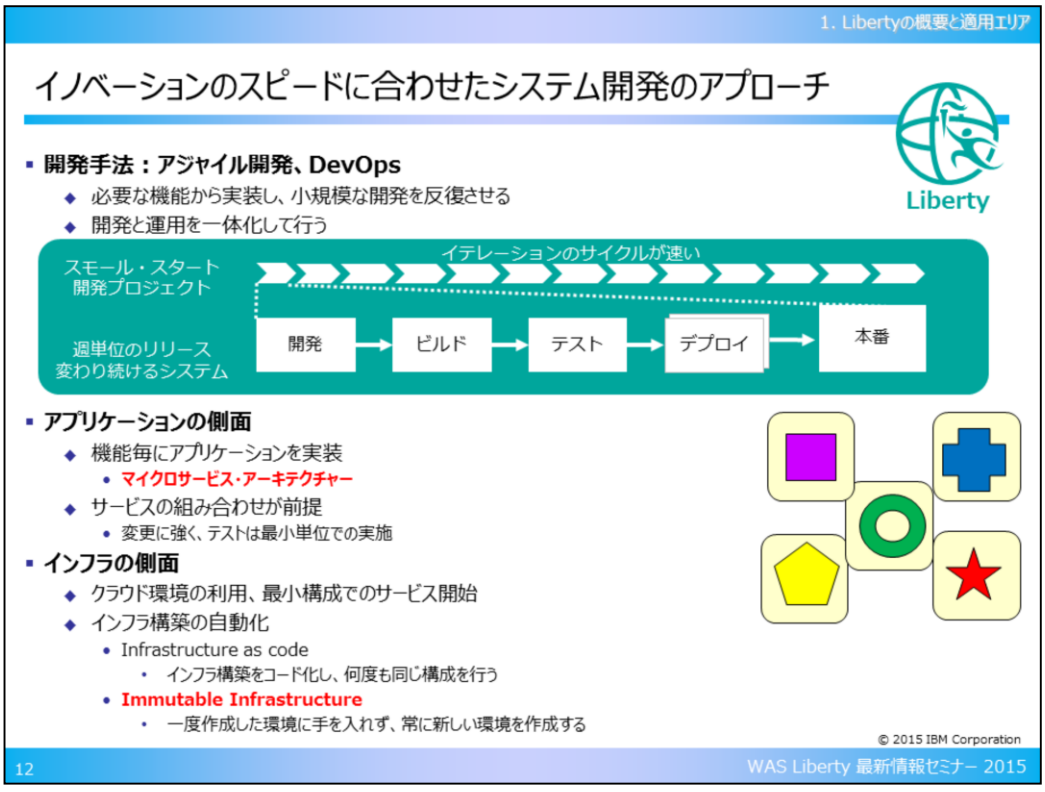
一方、SoEのシステムは、ビジネス要件の変化が激しく、素早く構築することが求められます。構築後も、継続的な機能追加、改善が求められ、スプリンターに例えることができます。SoEのシステムに一番求められるのは、スピードです。ここ数年こういったSoEのシステムは、IT部門主導ではなく、ユーザー部門などのLOB(Line of Business)主導で行われ始めています。Libertyはこうした所で活躍できるランタイムです。

もう少し詳しく、SoR、SoEのシステム開発、運用の違いを、開発手法、アプリケーション、インフラの3つの観点で特長を挙げます。

開発手法：SoRのシステムは、要件があまり変わらない、また大規模プロジェクトになりがちなため、開発手法としてはウォーターフォール型になることが多いです。

アプリケーション：全ての機能を一つのアプリケーションとして実装する、いわゆるモノリシック・アプリケーションで多くの場合作成されます。モノリシック・アプリケーションは、機能追加や機能変化の要件には、アプリケーション全てのテストが必要になり、負荷が高くなります。

インフラ：数年後を見据えたHWを用意し、モノリシック・アプリケーションを動かすためのOSやMWの詳細な構成設計/設定が行われます。また、そのシステムに固有なパフォーマンス・チューニングが実施されます。SoRのシステムは動作していることが重要であるため、いったん設定したOSやMWは、セキュリティ上の問題などどうしても変更が必要な場合を除き、極力変更は行わず稼動させます。



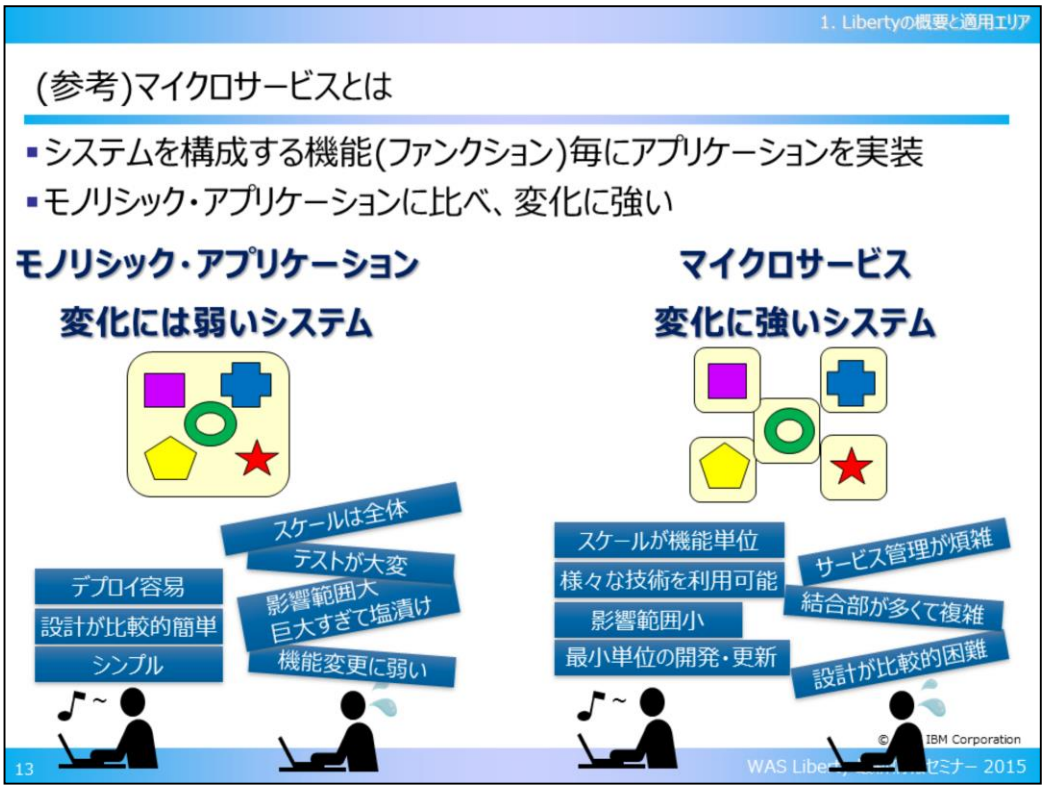
一方、SoEのシステム開発は一般的にどのように行われているでしょうか？こちらも同様に、開発手法、アプリケーション、インフラの3つの観点で特長を挙げます。

開発手法：SoEのシステムはスピードが重要です。必要な機能から実装したり、開発と運用を一体化して行う、アジャイル開発や、DevOpsのアプローチが主に使用されています。

アプリケーション：スピード重視のため、機能毎にアプリケーションを実装し、複数のサービスを組み合わせる、マイクロサービス・アーキテクチャーのアプローチが増えてきています。単機能毎のアプリケーションは、機能変更に強く、テストも最小単位で実施が可能で、SoEのスピード重視にマッチします。

インフラ：サービス提供までのスピードが重要視されるため、すぐに利用可能なクラウド環境も活用されています。インフラ構築を簡単に行う工夫として、インフラ構築をコード化するInfrastructure as codeや、一度作成した環境には手を入れず、常に新しい環境を作成する、Immutable Infrastructureの考え方を採用されるケースも増えています。

