WAS Liberty 最新情報セミナー 2015

IBM

～ 次世代ランタイム Libertyで変わる Java EE システム～

7. Libertyプロファイル Java EE 7対応 ～API編～



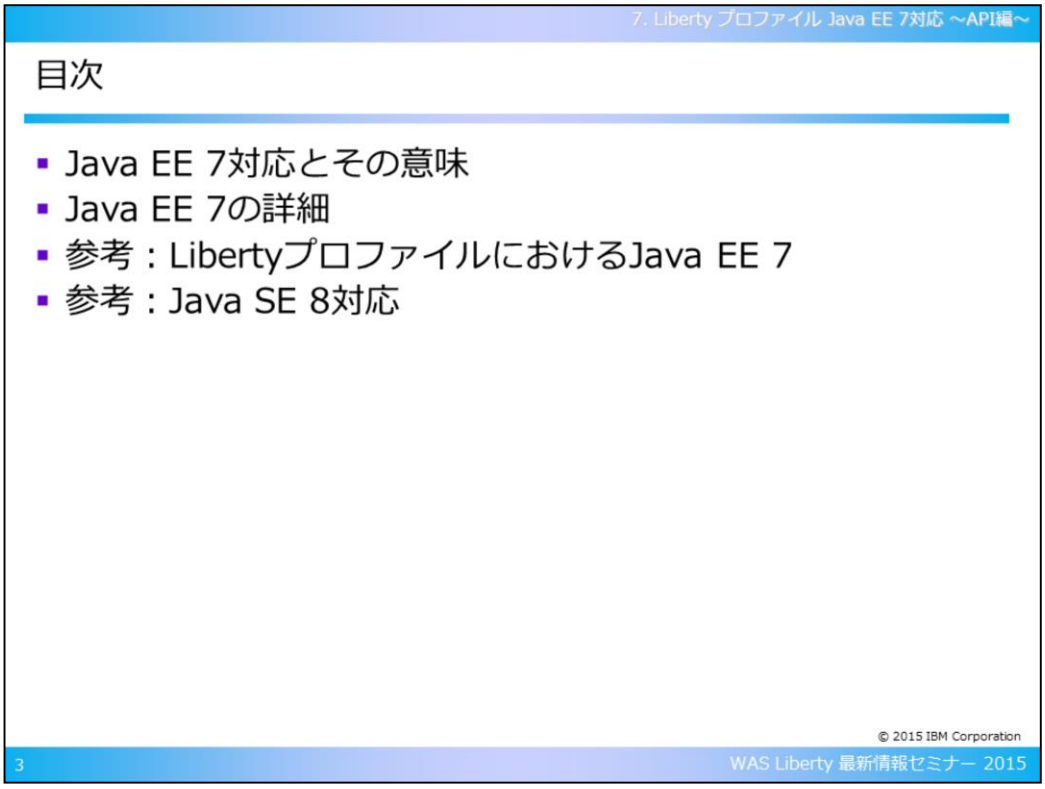
© 2015 IBM Corporation

1

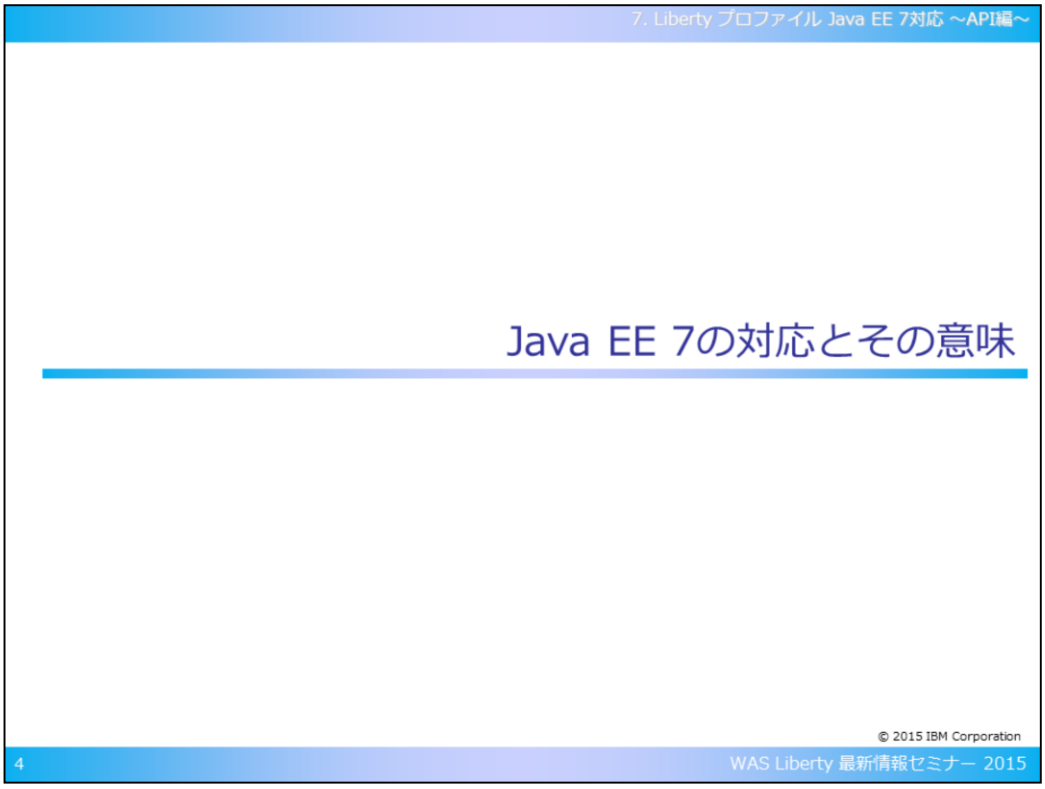
Disclaimer

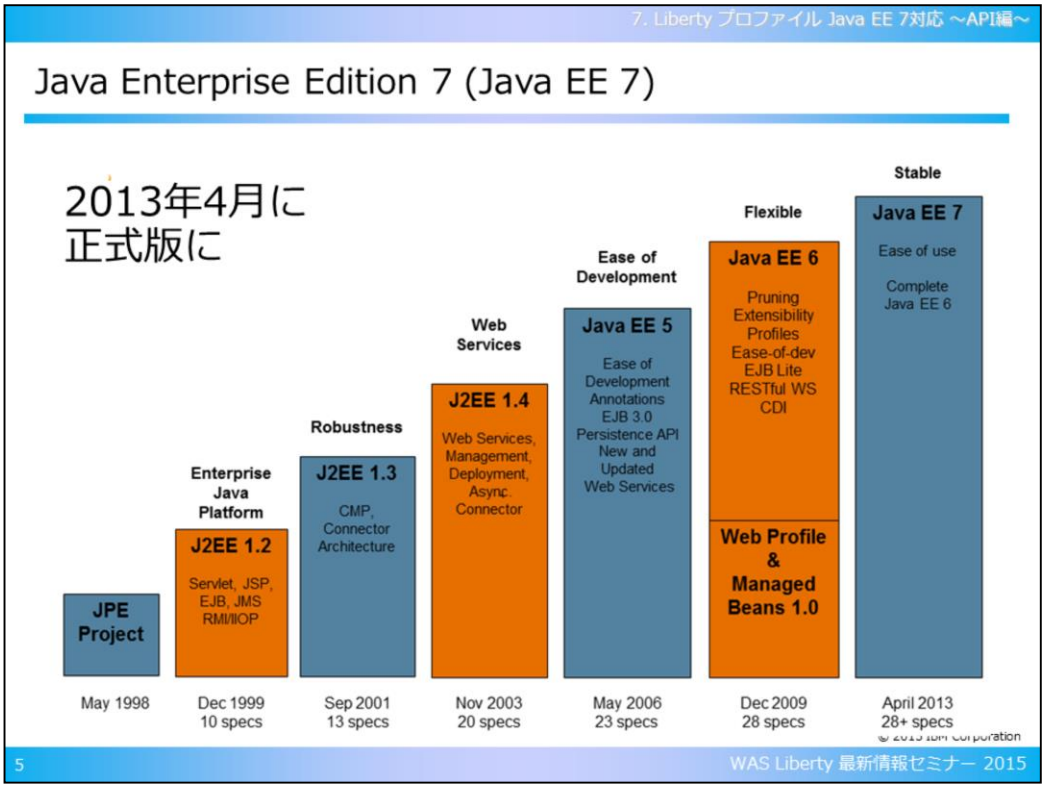
- この資料は日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム　システムズ・エンジニアリング株式会社の正式なレビューを受けておりません。
- 当資料は、資料内で説明されている製品の仕様を保証するものではありません。
- 資料の内容には正確を期するよう注意しておりますが、この資料の内容は2015年7月現在の情報であり、製品の新しいリリース、PTFなどによって動作、仕様が変わる可能性があるのでご注意下さい。
- 今後国内で提供されるリリース情報は、対応する発表レターなどでご確認ください。
- ＩＢＭ、ＩＢＭロゴおよびibm.comは、世界の多くの国で登録されたInternational Business 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

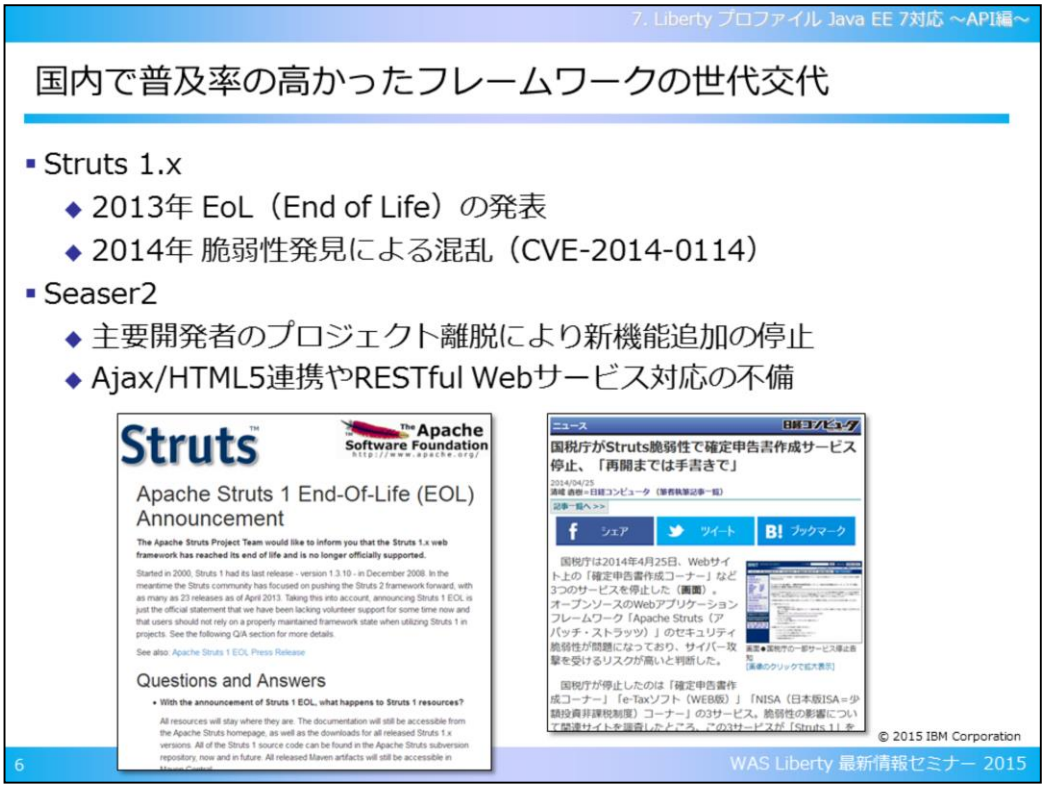


このセッションではLibertyプロファイルが対応しているJava EE 7仕様で提供されているAPIを中心にお伝えします。


7. Liberty プロファイル Java EE 7対応 ～API編～
Java EE 7の対応とその意味
© 2015 IBM Corporation
4
WAS Liberty 最新情報セミナー 2015



Java EEは，JCPというJavaのコミュニティプロセスで策定されている業界標準の仕様群です。1999年に登場したJ2EE 1．2以降，その時代の最新の技術動向を取り込みながら進化を続けてきました。現在の最新のJava EE仕様は，2013年に正式公開となったJava EE 7です。

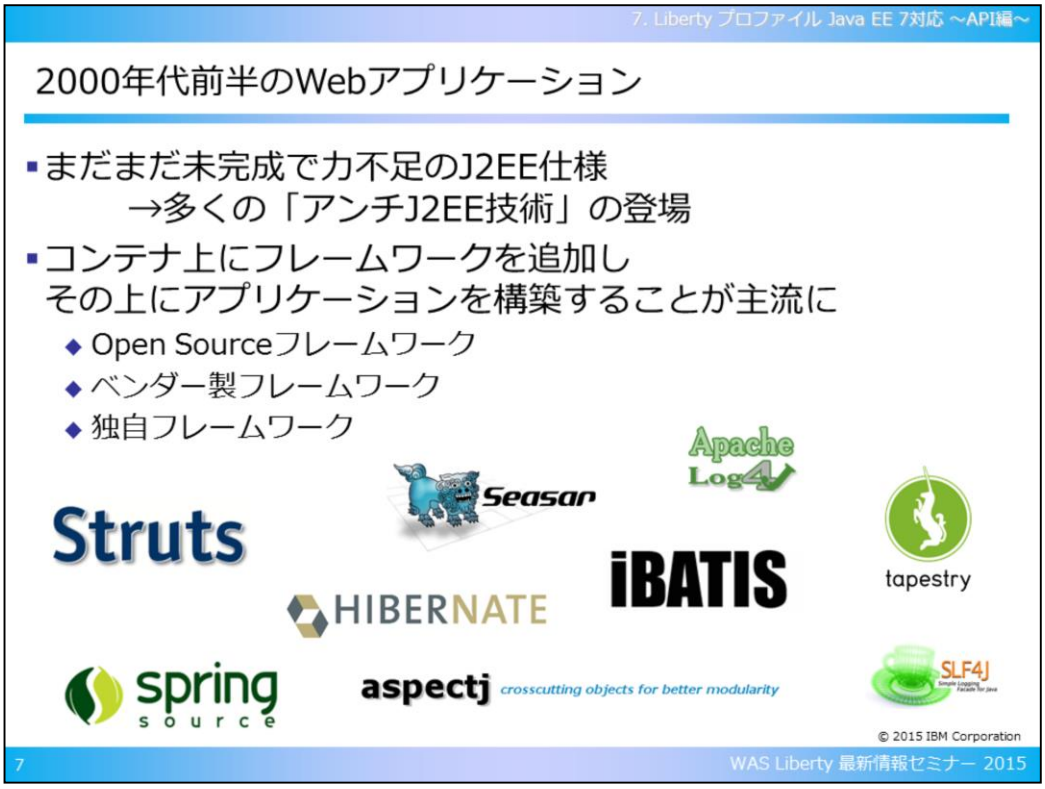


エンタープライスJavaをとりまく昨今の状況を説明します。

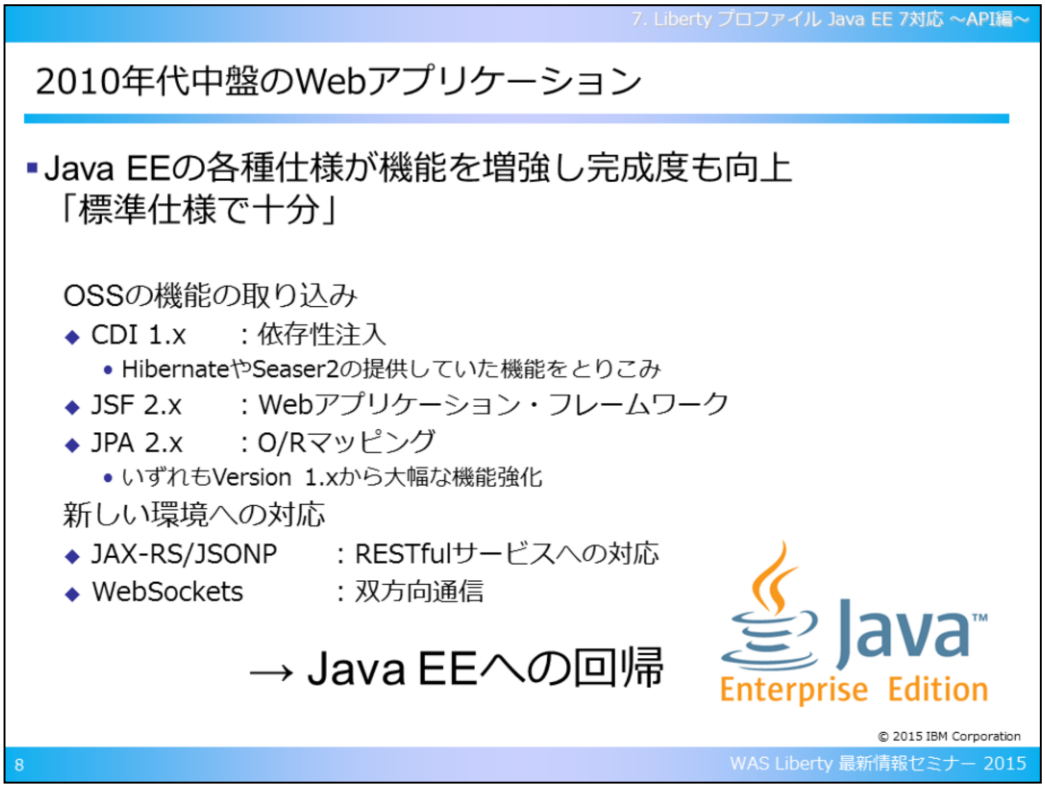
2000年代から，国内外の多くの企業でJavaによるシステム開発が行われました。その際にはStrutsや日本産のOSSであるSeaser2などのフレームワークを使用したアプリケーション開発がよく行われていました。これらのフレームワークの中には，ソフトウェアの寿命を迎えてセキュリティ上の問題が発生したり新機能の実装が行われなくなっているものがあります。また今日でも開発が続けられているものでも，互換性のない大きなバージョンアップを実施しているものがあります。