こうしたアプローチを行おうとすると、MWにはどのような機能が必要でしょうか？



マイクロサービスは、機能（ファンクション）毎にアプリケーションを実装し、主にRESTのインターフェースを持ちます。マイクロサービスのメリットは、以下挙げられます。

- ・システムの機能という最小単位の開発・更新が可能
- ・機能単位のため、影響範囲が小さい
- ・機能の実装する技術に特に制約がない
- ・機能毎にスケールすることが可能

一方で、デメリットとしては以下が挙げられます。

- ・機能毎に分けるため、設計が比較的困難
- ・複数サービスを組み合わせて使う場合、結合部が多くなり複雑化する
- ・多くのサービスが提供されると管理が煩雑になる

マイクロサービスは、変化に強いシステムを作る場合には非常に有効です。

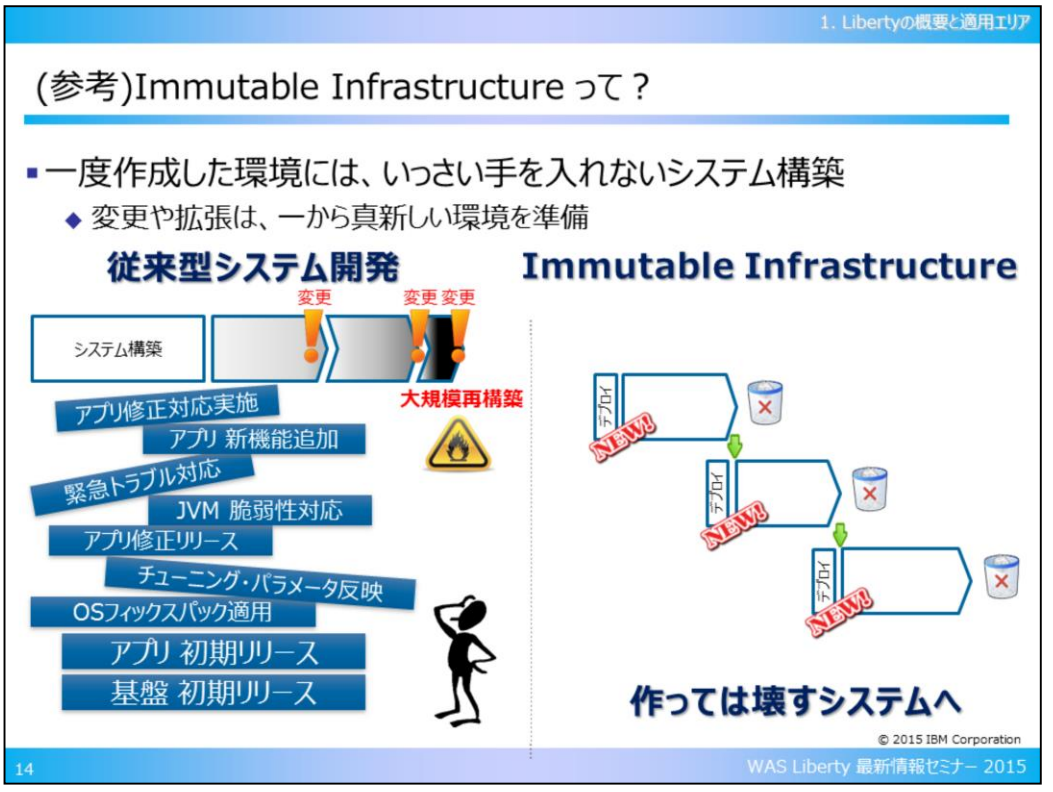
参考として、SoRでよく作られるモノリシックアプリケーションについても、メリット、デメリットを以下に挙げます。モノリシック・アプリケーションは、要件変更がないものに対して有効です。

メリット

- ・1つのアプリケーションで全機能を提供すると言う意味でシンプル
- ・機能分離する必要がない点で比較的簡単
- ・デプロイが容易

デメリット

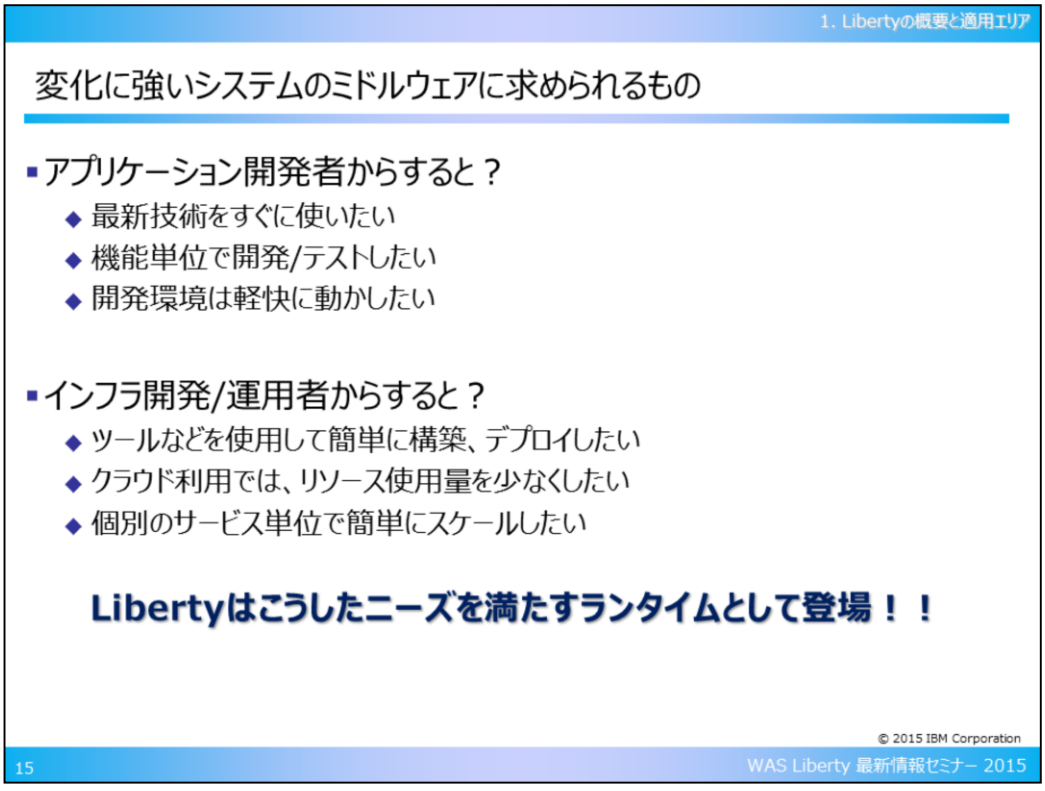
- ・機能変更に弱い
- ・機能変更が行われる場合、影響範囲が大きい→塩漬け
- ・影響範囲が大きくなり、テストが大変
- ・1機能だけのスケールができない



Immutable Infrastructureは、一度作成した環境には一切手を入れないシステム構築です。

これまでのシステム構築では、一度リリースした環境に、OSのFix、チューニングパラメーターの変更、アプリ修正、MWのFixなどを随時行い、その変更に伴いドキュメントも修正を入れていくという運用を行っていました。こうした運用を数年単位で行っていくと、ドキュメントの修正漏れや、担当者依存に陥り、安定稼動に大きな影響をもたらすことがありました。運用の最終局面では、塩漬けになるシステムが多々あります。

このようなことが起こらないようにするため、環境構築を自動化できる仕組みを作り、環境に手を入れる必要が出てきた場合には、再度作り直すのがImmutable Infrastructureです。



SoEに代表される、スピード、変化が求められるシステムのミドルウェアに求められるものはどういうものでしょうか？

アプリケーション開発者からすると、以下のような点が挙げられます。

・最新技術をすぐに使いたい

・機能単位で開発/テストしたい

・開発環境は軽快に動かしたい

SoEアプリケーションは新規性を求められることが多いです。このため最新技術がすぐに使えることが要求されます。SoEアプリケーションは、度重なる変更要求に対しすぐに対応する必要があります。機能単位で開発やテストが容易に出来る必要があります。PC端末のような開発環境でも軽量に動かすことが求められます。

インフラ開発/運用者からすると、以下のような点が挙げられます。

・ツールなどを使用して簡単に構築、デプロイしたい

・クラウド利用では、リソース使用量を少なくしたい

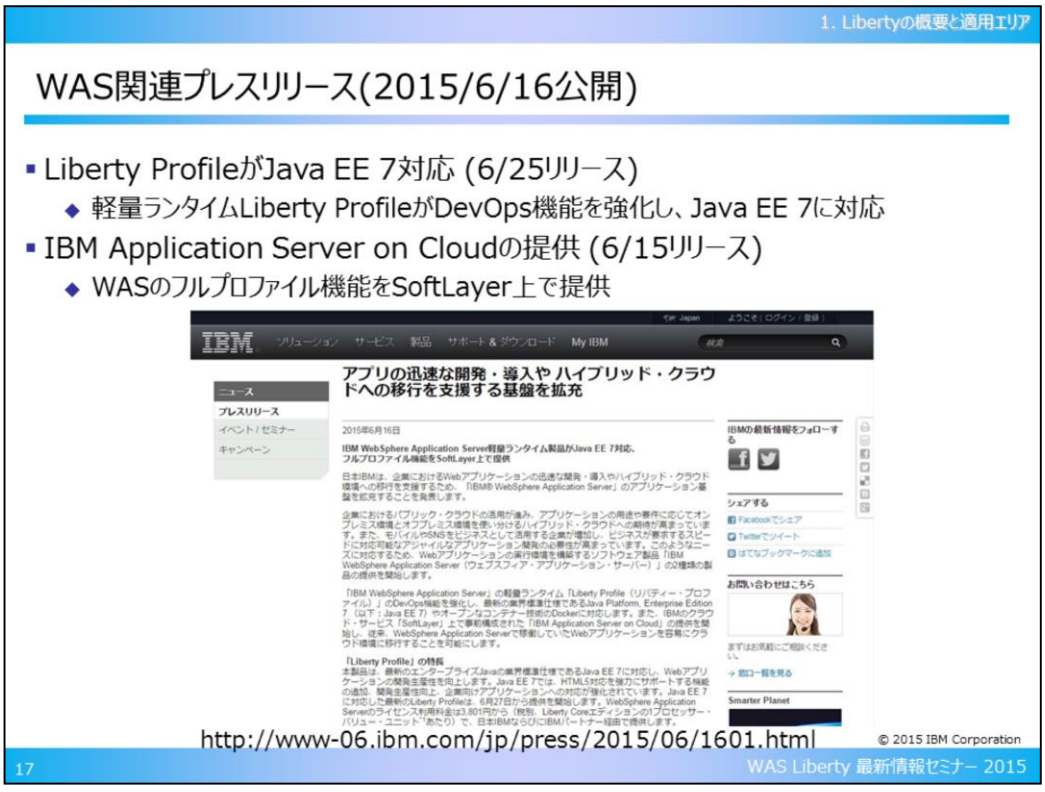
・個別のサービス単位で簡単にスケールしたい

SoEアプリケーションは、スピードが求められるため、インフラとしてクラウドを検討されることが多いです。クラウドで動くことはもちろんのこと、クラウドの課金対象であるリソース使用量を少なくすることが求められます。度重なる変更要求、機能追加などに応じて、環境やアプリケーションの簡単構築、デプロイが求められます。また、サービスがヒットした場合、簡単にスケールできることも求められます。

こういった、ニーズを満たす、Java EE ランタイムとしてLibertyは登場しました。

Libertyの特長

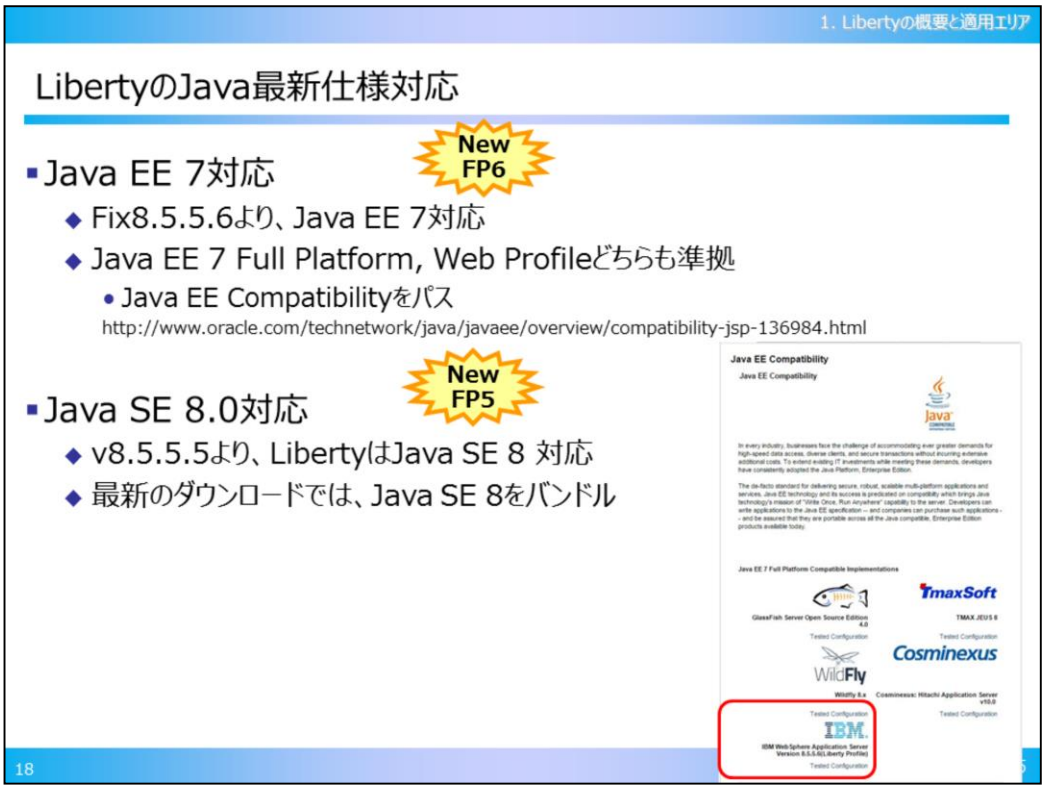
© 2015 IBM Corporation



2015年6月16日にIBMのプレスリリースでWAS関連の発表がありました。

1つはLiberty ProfileがJava EE 7に対応する発表です。Java EE 7対応は、WAS V8.5.5.6のFixPackで提供され、6/25にリリースされています。

もう1つの発表は、SoftLayer上のWASのSaaSサービスの発表です。IBM Application Server on Cloudでは、WASのフルプロファイル機能をSoftLayer上の専用インスタンスで提供します。サーバー、OS、WASの設定が事前に構成されるため、基盤構築の時間を短縮し、迅速にアプリケーションのデプロイが可能です。