これらの状況から，多くの企業で「次世代のJavaアプリケーションのアーキテクチャー」の検討が行われています。

添付画像は以下のサイトより引用
https://struts.apache.org/struts1eol-announcement.html
http://itpro.nikkeibp.co.jp/article/NEWS/20140425/553462/



Javaによる企業システムが多くの企業で実施された2000年代前半。当時のJ2EE仕様はServlet/JSPなどの基礎技術が提供されているだけであり，それだけで企業システムを構築するには機能が不足していました。そのためコンテナが提供するJ2EE基本機能の上に「フレームワーク」を追加することが一般的に行われていました。国内でも多くのベンダーがJavaアプリケーション用のフレームワークを販売していました。またそれぞれの企業ごとに全社標準の「標準フレームワーク」を策定することもよく行われていました。



Java EE仕様は，過去の進化の中で，そのときそのときの主流のフレームワークで使用されている技術を貪欲に取り込んできました。また批判の多かった冗長なコーディングについても順次改善をおこない，開発生産性の高い標準仕様と変わってきています。

現在の最新仕様であるJava EE 7では，OSSフレームワークで提供されてきた多くの機能を標準技術として提供しています。また最新の技術動向も積極的に取り込んでいます。

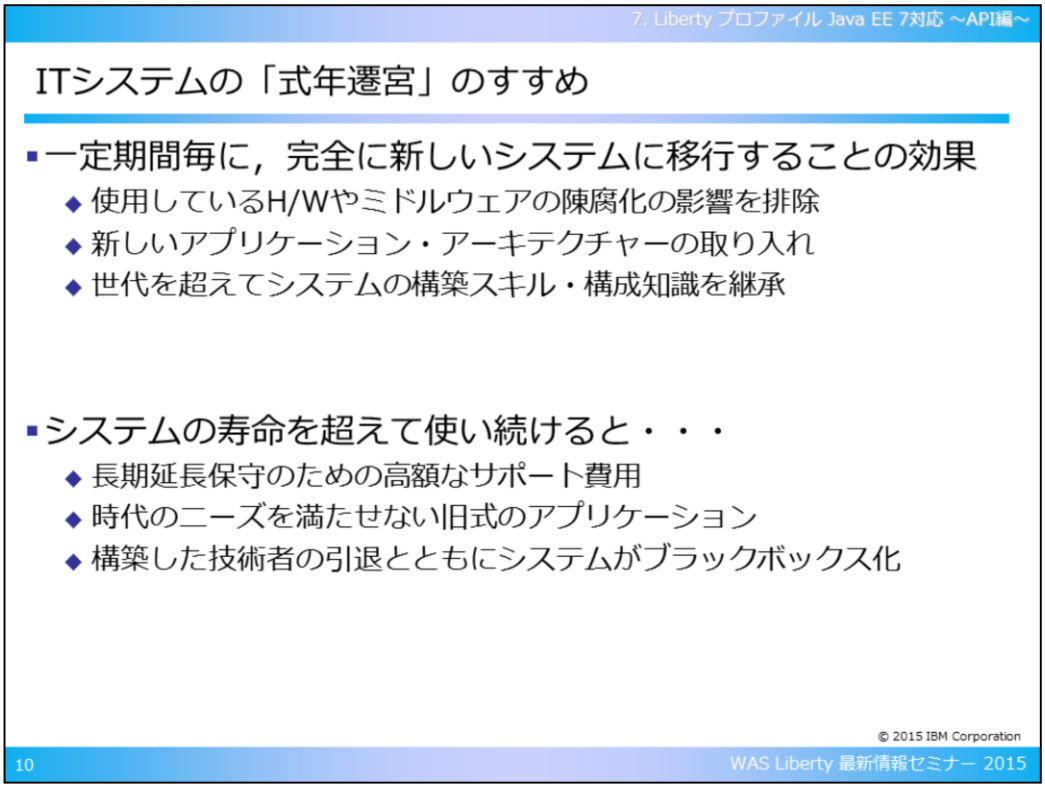
そのため，アプリケーションの基本フレームワークを別途追加する必要は年々小さくなってきています。もちろん機能ごとの外部OSSライブラリを使用することは行われていますが，基本はJava EEを使用するという「Java EEへの回帰」が鮮明になってきています。



企業システムを改良・改善しようとする場合，大きく二つの流儀があるでしょう。

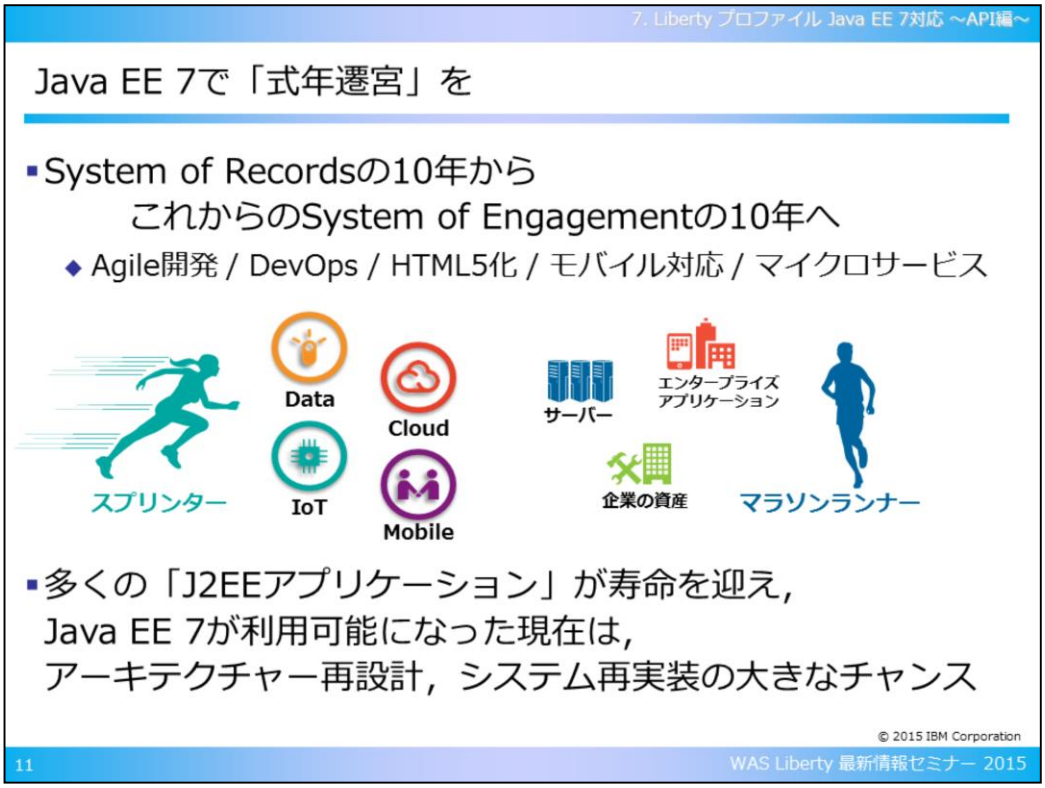
一つは基礎から再設計・再実装をおこなって全く新しいシステムを作り直すこと。もう一つは既存のシステムをベースに拡張をおこなうこと。

もちろん改良を行うたびに毎回システムを再構築するわけにはいきません。既存の投資を生かしビジネスを連続して提供するためにも，既存のシステムの活用は不可欠です。しかし，それも度を超すと弊害が大きくなります。寿命を越えて継ぎ足しを繰り返したシステムは，各所で整合性がとれず複雑性が増し，多くの不利益をもたらす負債となってしまうことがあります。



企業のシステムにおいても，技術パラダイムが大きく変化する10年程度を目安に，単なる機能追加や改良にとどまらない完全なシステムの再作成，いわば「ITシステムの式年遷宮」が必要なのではないでしょうか。

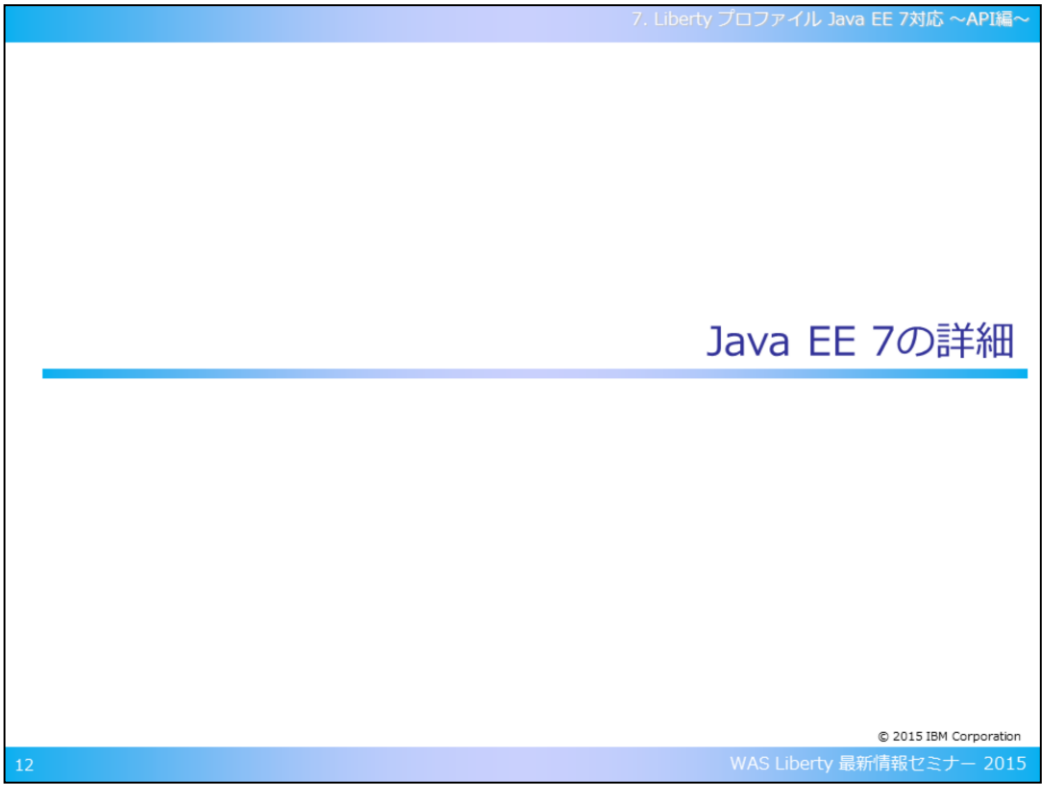
「式年遷宮」には，多くの企業がおちいりがちなシステムの負の側面を解消する効果があります。

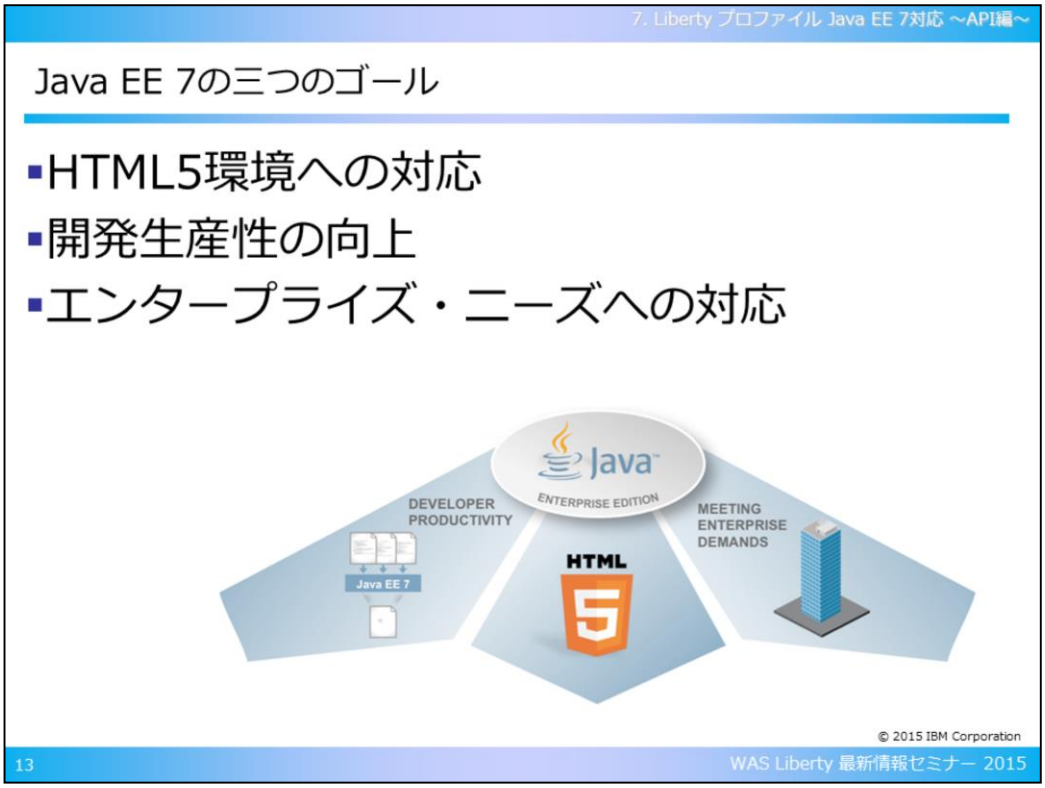


多くの企業で，Javaによるシステム開発がはじまってから10年前後が経過しました。当時からJavaによるアプリケーションを継続して利用しているお客様では，一部に老朽化したコンポーネントやアーキテクチャーがみられるようになってきたかと思います。またシステムをとりまく外部の状況も変化してきました。モバイル対応やSNS対応，IoTへのとりくみなどで，非常に早い変化の対応が求められるようになってきています。実行環境もオンプレミスからクラウドへひろがってきています。

システムの見直しは，単なるアプリケーションの機能にとどまるものではありません。それをどのように開発し，テスト・運用していくかというITシステムの全般を改善していく過程です。Agile開発やDevOpsなど，多くの思想がひろまってきていますが，既存のアプリケーションをそのまま持って移行しようとしても，なかなかうまくいかないケースが多いでしょう。

Java EE 7という仕様が登場し，それに対応した商用のアプリケーションサーバーが登場した2015年という現在は，あたらしいJavaのシステムをつくりはじめる最適な時期であると言えます。