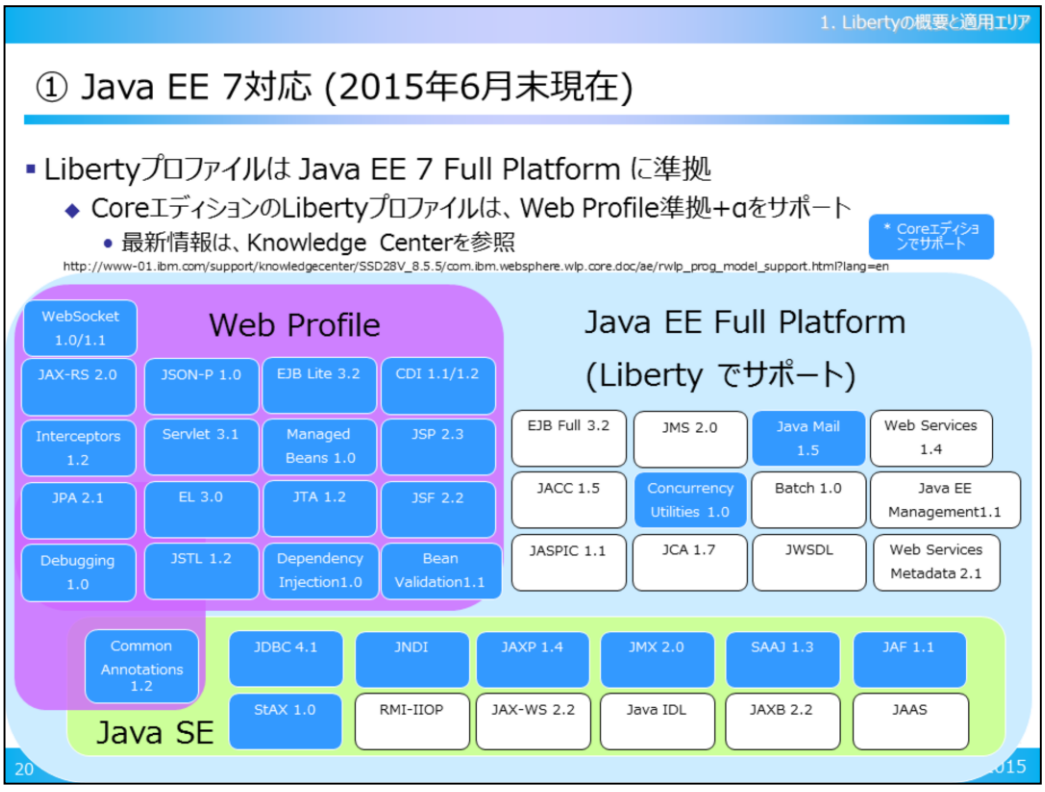
LibertyはFixPack 6では、Java EE 7対応を行いました。Liberty Core エディションでは、Java EE 7のWeb Profileに準拠し、Base/Express/NDのエディションでは、Java EE Full Platformに準拠しています。Oracle のJava EE Compatibilityページにも記載があります。

また、Java SE 8にもFixPack 5で対応しています。最新のLibertyでは、Java SE 8のバンドルもダウンロードが可能です。

Libertyは登場の背景でも述べた通り、SoEのシステムに求められる特長を持っています。
Libertyの代表的な特長は以下になります。

・Java EE 7対応 → ①
・軽量高速、簡単構成 → ②③④
・ツール連携 → ⑤⑥

以降のページでこれらの点について説明します。

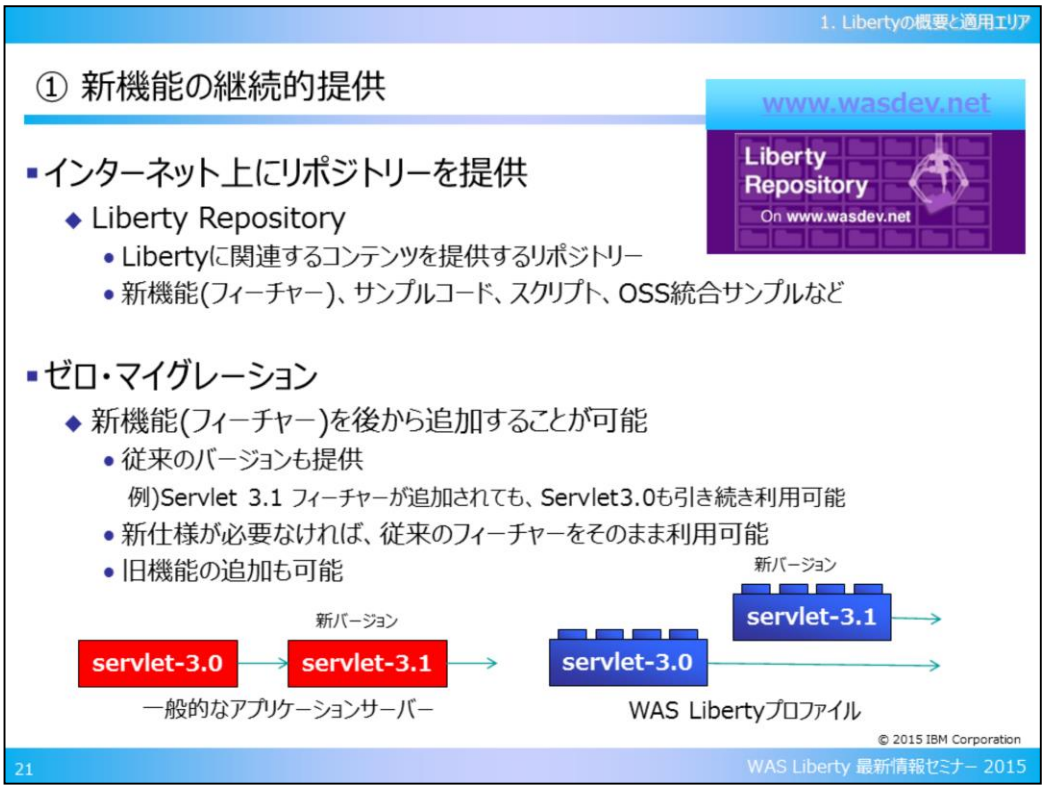


前述の通り、Libertyプロファイルは、Java EE 7 Full Platformに準拠しています。

また、Liberty Core エディションのLiberty プロファイルは、Web Profileの仕様に準拠しており、Web Profile仕様の他に追加でいくつかAPIをサポートしています。

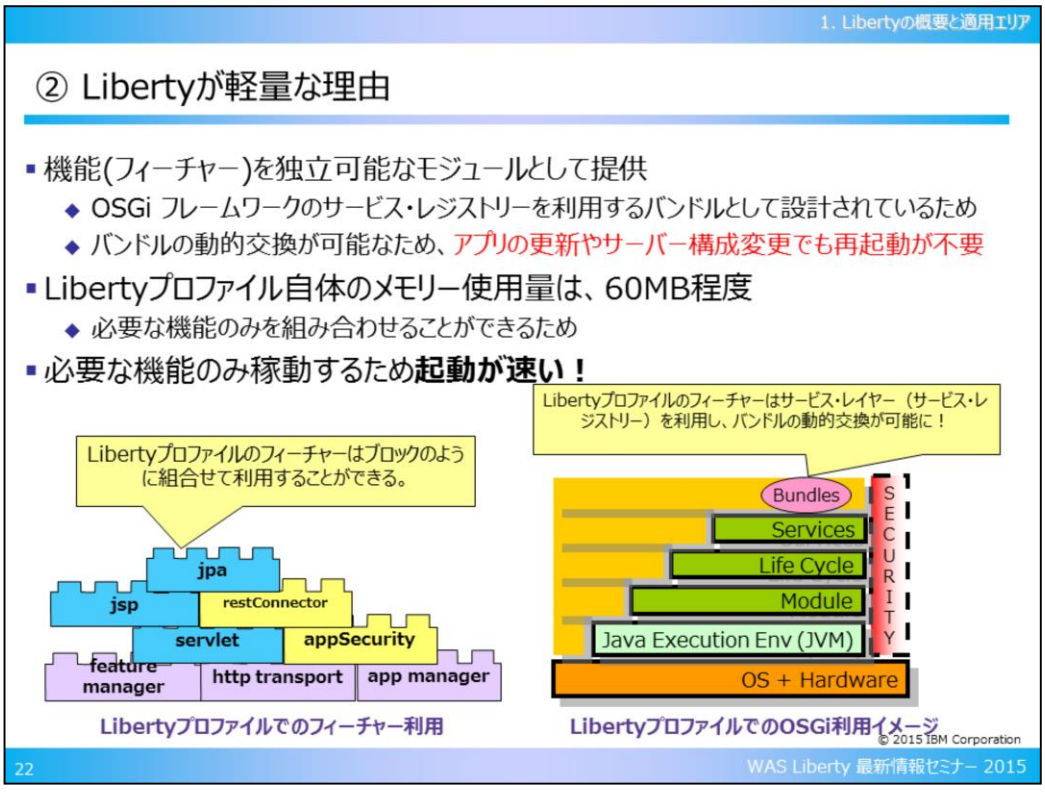
サポートしている仕様は追加される可能性があります。最新情報に関しては、Knowledge Centerを確認ください。

http://www-01.ibm.com/support/knowledgecenter/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/rwlp_prog_model_support.html?lang=en



Libertyは、インターネット上にLiberty Repositoryというリポジトリーを提供しています。そこで、Libertyに関連するフィーチャー、サンプルコードなどを提供しています。ここから新しいフィーチャーをダウンロード可能です。Liberty Repositoryについては、「3. Libertyプロファイルの基本構成」で詳しく説明します。

Libertyは、ランタイムを置き換えるのではなく、追加するというゼロ・マイグレーションという概念を持っています。必要に応じて新機能等を追加し、必要なければ従来の機能を使い続けることができます。



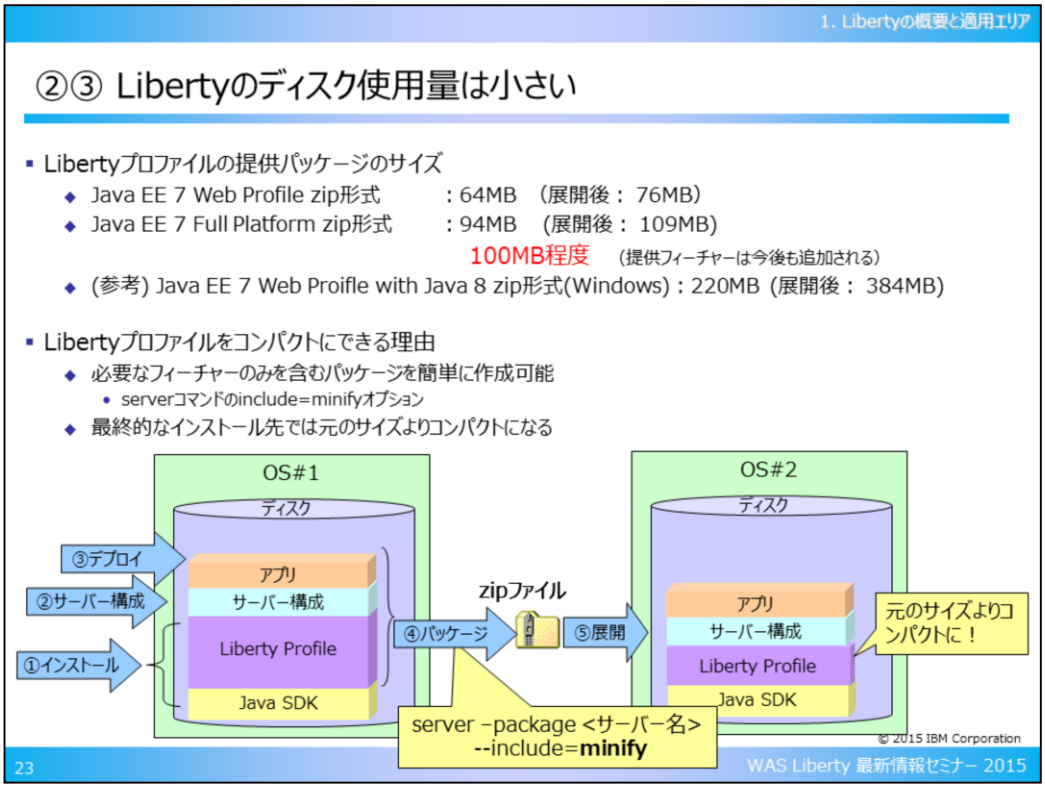
Libertyが軽量な理由は、必要な機能単位（「フィーチャー」と呼びます）だけを柔軟に組み合わせて利用しているからです。

左の図がLibertyプロファイルでのフィーチャー利用のイメージ図です。Libertyのフィーチャーをブロックに喩えて描いています。Libertyでは、まるでブロックを積み上げるがごとく、必要なフィーチャーだけを組み合わせて利用できます。

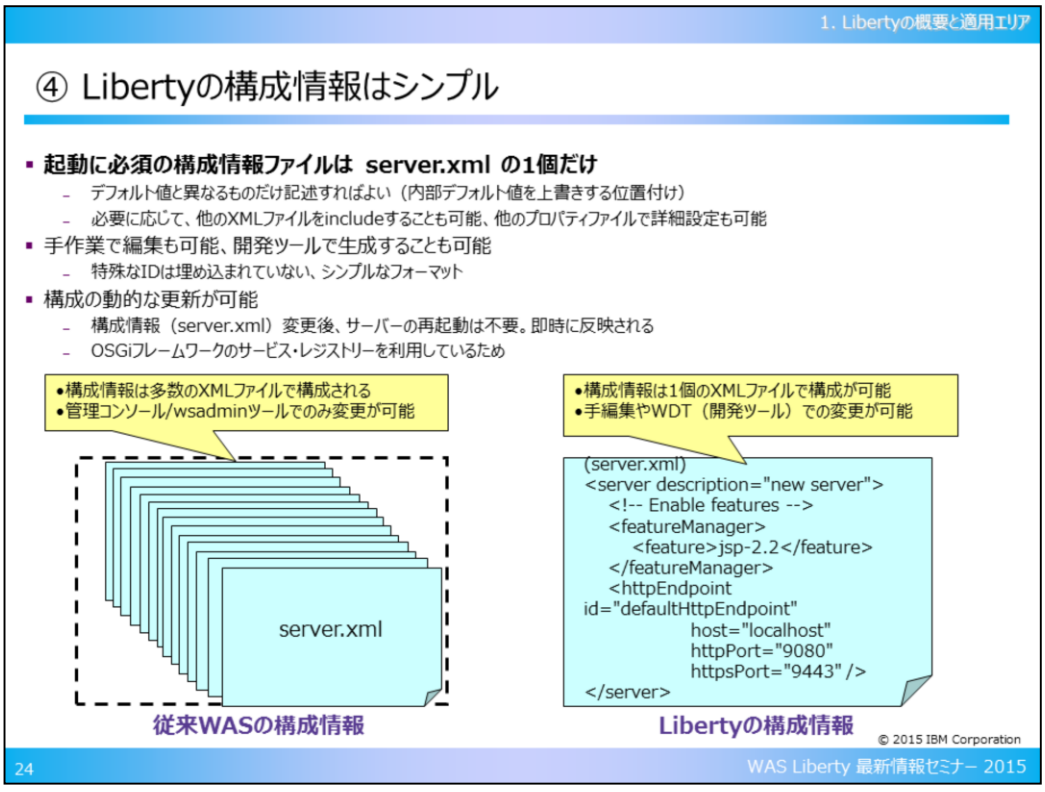
このようにフィーチャーを組み合わせて利用できるのは、LibertyがOSGiフレームワークを利用しているからです。中でも特に、OSGiのサービス・レイヤー（サービス・レジストリー）を利用していることによる効果が大きいのです。OSGiの仕組みは主に３つの要素から構成されています。

右の図にある様に、下から①Moduleレイヤー、②ライフサイクル・レイヤー、③サービス・レイヤーとなっています。Moduleレイヤーは、バンドル（OSGiのモジュールを「バンドル」と呼びます）のバージョン管理、依存関係解決の仕組みを提供します。ライフサイクル・レイヤーは、バンドルの一生における状態（インストール済み→解決済み→開始中→活動中→停止中→解決済み→アンインストール済み）を管理する仕組みを提供します。サービス・レイヤーは、バンドルに含まれるサービス（クラスやインターフェースの単位）を他のバンドルも利用可能なようにするための仕組みを提供します。

従来のWASもOSGiのモジュール・レイヤーやライフサイクル・レイヤーは利用していましたが、サービス・レイヤーは利用していませんでした。Libertyはサービス・レイヤーも利用するため、バンドルの動的交換が可能になりました。このため、アプリケーションの更新やサーバー構成の変更の場合でも、サーバーを再起動する必要がなくなりました。