7. Liberty プロファイル Java EE 7対応 ～API編～
Java EE 7の詳細
© 2015 IBM Corporation
12
WAS Liberty 最新情報セミナー 2015



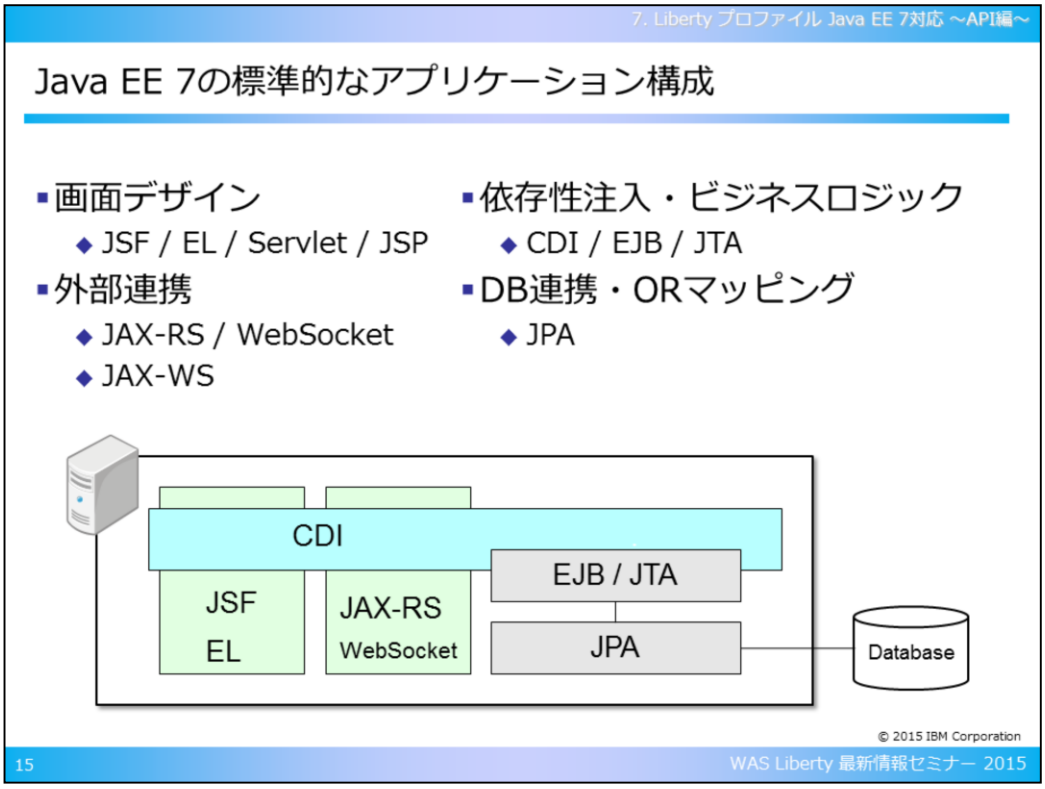
Java EE 7は，仕様策定にあたって三つのゴールが設定されていました。



Java EE 7に含まれる仕様群をこれらの三つのゴールにマップするとこのようになります。

このうち赤字の仕様が今回のJava EE 7で新規に追加されたものです。ここに記載されている他の黒字の仕様についても全てバージョンアップしています。JAX-RSやJMSなどはメジャーバージョンアップして大きな機能改善が行われています。

またJava EE 7仕様の正式公開後に個別にバージョンアップした仕様（WebScoket 1.0→1.1，CDI 1.1→1.2）もあります。

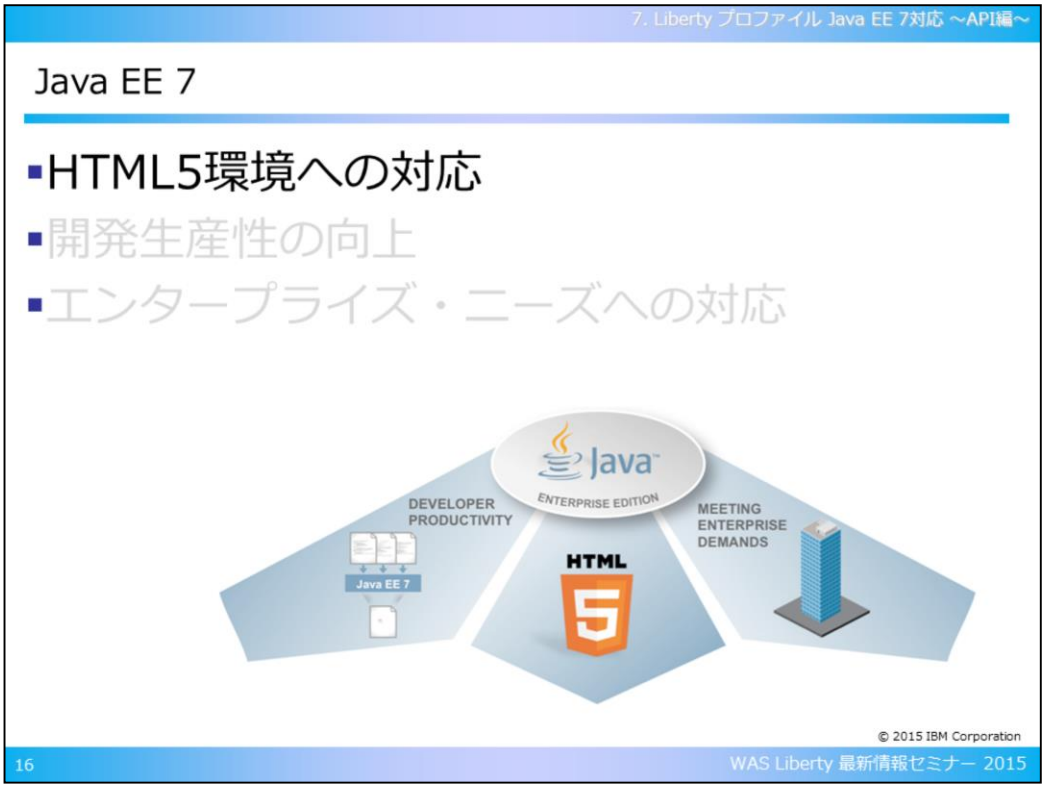


Java EE 7アプリケーションの典型的なアプリケーション構成は図のようになります。

Webアプリケーションの画面は，JSFを使用したり，Servlet/JSPなどを使用して実装されます。動的に結果を埋め込むためにELなどが使用されることもあります。また，外部のサービスとの連携にはJAX-RSやJAX-WSなどのWebサービスが使用されたり，ブラウザとの双方向通信にWebSocketが使用されたりします。

一方，ビジネスロジックの実装にはEJBやPOJOが使用されます。トランザクションの管理にはJTAが，DBとの連携にはJPAが使用されることもあります。

これらを結びつける仕組みとしてCDIが使用されます。CDIを使用することにより，プレゼンテーション層とビジネスロジック層の依存関係をへらし，独立したコンポーネントとして開発・利用することが容易になります。

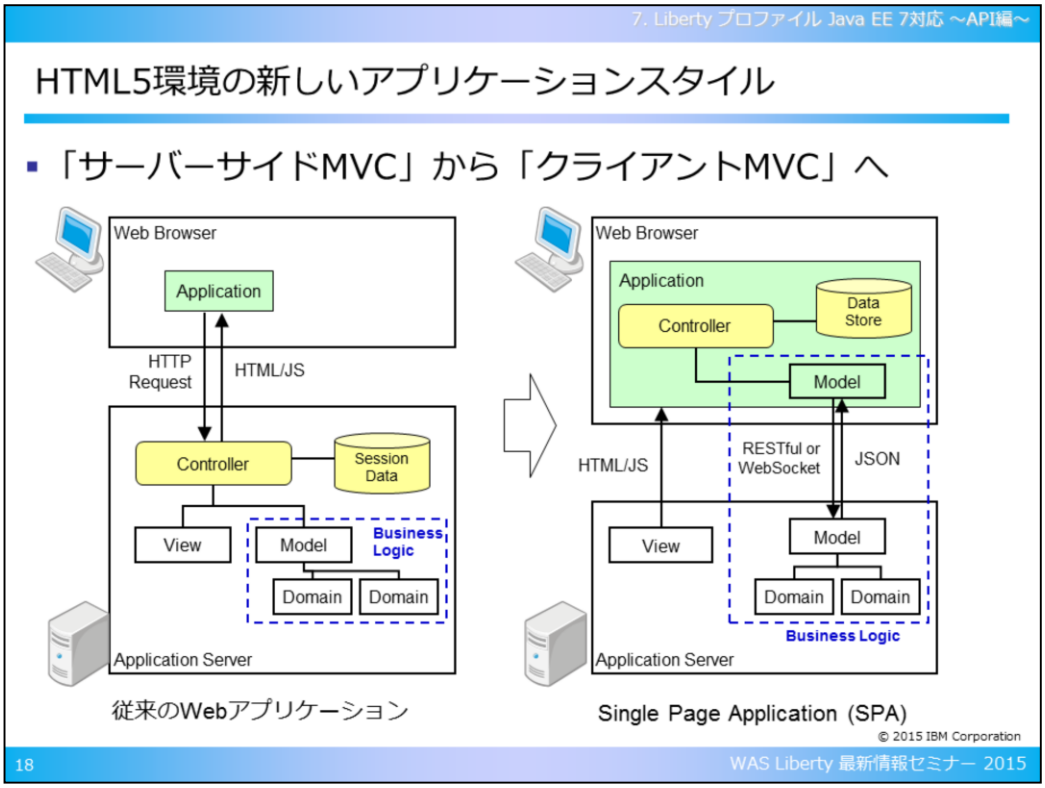


Java EE 7のゴールの一つ，「HTML5環境への対応」をどのように実現しているかをご紹介します。



2014年にW3Cから正式勧告されたHTML5は，それまでのHTML仕様と大きく性質が異なっています。

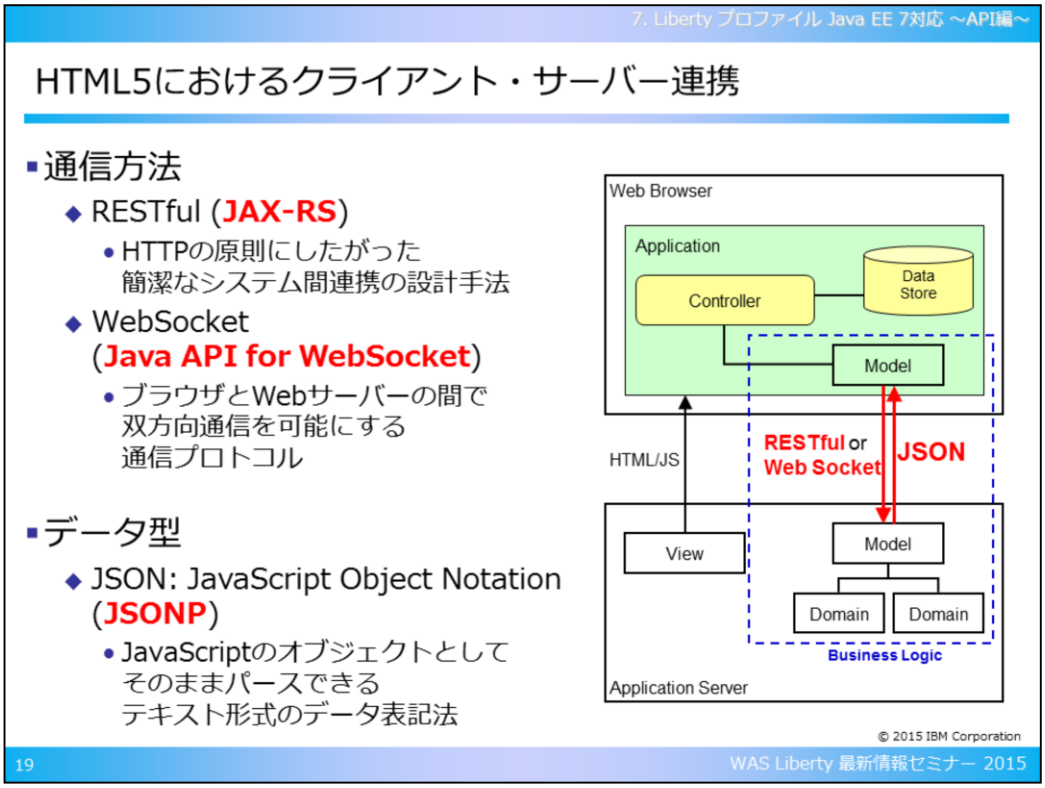
従来のHTMLがWebの画面をレンダリングする機能を中心に策定されていました。これに対し，HTML5ではCSS3やJavaScriptなどの技術を組み込むことにより，ブラウザの上で動くアプリケーションを記述するための仕様として策定されています。



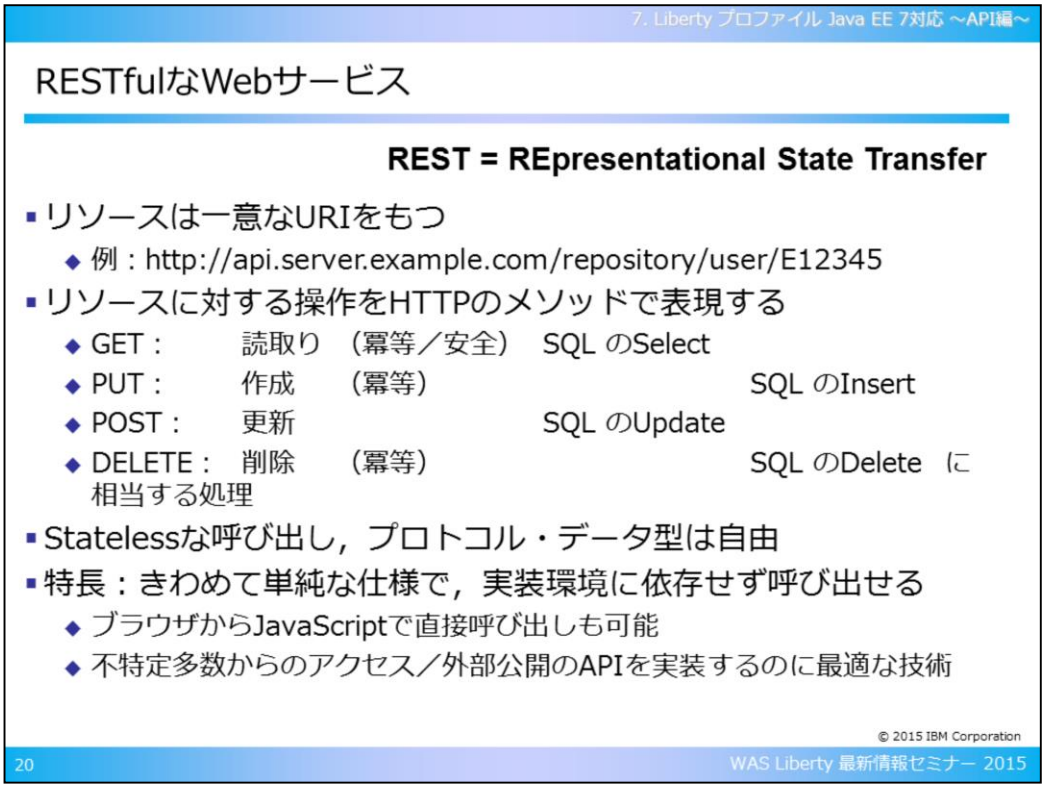
これにより，従来は全ての処理をサーバー側で実行する必要があったのですが，アプリケーション処理の大部分をブラウザの上で動くJavaScriptにまかせることもできるようになりました。これにより，ブラウザ上のアプリケーションの操作性が大きく向上します。従来の入力欄を持つだけのFORM型のアプリケーションにとどまらず，GUIによる柔軟な操作性を持ったアプリケーションを実装することができるようになりました。

かつて，クライアント側のプログラムを管理をより容易にするために，クライアント・サーバーシステムからWebアプリケーションシステムへの移行が起こりました。このときに犠牲になったクライアント側の操作性を，Webアプリケーションのメリットをのこしたまま回復できるようになったのです。

JavaScriptは，ページごとに操作対象であるDOM（Document Object Model）とよばれるデータ構造を作成します。このDOMを操作ごとに再作成することは，パフォーマンスやアプリケーション実装の点から望ましくありません。そのため基本的にはページを移動せず，一度作成したDOMを再利用しながら処理を行うという実装をとるようになります。そのため，このようなアプリケーションをSingle Page Applicationという呼び方をすることがあります。