Libertyプロファイルが提供しているパッケージは、従来のWASと比べると非常に小さくなっています。提供されるフィーチャーは今後も追加されていく予定ですが、それでもLibertyのディスク使用量は小さく留めることが可能です。それは、Libertyの提供するserverコマンドでは、特定のサーバー構成で利用されているフィーチャーのみをパッケージするオプションが用意されているからです。そのオプションは、--include=minify　です。minifyは「小さくする」という意味です。



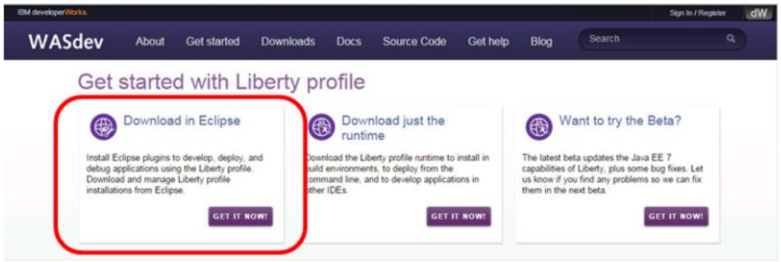
Libertyの構成情報は、最低限1個のXMLファイルで記述することが可能です。構成情報は、server.xmlという名前のファイルで、Libertyが内部に保持しているデフォルト値を上書きしたいものだけを記述すればよい仕組みになっています。右側に例として書かれているserver.xmlファイルは、わずか10行程度のものですが、これはデフォルトで生成されるserver.xmlファイルで、これだけでサーバーを起動することが可能です。もちろん、他のXMLファイルをインクルードしたり、詳細なオプションをプロパティー・ファイル（server.env, bootstrap.properties, jvm.options）で指定することも可能です。

また、Libertyのserver.xmlファイルは、普通のエディターを利用して手作業で編集することもできますし、開発ツール（WDT）で編集することもできます。従来のWASでは、構成情報ファイルは40個以上のXMLファイルで構成され、手作業で編集することはできませんでした。これは、XMLファイルの中にWASが自動で生成する特殊なIDが埋め込まれていたからです。Libertyのserver.xmlファイルには、そのような特殊なIDは使われていません。

また、Libertyの構成情報ファイルの変更後は、その変更が即座にサーバー環境に反映されます。サーバーの再起動は不要です。

⑤ WebSphere Developer Tools(WDT)の提供

- WebSphere Developer Tools for Eclipseとは？
  - 無償で提供される WebSphere開発者向けの軽量Eclipseプラグイン
    - Java EEアプリケーション開発、アセンブル、デプロイ
    - Libertyプロファイルのサーバー構成に利用
  - wasdev.netサイトからダウンロード可能
    - https://developer.ibm.com/wasdev/downloads/

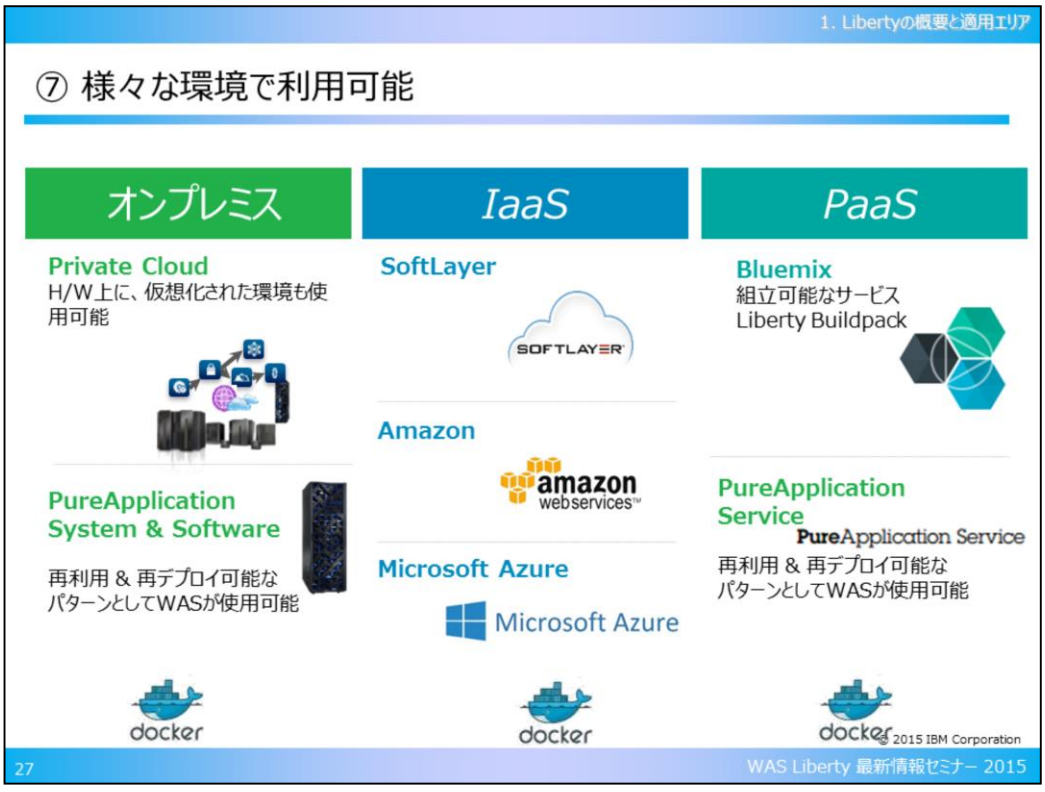


© 2015 IBM Corporation

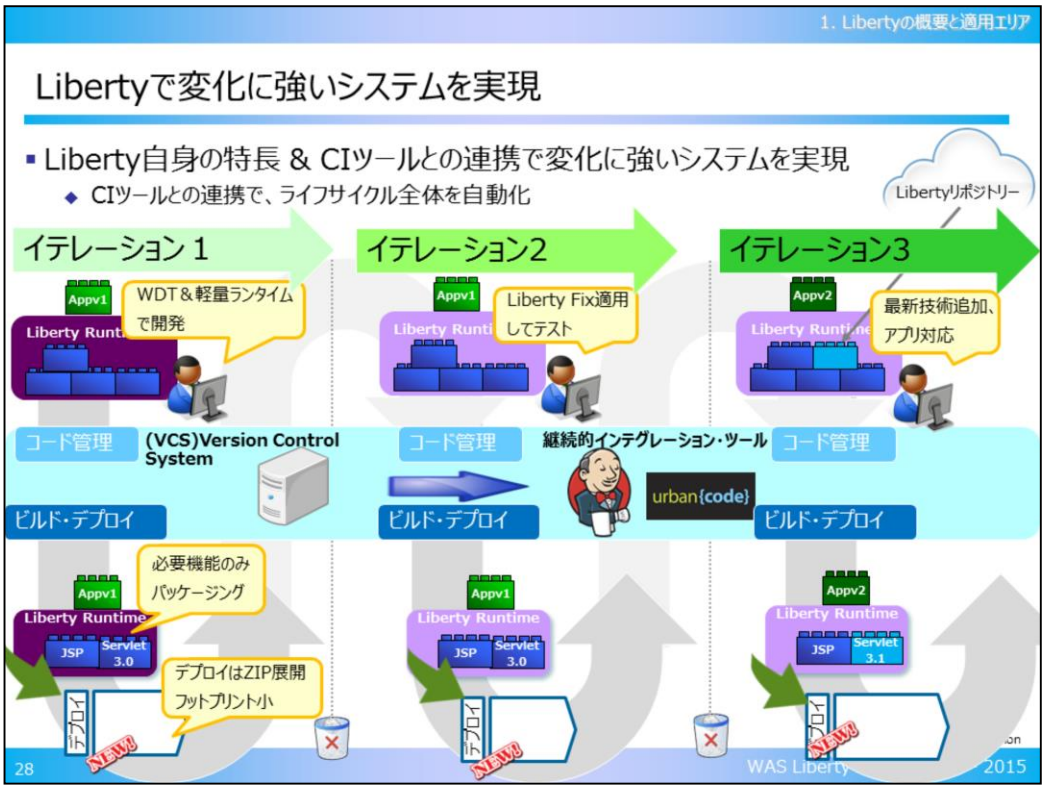
WebSphere Developer Toolsは、無償で提供されるWebSphere開発者向けの軽量Eclipseプラグインです。WebSphere Application Serverに向けて、Java EEの開発やアセンブルをヘルプする機能が提供されます。さらにLibertyプロファイルのサーバー構成をサポートする機能が豊富に含まれています。

WDTは、wasdev.netサイトからダウンロードすることが可能です。後続のセッションでもご紹介します。

Libertyは、各種自動化ツールと連携するプラグインを提供しています。
IBMの継続的デリバリをサポートするツールであるUrbanCodeと連携するプラグインを提供しています。
OSSとの連携機能は、それぞれのOSSツールに対して、GitHub上でプラグインを公開しています。
Libertyと連携して仕様することで、システム開発のライフサイクルを自動化することが可能です。



最後の特長として、Libertyは様々な環境で動作することができます。
Libertyは、オンプレミス、オフプレミス問わず様々な環境で動作させることができます。
前提OSなどは、以下サイトを参照ください。
http://www-01.ibm.com/support/docview.wss?uid=swg27006921



Libertyは、これまでご紹介した特長を活用し、スピードや変化が求められるシステムで上記図のように活用できます。

アプリケーションの開発では、EclipseのプラグインとしてWDTを使用してアプリケーションを容易に開発することができますし、サーバー構成についても設定することができます。CIツールと連携することで、ビルド、デプロイを自動化することも可能です。本番環境にデプロイするときには、Libertyでは必要機能のみZIPパッケージングすることが可能で、インストールも容易でかつフットプリントを小さくすることができます。

SoEのアプリケーションは特性上、公開されたアプリケーションであることが多いです。このため、脆弱性などにすぐに対応する必要があります。こうしたニーズにも、開発環境にすぐにFixをあててFixをあてた環境での開発・テストをすることが可能です。Fixと同様に、最新技術に対してもすぐに対応する必要があります。Libertyでは、Libertyリポジトリーから最新技術をダウンロードし利用することで、最新技術に対応することができます。

このようにスピードが求められるシステム開発において、アプリケーション、インフラの両面で役に立つ特長を有しています。

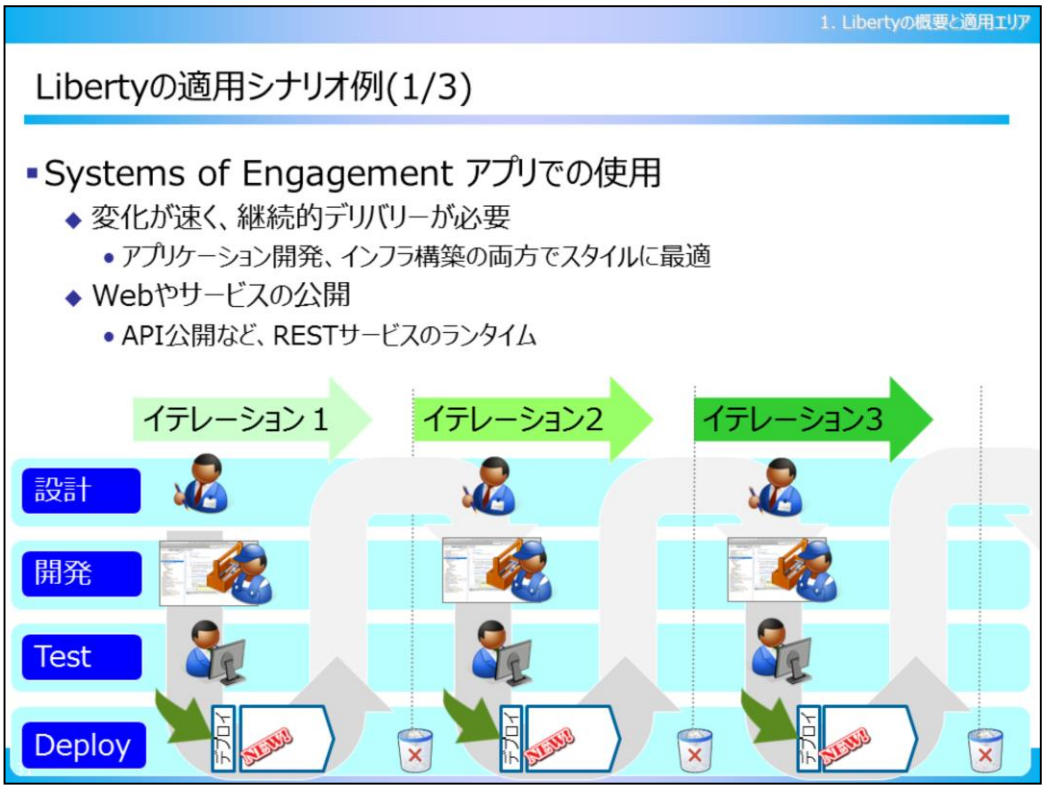
Libertyの適用エリア

© 2015 IBM Corporation

最初にLibertyが使用されている事例をご紹介します。

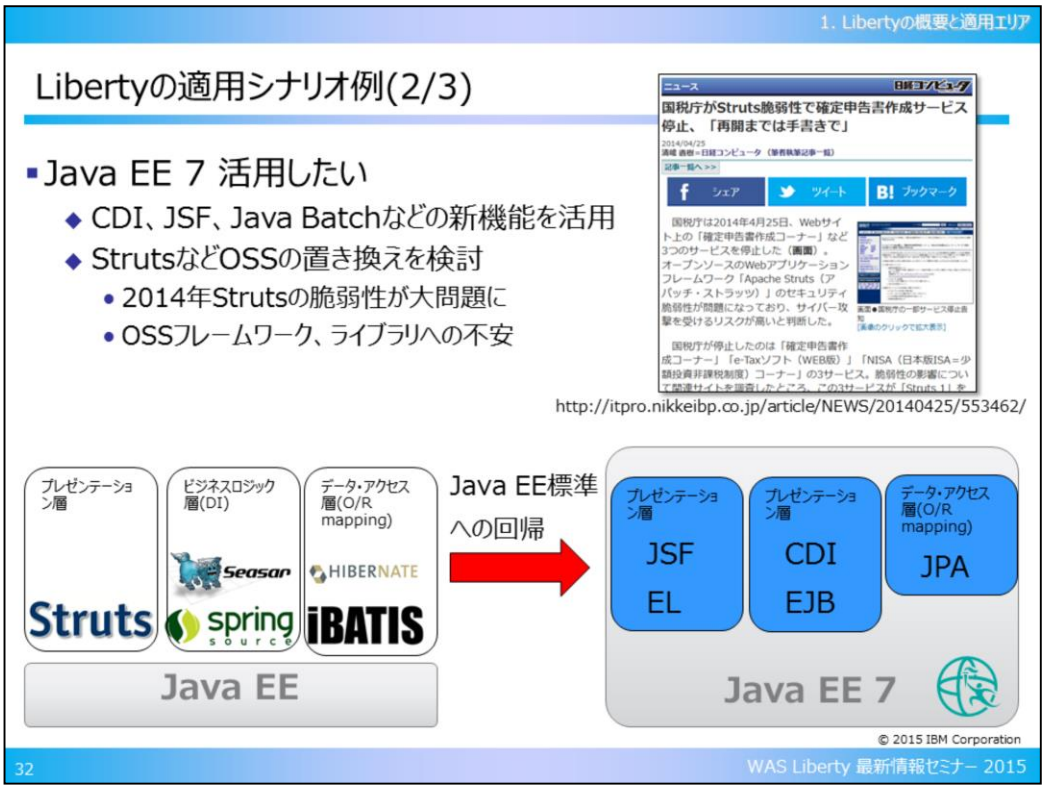
WAS フル・プロファイルはこれまで、多くのIBM製品に組み込まれていました。Libertyも同様に、IBM製品に組み込まれ始めています。代表的なものとしては、Watson Analyticsのクエリー処理のためや、モバイルアプリケーションの開発実行基盤であるMobile First Platformのサーバー側のコンポーネントとして使われています。