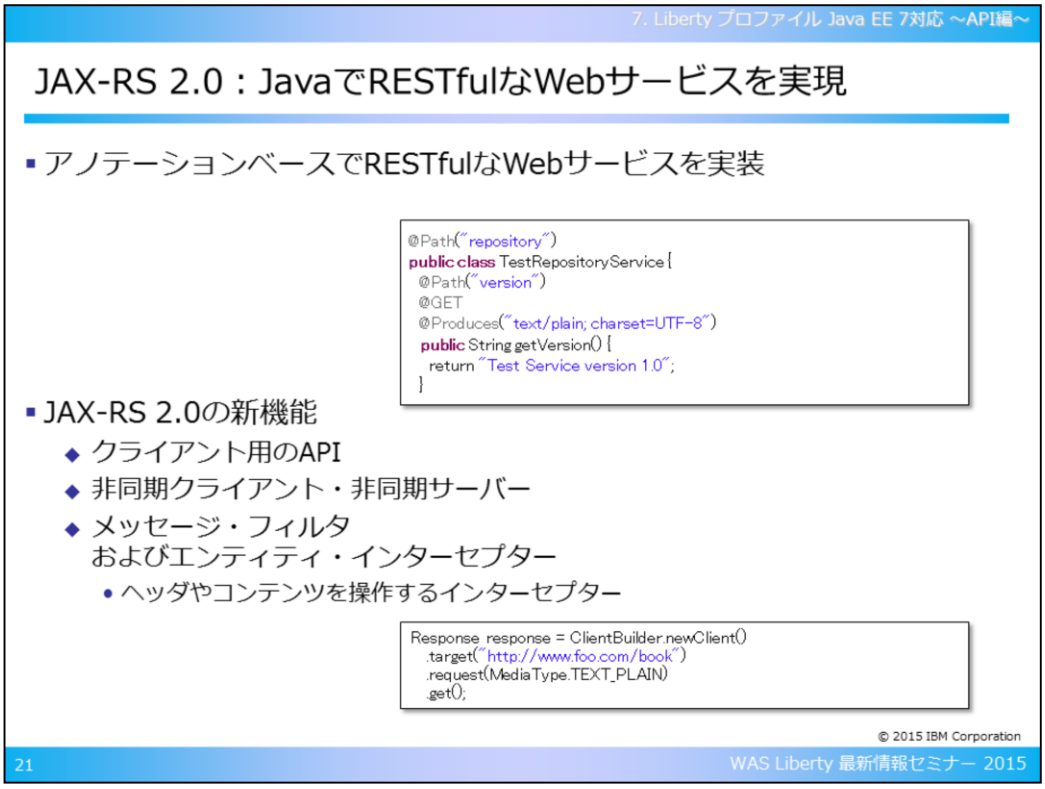


SPAでは，サーバー側と連携するための仕組みとして，RESTfulなWebサービスやWebSocket通信などが仕様がつかわれます。またやりとりされる情報のデータ型としては，JavaScritpで扱いやすいJSONがつかわれます。これらをJavaから扱うための仕様がJAX-RSやJava API for WebSocket，そしてJSONPになります。



RESTfulなWebサービスとは，特定の仕様があるわけではなく，上記のような特徴を持つように実装されたWebサービスをいいます。

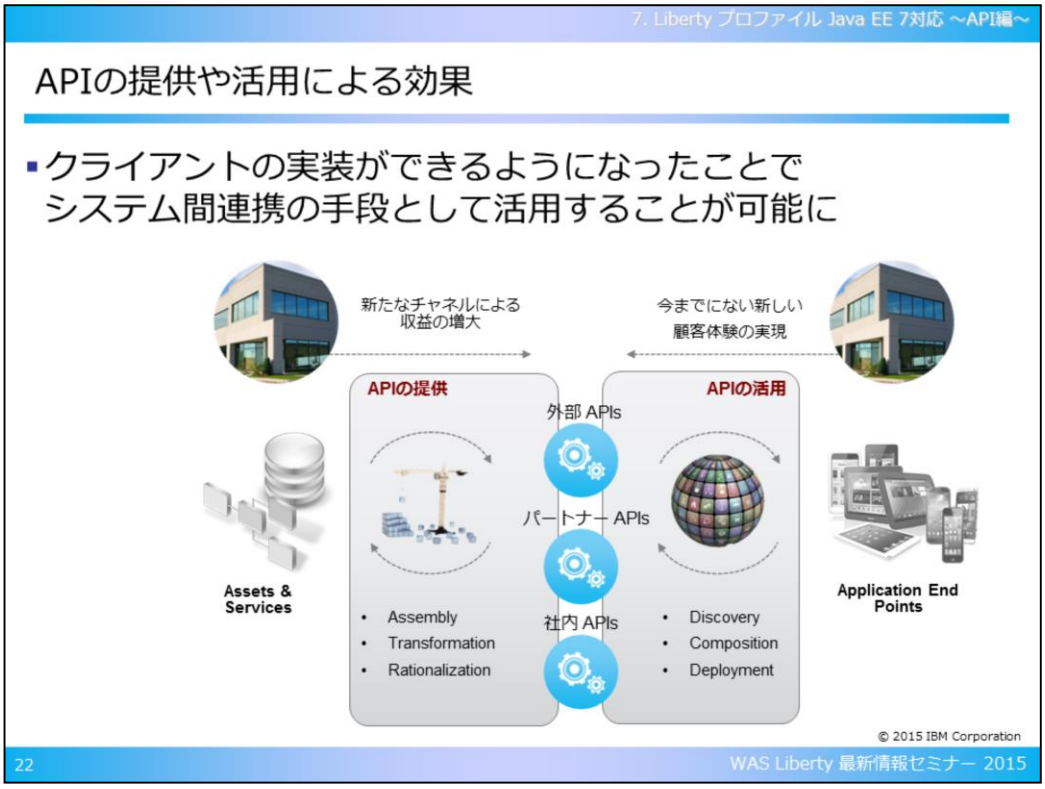
RESTfulなWebサービスは，HTTP通信を行う機能を持ったプログラムであれば，どこからでも呼び出すことができます。そのため，ブラウザのJavaScriptから呼び出すサービスや，接続相手を限定できない不特定多数からアクセスされるサービスを実装する際によく利用されます。



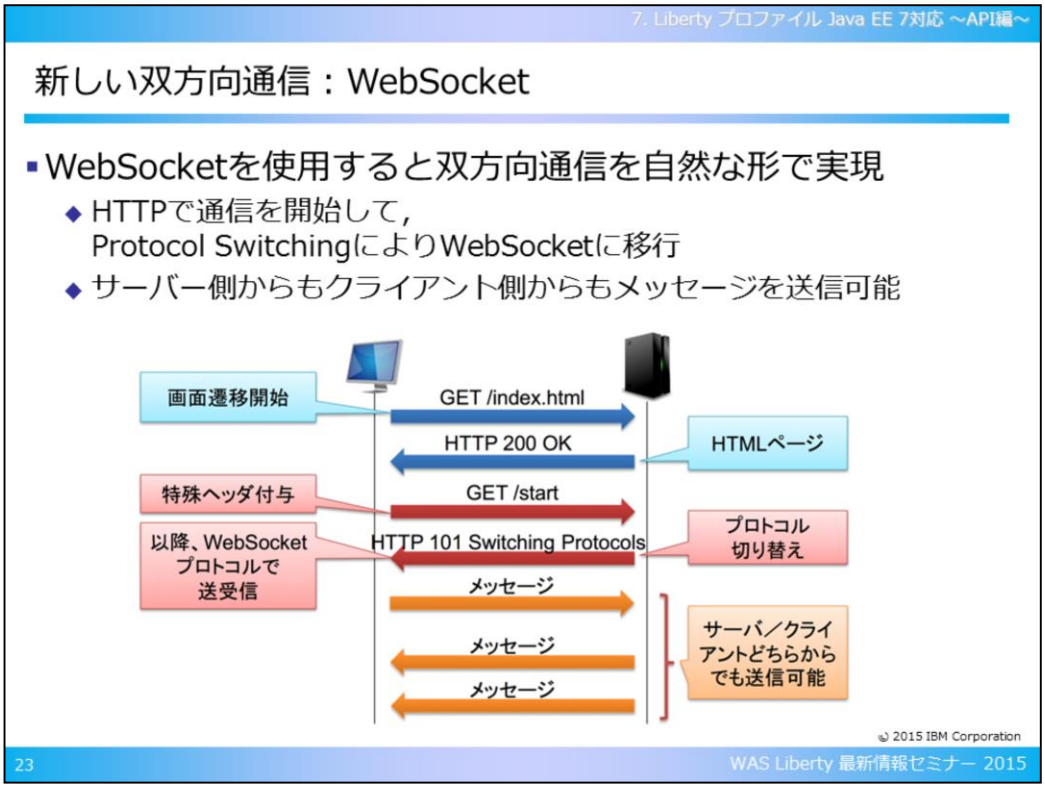
JavaでRESTfulなWebサービスを実装するための仕様として，Java EE 6からJAX-RSが追加されました。JAX-RSを使用すれば，RESTで呼び出すWebサービスを簡単に実装することができます。

Java EE 7では，このJAX-RSがメジャーバージョンアップしてJAX-RS 2.0になりました。様々な新機能がありますが，最も大きなものはクライアント用のAPIが提供されるようになったことです。これにより，単に外部にサービスを公開するだけではなく，外部のサービスを呼び出すこともできるようになりました。ブラウザとの連携だけではなく，システム間連携の仕組みとしてもJAX-RSが使用できるようになったのです。

システムを機能ごとに分割されたサービスの集合として実装する，というマイクロサービスの思想にしたがってアプリケーションを実装する場合にも，RESTfulなWebサービスは有効な実装方法となります。特に，軽量なLibertyプロファイルを使用すれば，サービスごとに個別のJVMをたてるような運用も現実的な選択肢として検討することができます。

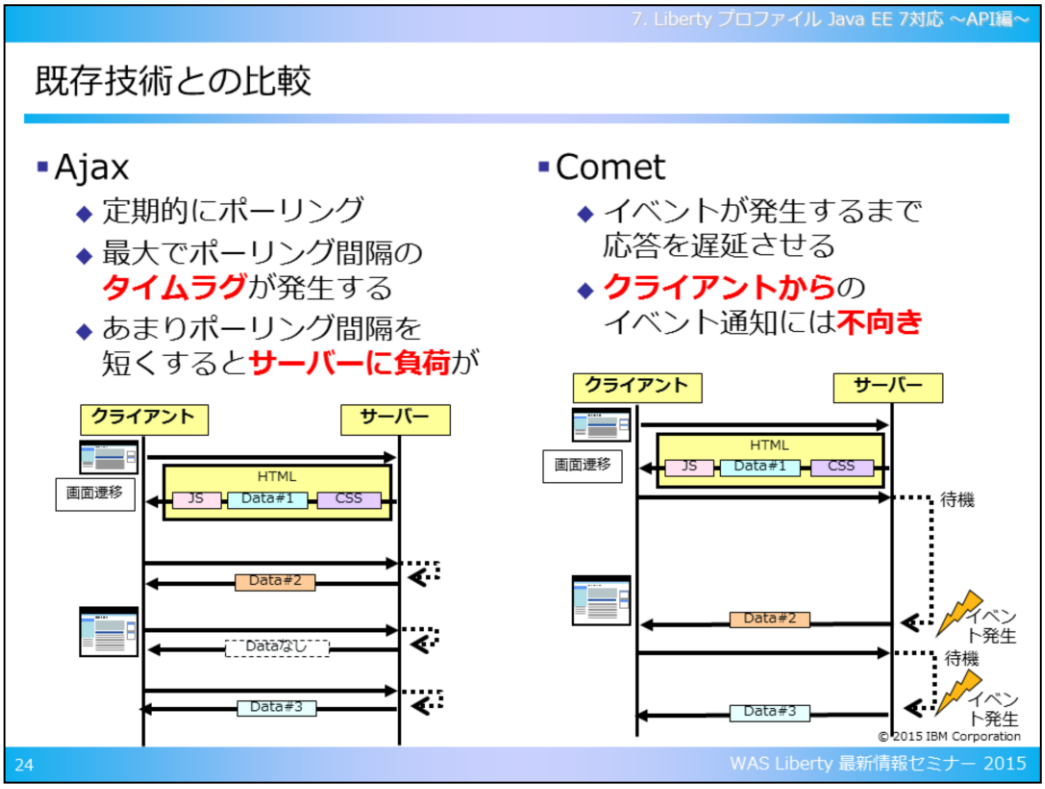


システムの機能を外部にAPIとして公開する，外部で公開されているAPIをシステムの機能として取り込む。これらのAPIエコシステムの構築は，ビジネスチャンスを増やす大きなポテンシャルを持っています。これからのシステム構築では，ぜひともAPIの考えを取り込んだ設計を取り込んでいってください。

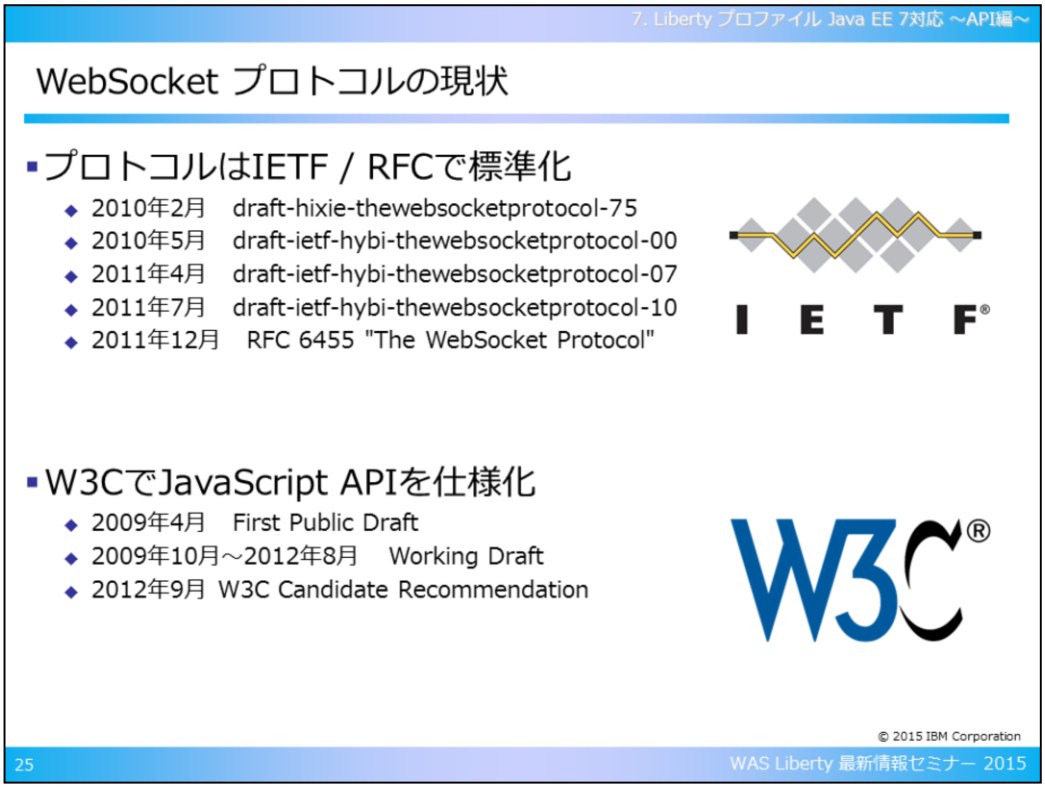


WebSocketは，ブラウザとサーバーの間で双方向通信を実現するための通信プロトコルです。

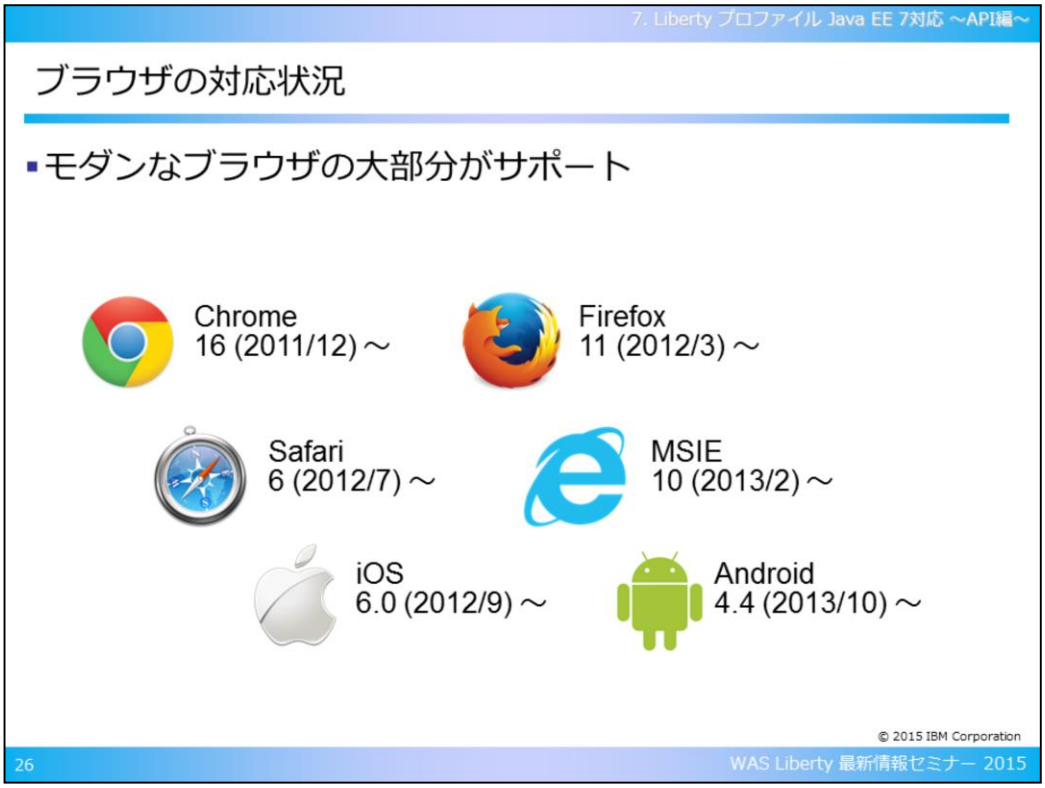
HTTPは，基本的にクライアントからのリクエストにサーバーがレスポンスを返すことを前提としているプロトコルであるため，サーバー側のイベントをトリガーにクライアントにメッセージを送信するのが難しいという欠点を持っています。WebSocketを使用すれば，このサーバーからの任意のタイミングでのメッセージを送信することができます。