実際のお客様としては、Grand Slam TennisのWebサイト、モバイルからのアクセス先としてLibertyが利用されています。



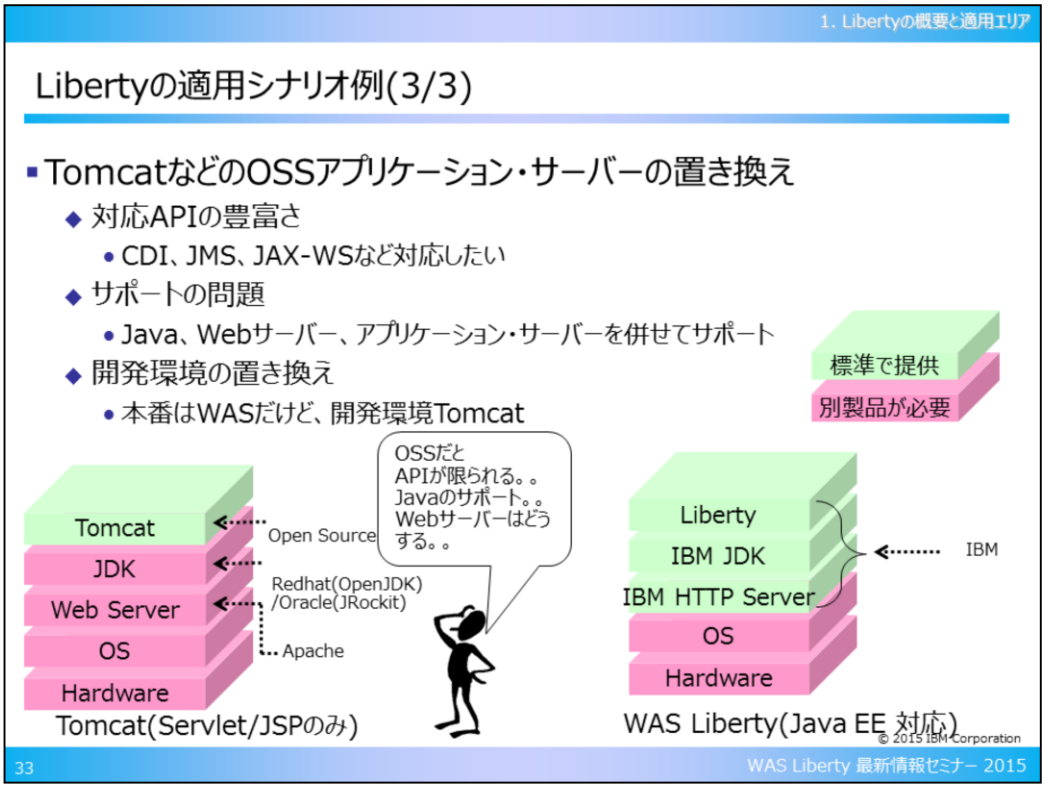
ここからLibertyの適用シナリオをご紹介します。

1つ目は、登場の背景でも述べたとおり、LibertyはSoEやIoTに代表されるようなスピードが求められるシステム構築に適しています。継続的デリバリーが必要なアプリケーションやそのインフラとしての利用が理想的です。モバイルアプリケーションなどのアクセス用APIや、SoRのサービス公開用のゲートウェイのランタイムとしても理想的です。



適用シナリオの2つ目は、Java EE 7 を利用するというシナリオです。

「7. Liberty プロファイルJava EE 7対応　～API編～」のセッションでもご紹介しますが、2000年台前半に多くの企業でフレームワークを利用したアプリケーション開発が行われました。それらシステムの更改を検討されている企業が多くあります。当時のJava EEの仕様は不十分で、多くのOSSフレームワークやライブラリが利用されました。現在のJava EE 7は仕様として充実しており、OSSからの回帰が起こっています。特に、2014年に起こったStrutsの脆弱性を元にOSSフレームワーク、ライブラリへ不安を感じられている方も多いと思います。標準仕様であるJava EE 7はこういった不安を解消することができ、最新の技術を利用することができます。是非Java EE標準を使用したアプリケーションに更改したいというシステムでご利用ください。



適用シナリオの3つ目は、OSSアプリケーション・サーバーの置き換えです。

OSSアプリケーション・サーバーは、Java EEの仕様のごく一部しか提供していません。例えばTomcatであれば、Servlet/JSPのみです。それ以上のことをやりたい場合は、必要なOSSのライブラリを探し、それらを組み合わせ、テストをして使う必要があります。Libertyは、Java EE 7に対応しているため、これらの苦労をすることなくJava EEが提供している全ての機能を利用可能です。

OSSでは、JDKやWebサーバーに何を使用するか、サポートをどうするか検討する必要があります。Libertyでは、JDK、Webサーバーも同梱しており、これらのサポートもあわせて提供しています。JDKやWebサーバー、アプリケーションサーバーに対して不安を感じられている場合、是非Libertyをご検討ください。

本番環境はWASのフル・プロファイルを利用しているのに、開発環境では軽量さを理由にTomcatを利用しており、Tomcat環境では動くのに、WAS上では動かないといったことを伺うことがあります。Libertyは軽量ですので、こうした開発環境のTomcatをLibertyに置き換えることが可能です。Libertyは、WASのフル・プロファイルとコードを共有しているため、WASのフル・プロファイルと親和性が高いです。開発環境のLibertyは無償でダウンロードすることができるため、本番はWASで開発環境はTomcatを使用してる場合がありましたら、Libertyを積極的にご利用ください。

まとめ

- LibertyはWAS V8.5から提供されている軽量ランタイム
- Libertyの特長
  - Java EE 7対応
  - 軽量、高速
  - ツールとの連携も容易
- SoE、IoTに代表されるスピードや変化を求められるアプリケーションのランタイムとして最適
  - SoEのアプリケーション
  - Java EE 7アプリケーション
  - OSSアプリケーション・サーバーの置き換え

© 2015 IBM Corporation

この章では、LibertyはWAS V8.5からリリースされた軽量ランタイムであることをご紹介しました。

Libertyの特長として、Java EE 7への対応、新機能を継続的に提供すること、軽量、高速、簡単構成が可能、ツールとの連携も可能であることをお特長としてご紹介しました。

Libertyの用途としては、SoE、IoTのエリアのアプリケーションで使うのが、アプリ、インフラ双方で最適です。

1. Libertyの概要と適用エリア

参考情報

- アプリの迅速な開発・導入や ハイブリッド・クラウドへの移行を支援する基盤を拡充
  - http://www-06.ibm.com/jp/press/2015/06/1601.html
- dW: IBM WebSphere Appication Server Liberty プロファイル
  - http://ibm.biz/LibertyJPN
- Knowledge Center
  - http://www-01.ibm.com/support/knowledgecenter/SSD28V/welcome

© 2015 IBM Corporation

1. Libertyの概要と適用エリア

(参考)SOAとマイクロサービス、何が違うのか？

	SOA	マイクロサービス
連携に使用される主な技術	SOAPによるWebサービス	RESTfulなWebサービス
サービスに分割する単位	企業の組織単位（Conwayの法則）	要求される機能単位
サービスの粒度	大きい	なるべく細かく
サービスで提供される機能	比較的高機能（トランザクション・順序性の保護・セキュリティなど）	比較的低機能
各サービスを実装する技術	Javaや.NETなどに統一されることが多い	サービスごとに最適な言語・技術を利用
サービス連携のトポロジー	ESBを導入しハブ型の構造を取ることが多い	メッシュ型の直接呼び出しになることが多い
サービスの主な利用範囲	社内のシステム間連携	社外を含めた連携

© 2015 IBM Corporation

マイクロサービスは、SOAとよく比較されます。ここでは簡単にまとめています。

1. Libertyの概要と適用エリア
THANK
YOU
© 2015 IBM Corporation
37
WAS Liberty 最新情報セミナー 2015