HTTP通信を使用してサーバーサイドからメッセージを送信するための手段としては，Ajaxによるポーリングや，Cometというような仕組みが使用されていました。ですが，それぞれに欠点や制限をかかえています。これに対してWebSocketは，サーバー・クライアントの双方が任意のタイミングでメッセージを送信することが自然な実装で実現できます。

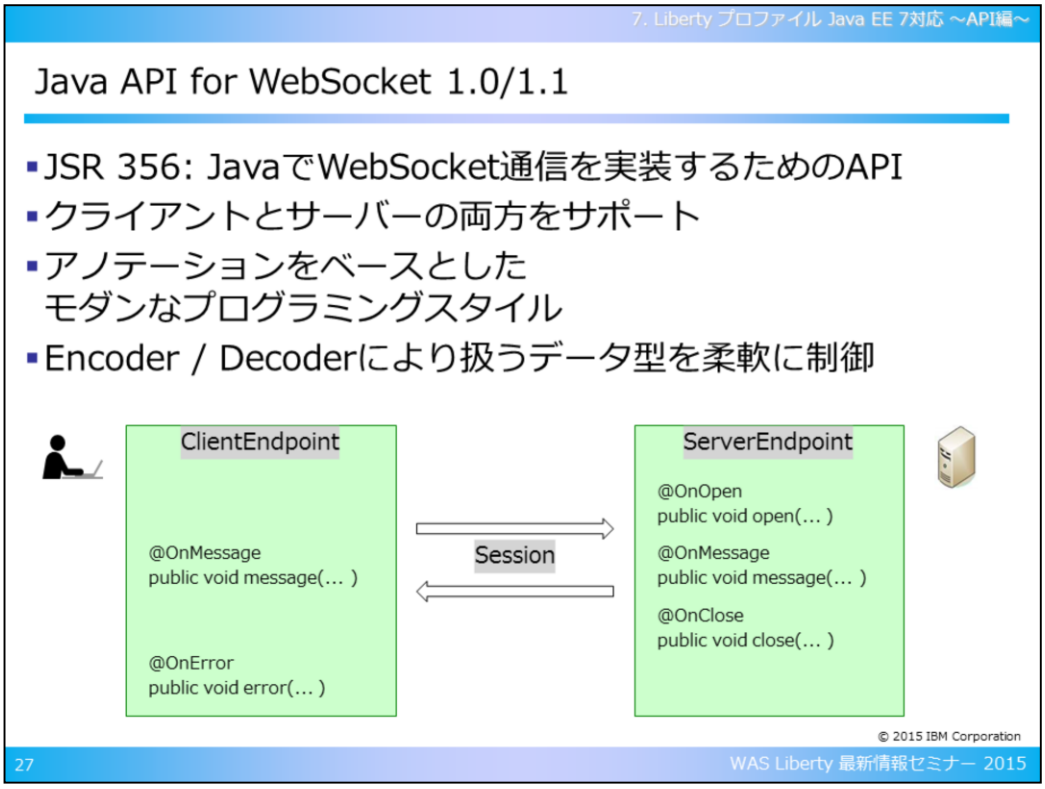


WebSocketのプロトコルおよびブラウザ上のJavaScriptでの仕様は，それぞれ公的機関で策定および公開が完了しています。



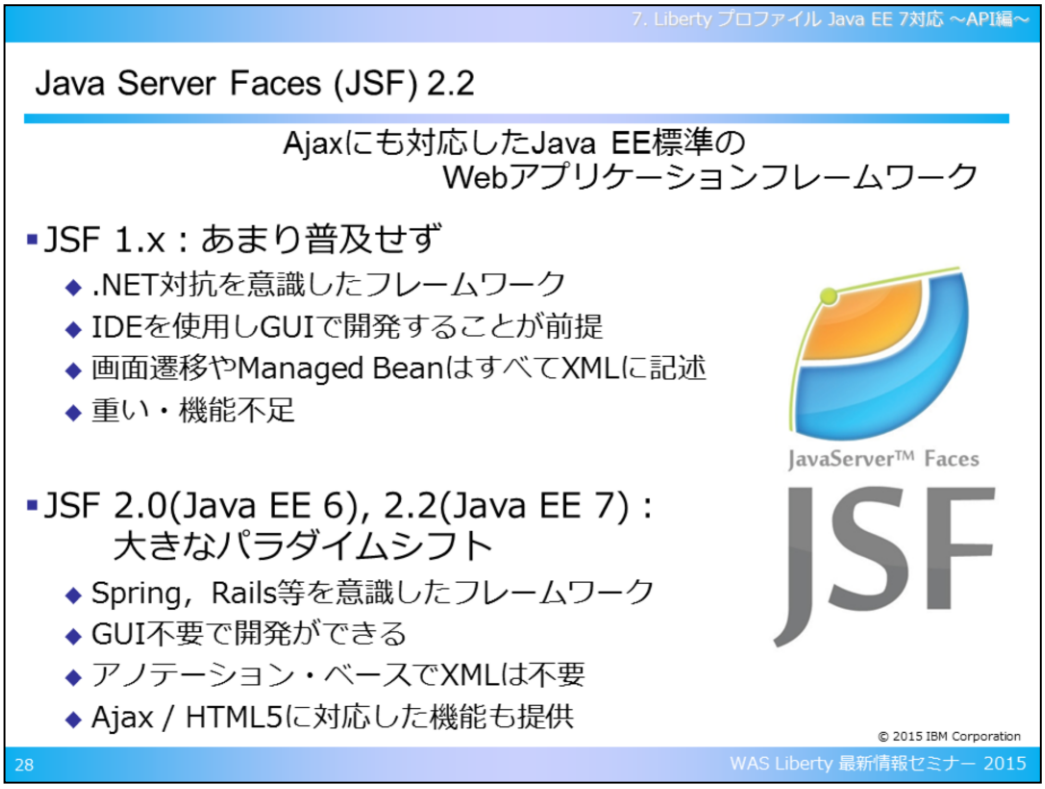
また，ブラウザがわの対応も進んできました。WindowsのIE 9以前を考慮しなくてはいけない場合を除いて，デスクトップ環境での使用は可能です。

モバイル環境では，実質的にほとんどの環境で利用が可能になっています（Android 4.3以前のOS標準ブラウザはWebSocketには対応していませんが，この標準ブラウザにはセキュリティ上の欠陥が多く報告されているため，Chromeの使用が推奨されています）。



Java EE 7から追加されたJava API for WebSocketでは，サーバー側のEndpointだけでなくクライアント側のEndpointも簡単実装することができるようになっています。

単純な文字列やJSONの送受信だけではなく，Decoder / Encoderを実装することによりJavaのオブジェクトを送受信することもできるようになっています。

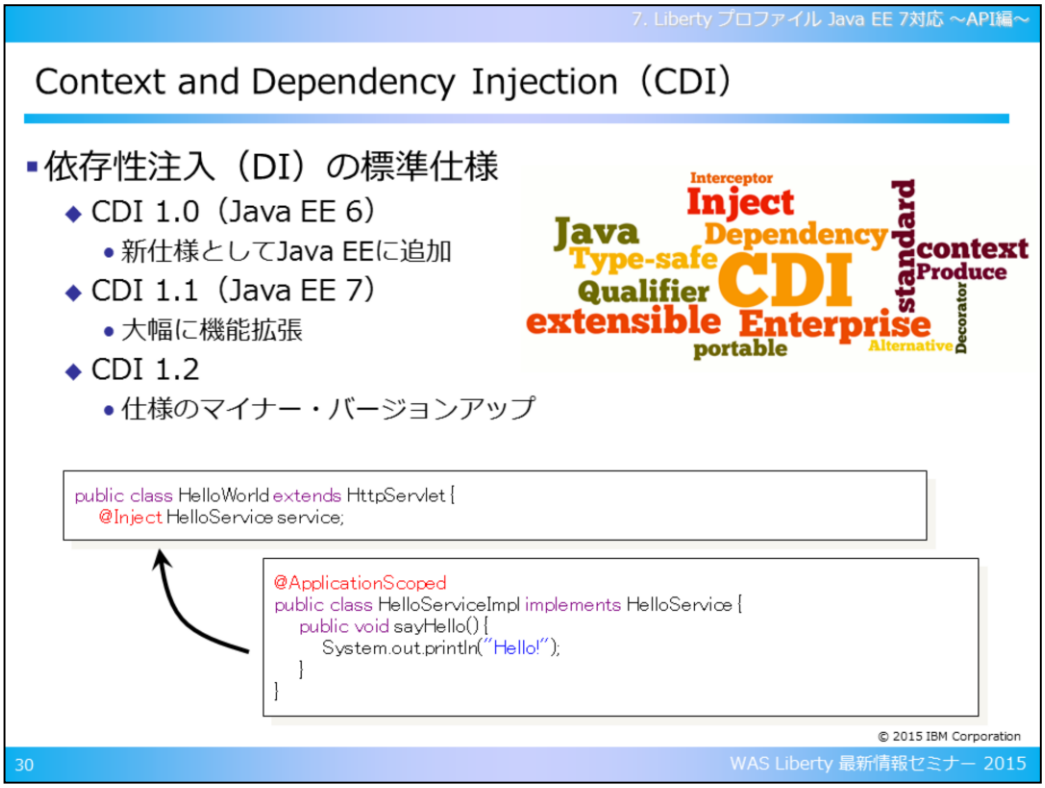


Java EEに含まれるWebアプリケーションの標準フレームワークがJSFです。

従来のJSFは，いろいろと問題点も多く，広く普及しているとは言えない状況でした。しかし，JSF 2.0のメジャーバージョンアップ以降，使いやすいフレームワークとして生まれ変わりました。国内外の採用数も増えており，Webアプリケーション実装の有力な選択肢の一つとなっています。

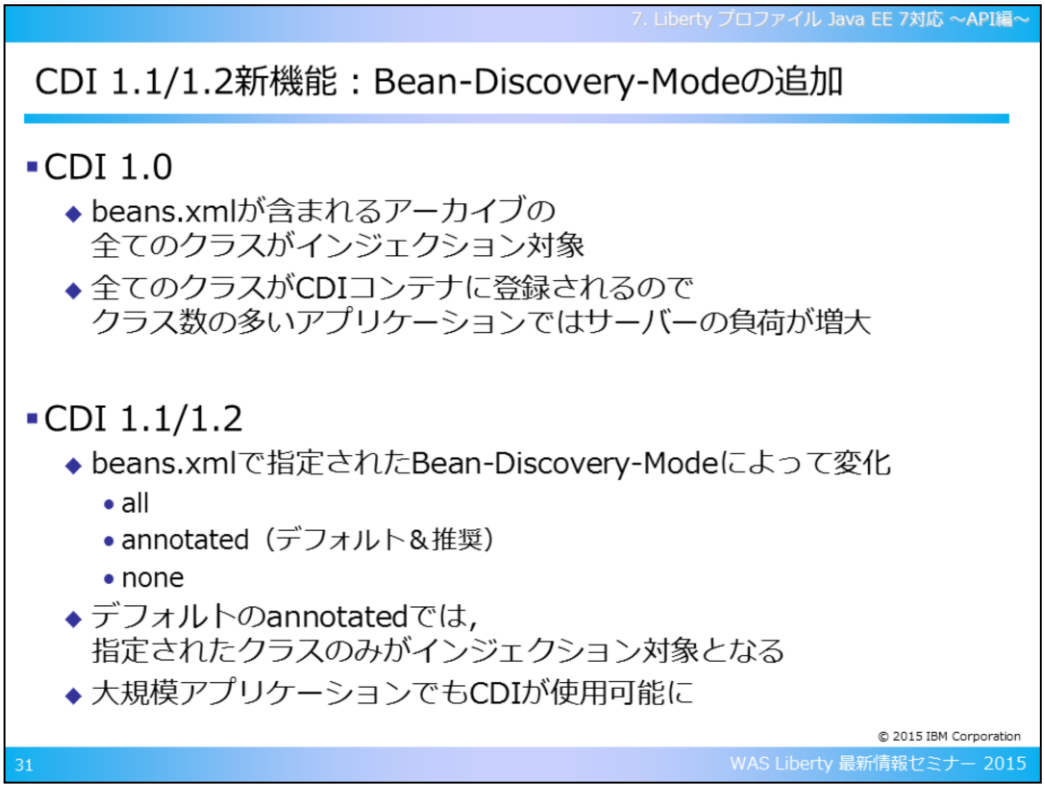


次にJava EE 7のゴールの二つめ，「開発生産性の向上」をどのように実現しているかをご紹介します。



依存性注入は，Javaのコードを実装する側と利用する側の依存性を解消し，インジェクションコンテナによって動的に依存性を解決するための仕組みです。アプリケーションコードの開発やテストを効率的におこなうことができます。また，インターセプターにより共通処理を追加することができるなど，AOP（Aspect Oriented Programming）を実現するための仕組みとしても利用できます。

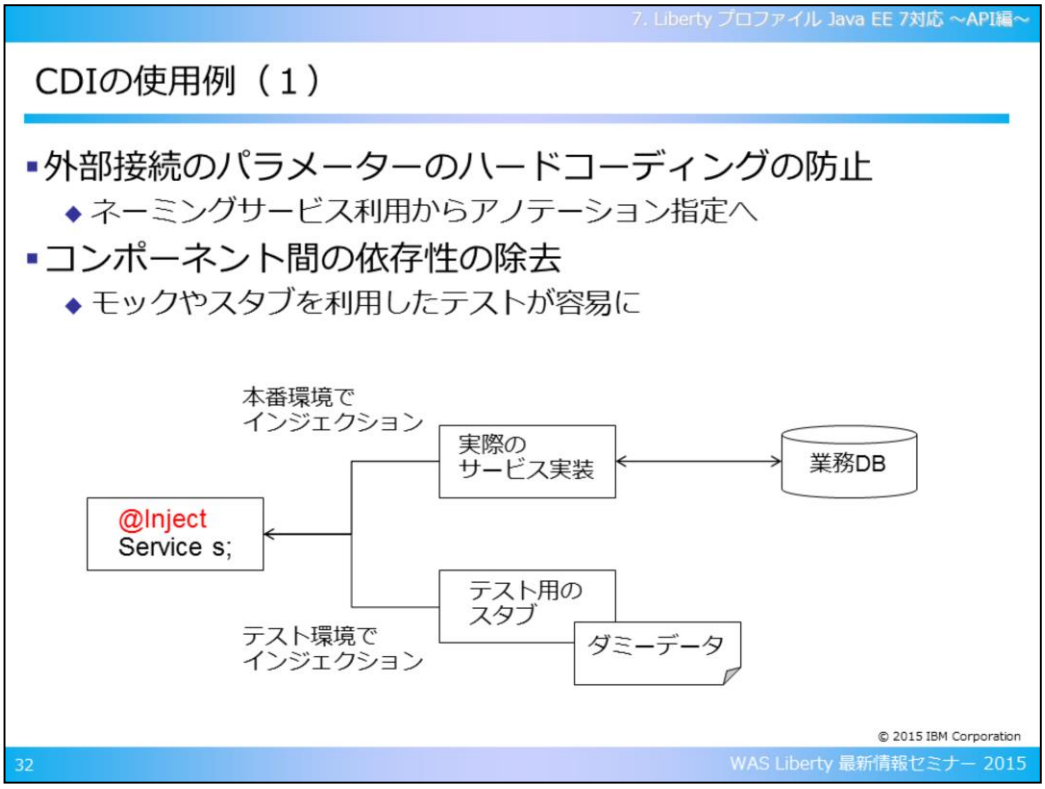
CDIは，Java EE 7で大きく改良されさました。最初の仕様であるCDI 1.0であった多くの問題が解消されています。



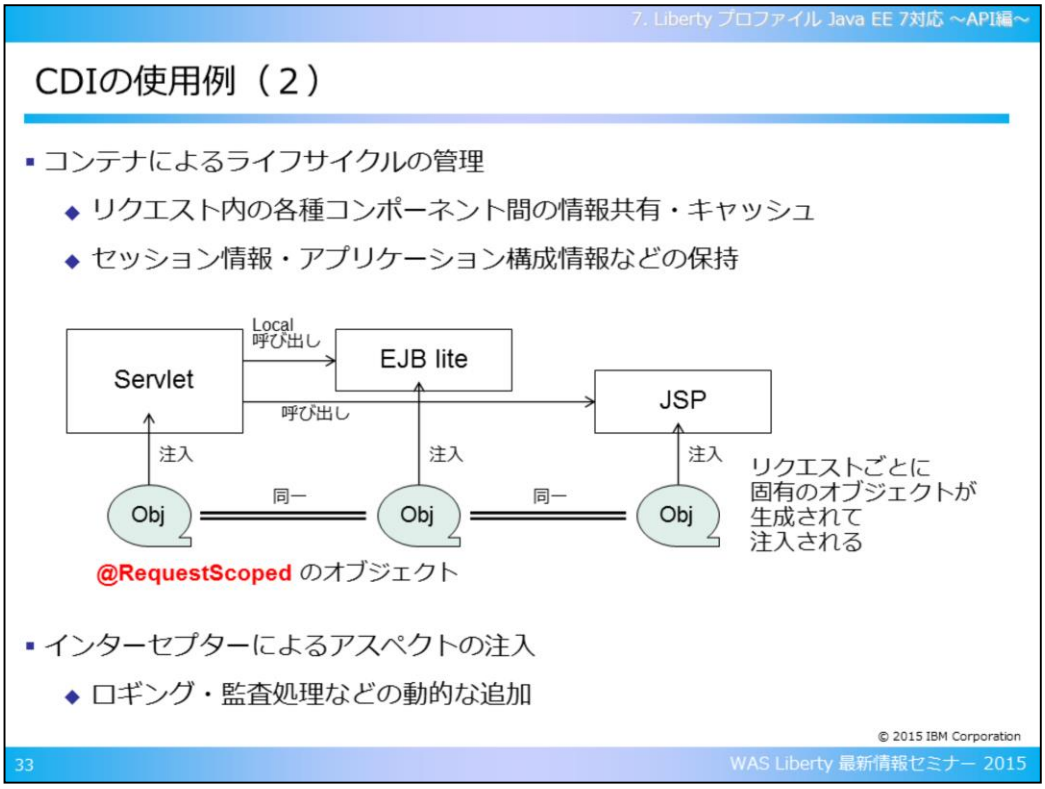
最も大きな変更点は，Bean-Discovery-Modeのannotatedが導入されたことです。

従来のCDIでは，アプリケーションに含まれる全てのクラスがインジェクションの対象となってしまっていたため，クラスが数万から十数万も含まれるような巨大なアプリケーションでは，CDIコンテナに異常な負荷がかかり実用にならないほどの速度低下が発生してしまっていました。

これに対してCDI 1.1以降では，アノテーションで指定したクラスのみをインジェクションの対象とすることができるようになりました。これにより大規模なアプリケーションであってもパフォーマンスの低下なくCDIを利用できるようになりました。

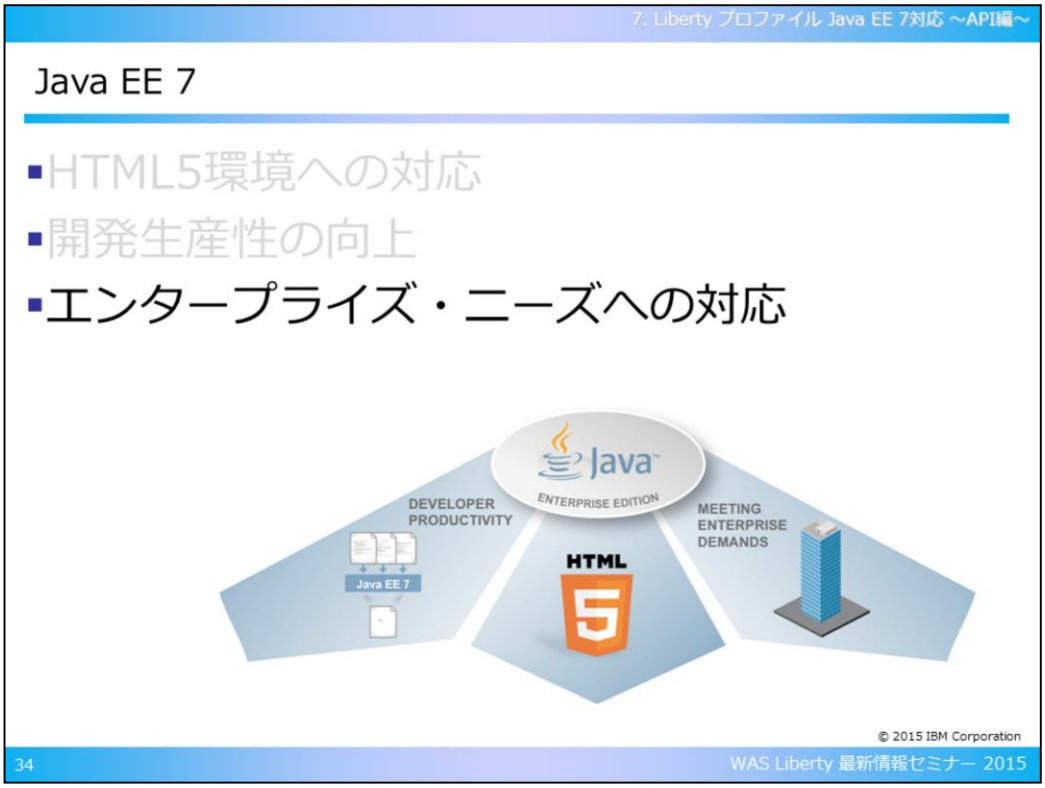


CDIの利点の一つは，アプリケーションのコードを変更することなく，インジェクションする対象を変更することができ，モックやスタブなどを使用したテストが簡単にできることです。もちろん，本番と同じクラス名を持ったテスト用のスタブを作成することでも同様の機能を実現できますが，異なる機能を持つクラスを同名で作成することは，プロジェクトに危険と混乱をもたらすためおすすめできません。

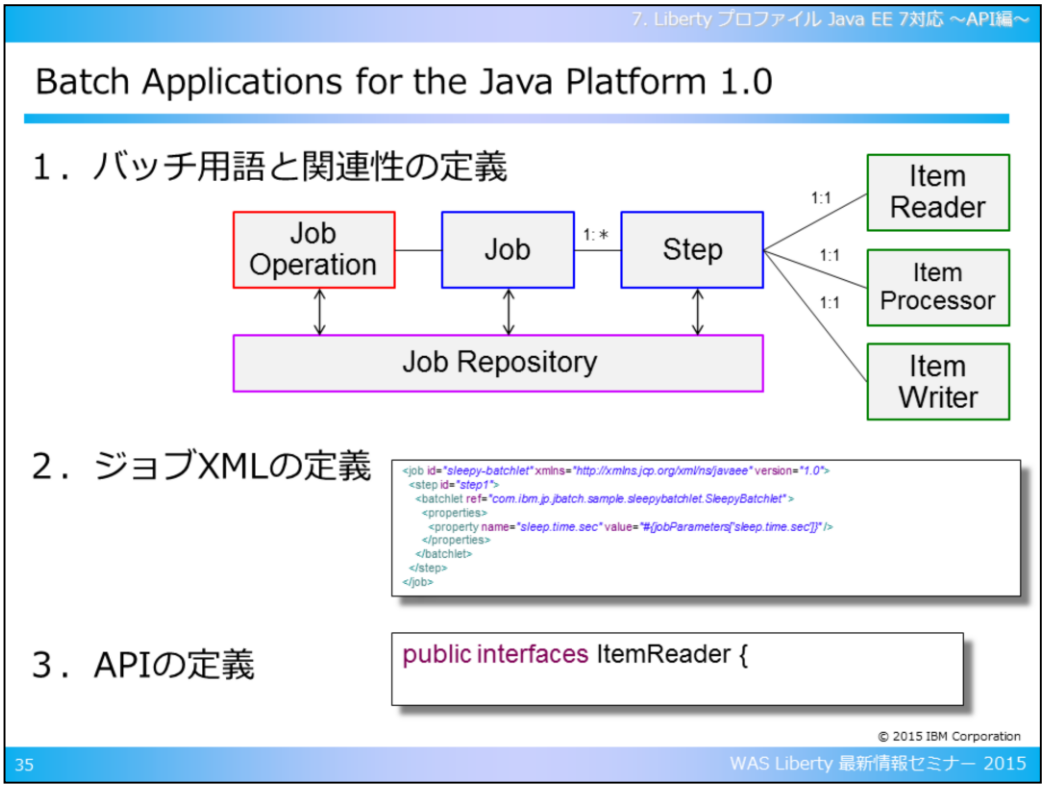


CDIコンテナはインジェクションするオブジェクトのライフサイクル管理も行ってくれます。リクエストやセッションのスコープを持つオブジェクトは，プログラムの複数の箇所から安全に情報を共有する手段として利用することができます。

また，インジェクションを行う際に処理を追加するインターセプターを活用すれば，AOPの実現も簡単にできます。ビジネスロジックと直接関係のない共通処理を，各所に挿入したい場合などに大きな力を発揮します。



最後はJava EE 7のゴールの三つ目，「エンタープライズ・ニーズへの対応」をどのように実現しているかをご紹介します。

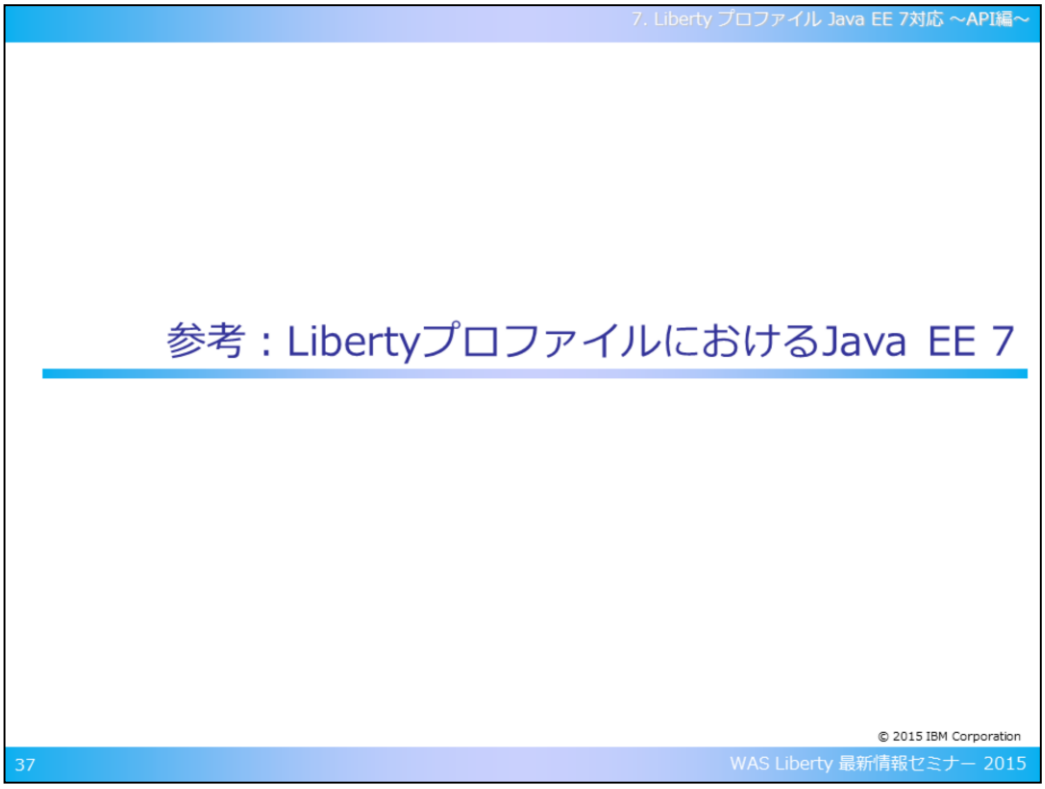


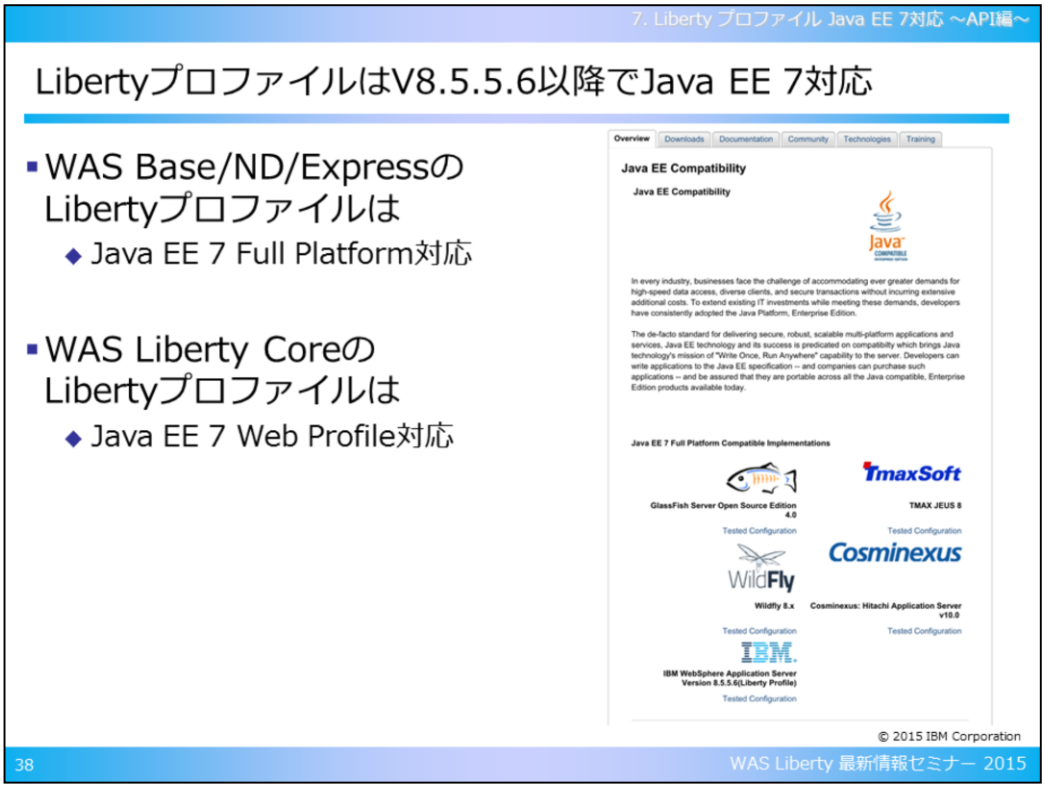
JMSの抜本的な改良などもありますが，エンタープライズ・ニーズへの対応でもっともインパクトの大きな変更が，Javaバッチアプリケーションの標準化でしょう。いままで，フレームワークやアプリケーションコードで独自に実装されていたバッチアプリケーションが，環境に依存しない標準的な方法でコーディングできるようになります。

現時点のJava Batch仕様は，アプリケーションの基本骨格をさだめたものにとどまっていますが，今後の拡張で多くの機能が拡張され，エンタープライズアプリケーションの基本のになう重要な仕様となっていくと考えられます。



Java EE 7の技術を使用してアプリケーションを設計・開発するためのガイドを，IBMのdeveloperWorksサイトで公開中です。このあとも順次公開される予定です。


7. Liberty プロファイル Java EE 7対応 ～API編～
参考：LibertyプロファイルにおけるJava EE 7
© 2015 IBM Corporation
37
WAS Liberty 最新情報セミナー 2015

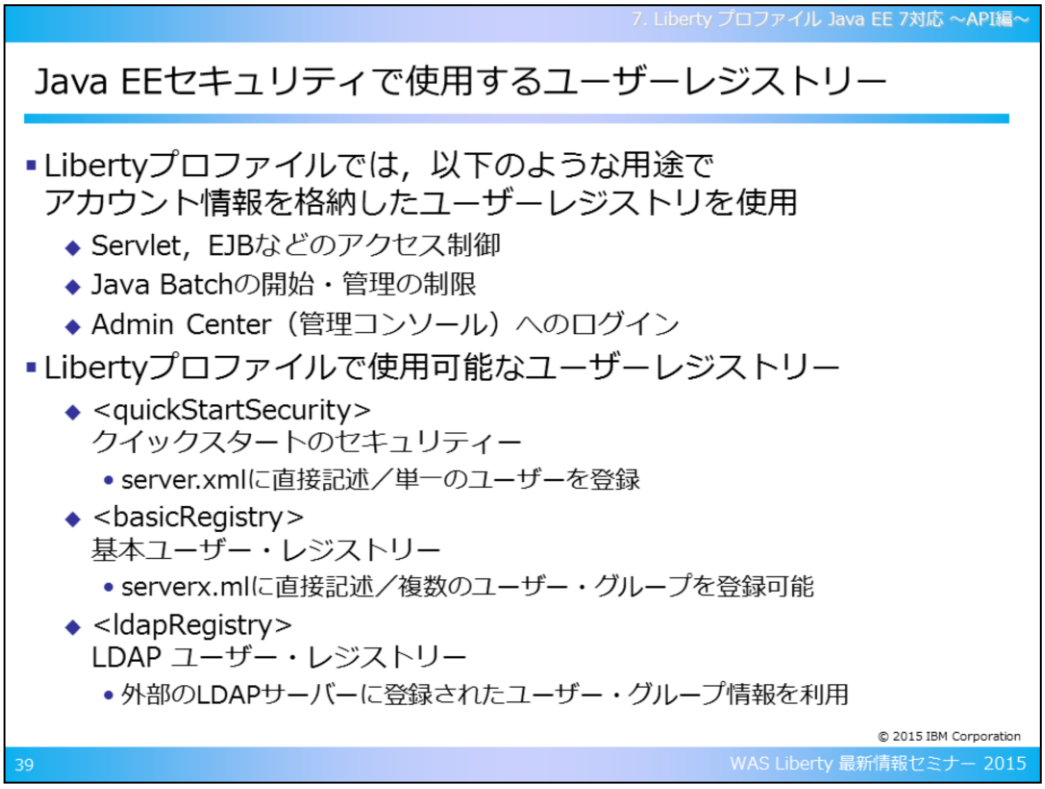


WAS Libertyプロファイルは2015年6月に出荷されたV8.5.5.6よりJava EE 7に対応しています。対応レベルはエディションによって異なります。Base/ND/Expressエディションはは Java EE 7 Full Platformに，Liberty CoreエディションはJava EE 7 Web Profileに対応しています。

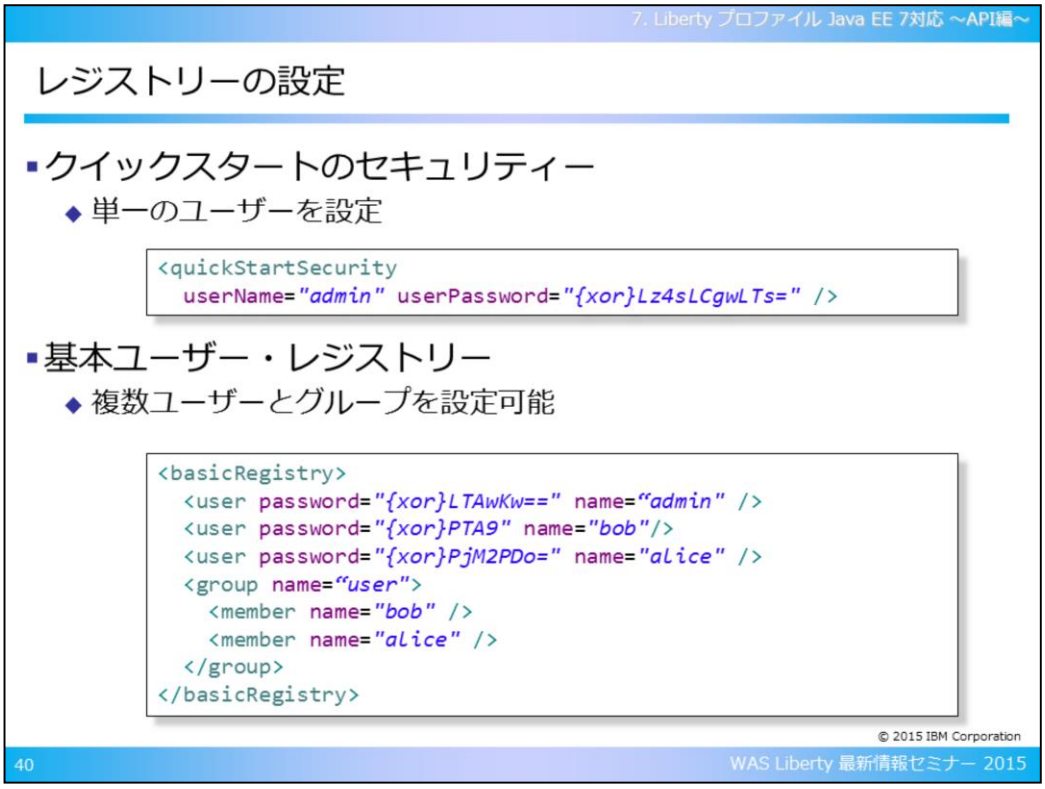
また，Java EE 7仕様で定義されている全てを実装しているのではなく，Optional指定されている仕様の一部（JAX-RやJAX-RPCなど）は実装されていません。

出典：Java EE Compatibility

http://www.oracle.com/technetwork/java/javaee/overview/compatibility-jsp-136984.html

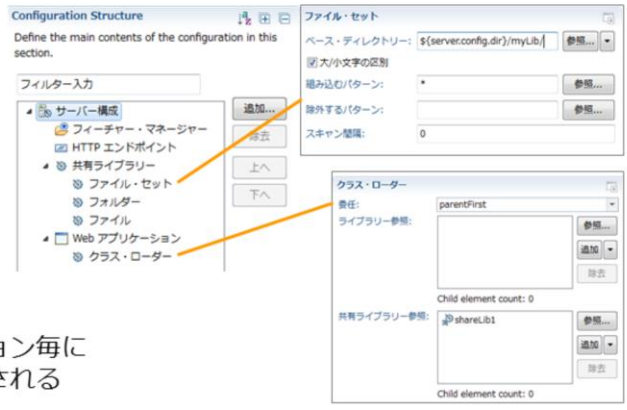


Java EEアプリケーションでは，様々な箇所で認証のためのユーザー・レジストリが利用されます。Libertyプロファイルでは，構成ファイルに簡単にテスト用・管理者用のユーザーアカウントを設定することができます。もちろん，外部のLDAPサーバーにおかれたユーザー・レジストリを利用することもできます。


7. Liberty プロファイル Java EE 7対応 ～API編～
レジストリーの設定
クイックスタートのセキュリティー
単一のユーザーを設定
<quickStartSecurity
userName="admin" userPassword="{xor}Lz4sLCgwLTs=" />
基本ユーザー・レジストリー
複数ユーザーとグループを設定可能
<basicRegistry>
<user password="{xor}LTAwKw==" name="admin" />
<user password="{xor}PTA9" name="bob"/>
<user password="{xor}PjM2PDo=" name="alice" />
<group name="user">
<member name="bob" />
<member name="alice" />
</group>
</basicRegistry>
© 2015 IBM Corporation
40
WAS Liberty 最新情報セミナー 2015

ライブラリ・共有ライブラリ

- ライブラリを構成
  - ファイル・セット，フォルダー，ファイル
- アプリケーションのクラス・ローダーに参照を定義
  - ライブラリー参照（privateLibraryRef）
    - 参照するアプリケーション毎に個別のクラスがロードされる
  - 共有ライブラリー参照（commonLibraryRef）
    - 参照するアプリケーション間でクラスが共有される
    - クラス変数などは共通のものを参照可能



© 2015 IBM Corporation

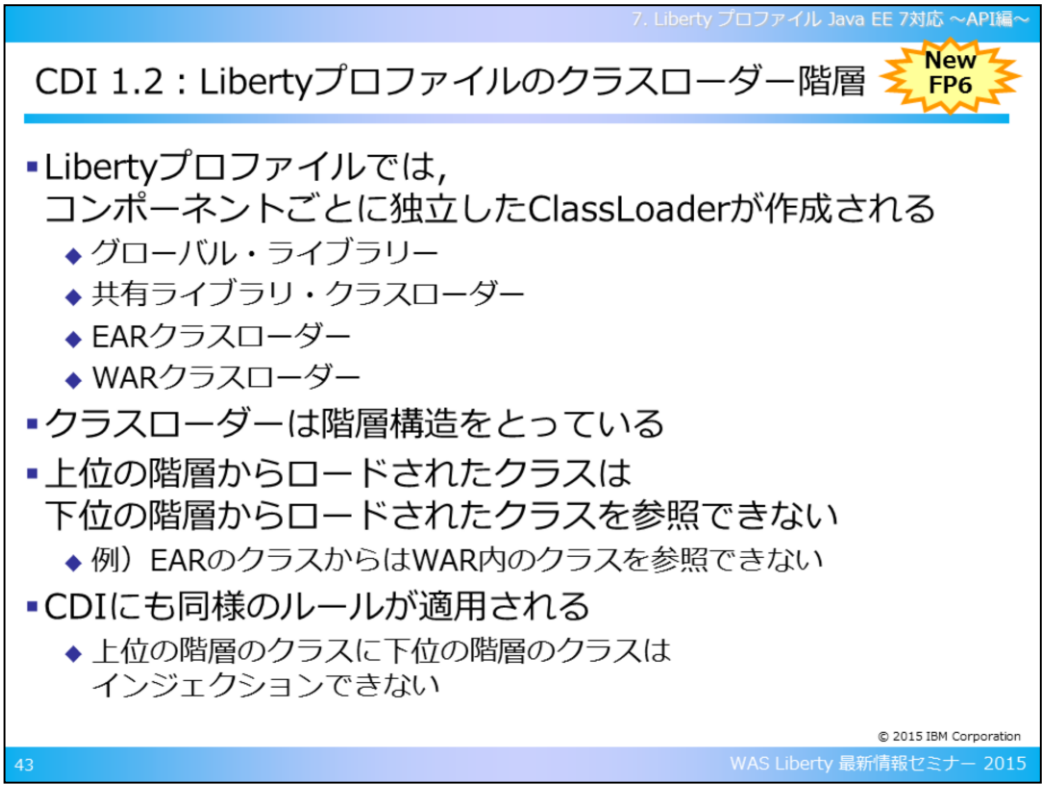
グローバル・ライブラリ

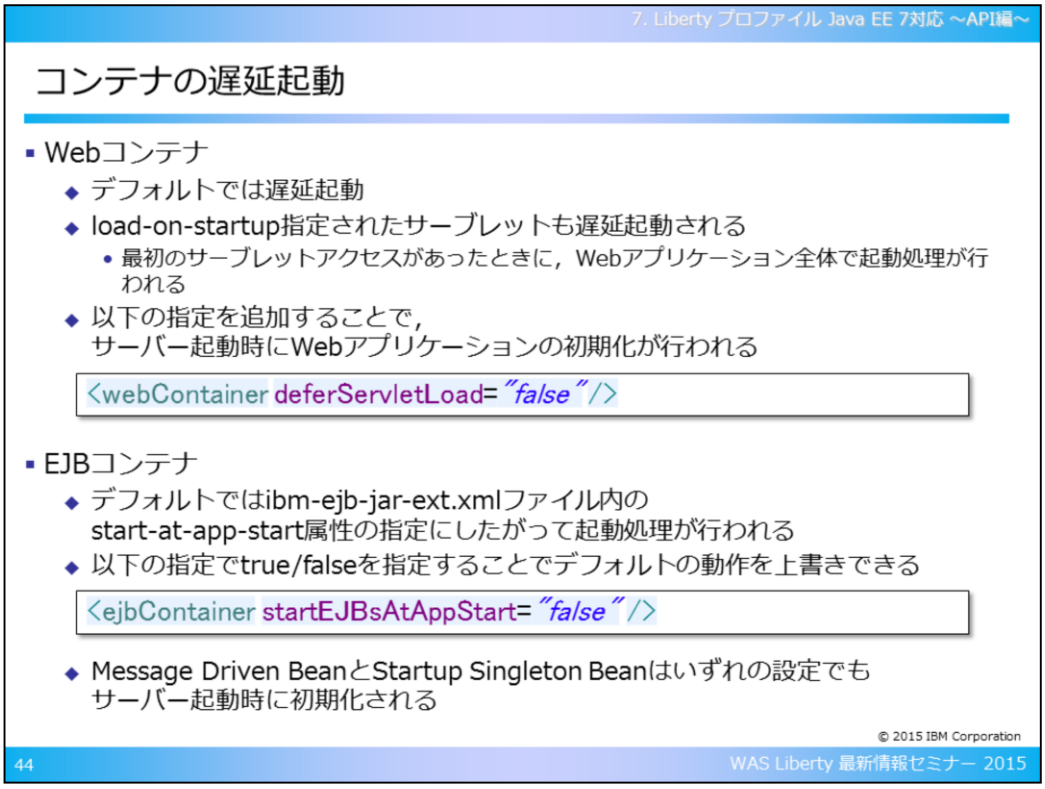
- アプリケーションが明示的なクラスローダー指定を持たない場合（dropinsに置かれた場合など），以下のディレクトリに置かれたJarファイルは，事前の構成をすることなく，自動的にJava EEアプリケーションから参照できる
  - ${shared.config.dir}/lib/global
    - デフォルトでは<wlp>/usr/shared/config/lib/global
  - ${server.config.dir}/lib/global
    - デフォルトでは<wlp>/usr/servers/<server_name>/lib/global
- <application>に<classloader>指定がある場合には，明示的にglobalを追加する。

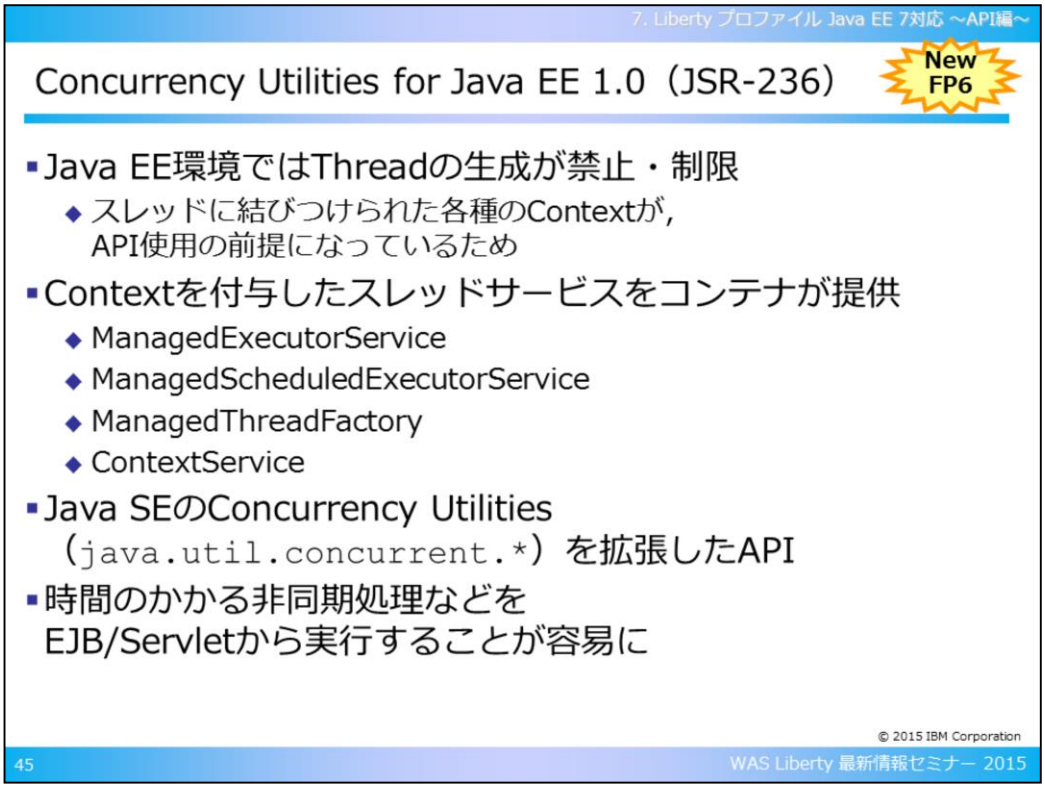
```
<application name="Scholar" type="ear" location="scholar.ear">
   <classloader commonLibraryRef="Alexandria, global" />
</application>
```

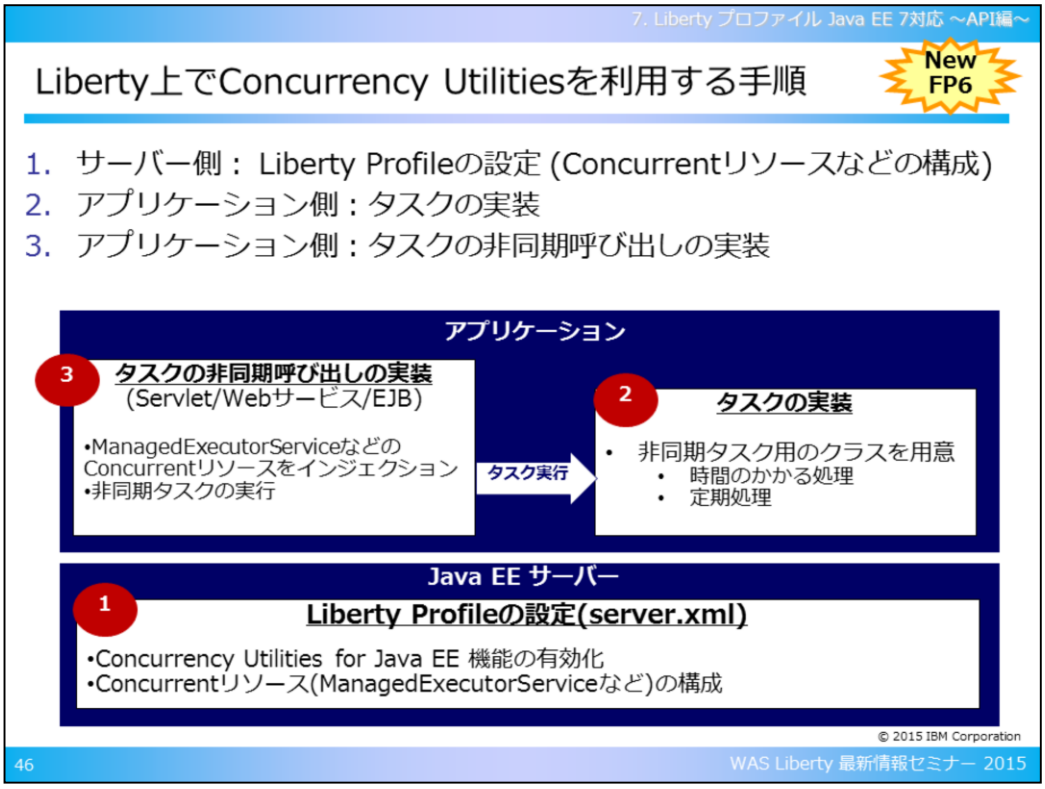
- OGSiアプリケーションなどからは参照できない

© 2015 IBM Corporation

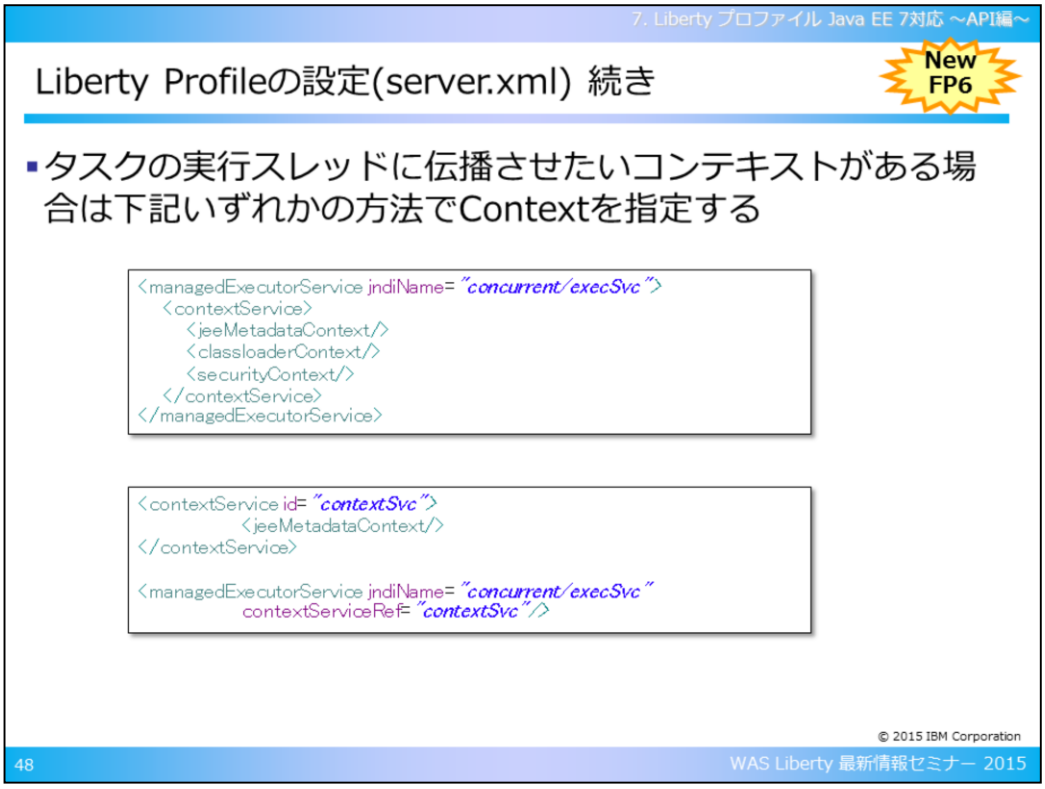

7. Liberty プロファイル Java EE 7対応 ～API編～
CDI 1.2：Libertyプロファイルのクラスローダー階層
New FP6
Libertyプロファイルでは,
コンポーネントごとに独立したClassLoaderが作成される
グローバル・ライブラリー
共有ライブラリ・クラスローダー
EARクラスローダー
WARクラスローダー
クラスローダーは階層構造をとっている
上位の階層からロードされたクラスは
下位の階層からロードされたクラスを参照できない
例）EARのクラスからはWAR内のクラスを参照できない
CDIにも同様のルールが適用される
上位の階層のクラスに下位の階層のクラスは
インジェクションできない
© 2015 IBM Corporation
43
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
コンテナの遅延起動
Webコンテナ
デフォルトでは遅延起動
load-on-startup指定されたサーブレットも遅延起動される
最初のサーブレットアクセスがあったときに，Webアプリケーション全体で起動処理が行われる
以下の指定を追加することで，
サーバー起動時にWebアプリケーションの初期化が行われる
<webContainer deferServletLoad="false" />
EJBコンテナ
デフォルトではibm-ejb-jar-ext.xmlファイル内の
start-at-app-start属性の指定にしたがって起動処理が行われる
以下の指定でtrue/falseを指定することでデフォルトの動作を上書きできる
<ejbContainer startEJBsAtAppStart="false" />
Message Driven BeanとStartup Singleton Beanはいずれの設定でも
サーバー起動時に初期化される
© 2015 IBM Corporation
44
WAS Liberty 最新情報セミナー 2015

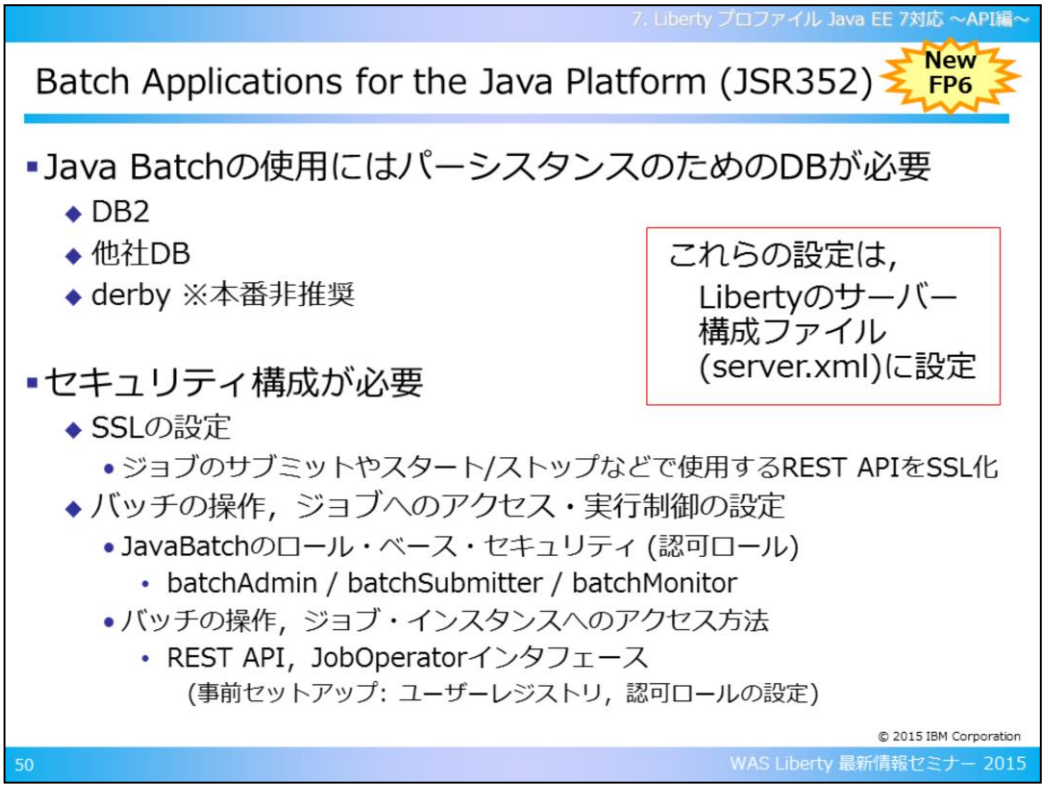

7. Liberty プロファイル Java EE 7対応 ～API編～
Concurrency Utilities for Java EE 1.0（JSR-236）
New FP6
Java EE環境ではThreadの生成が禁止・制限
スレッドに結びつけられた各種のContextが，API使用の前提になっているため
Contextを付与したスレッドサービスをコンテナが提供
ManagedExecutorService
ManagedScheduledExecutorService
ManagedThreadFactory
ContextService
Java SEのConcurrency Utilities（java.util.concurrent.*）を拡張したAPI
時間のかかる非同期処理などをEJB/Servletから実行することが容易に
© 2015 IBM Corporation
45
WAS Liberty 最新情報セミナー 2015

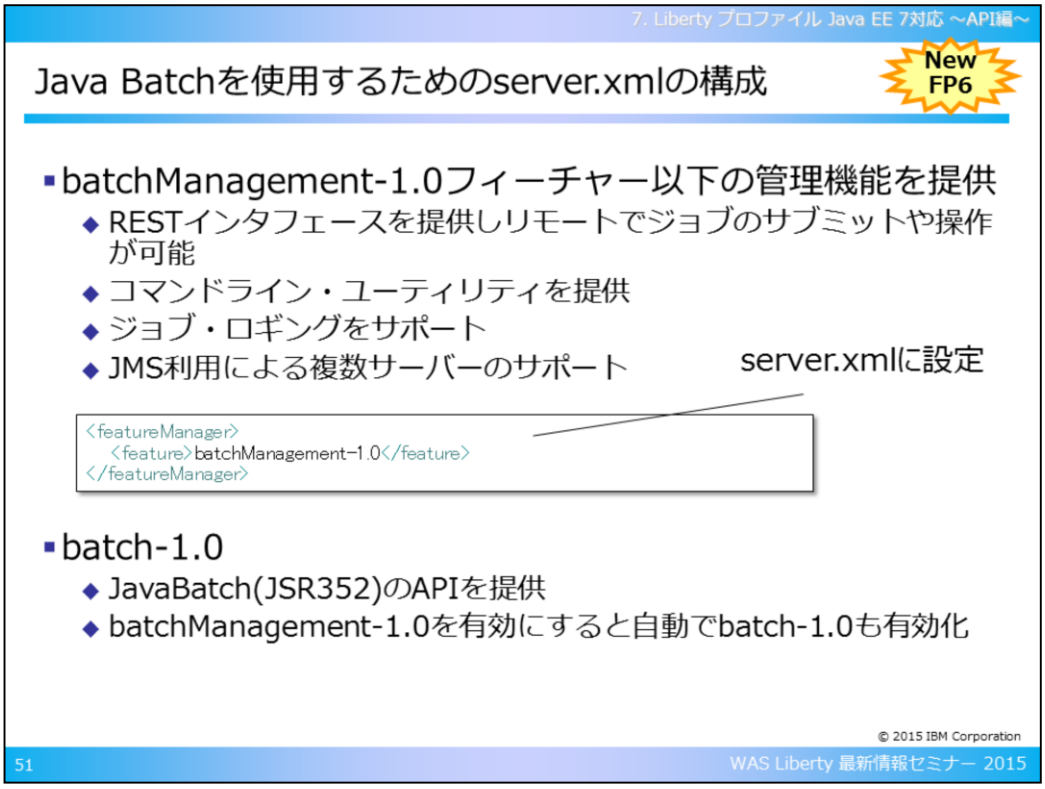

7. Liberty プロファイル Java EE 7対応 ～API編～
Liberty上でConcurrency Utilitiesを利用する手順
New FP6
1. サーバー側： Liberty Profileの設定 (Concurrentリソースなどの構成)
2. アプリケーション側：タスクの実装
3. アプリケーション側：タスクの非同期呼び出しの実装
アプリケーション
3
タスクの非同期呼び出しの実装
(Servlet/Webサービス/EJB)
•ManagedExecutorServiceなどの Concurrentリソースをインジェクション
•非同期タスクの実行
タスク実行
2
タスクの実装
• 非同期タスク用のクラスを用意
• 時間のかかる処理
• 定期処理
Java EE サーバー
1
Liberty Profileの設定(server.xml)
•Concurrency Utilities for Java EE 機能の有効化
•Concurrentリソース(ManagedExecutorServiceなど)の構成
© 2015 IBM Corporation
46
WAS Liberty 最新情報セミナー 2015

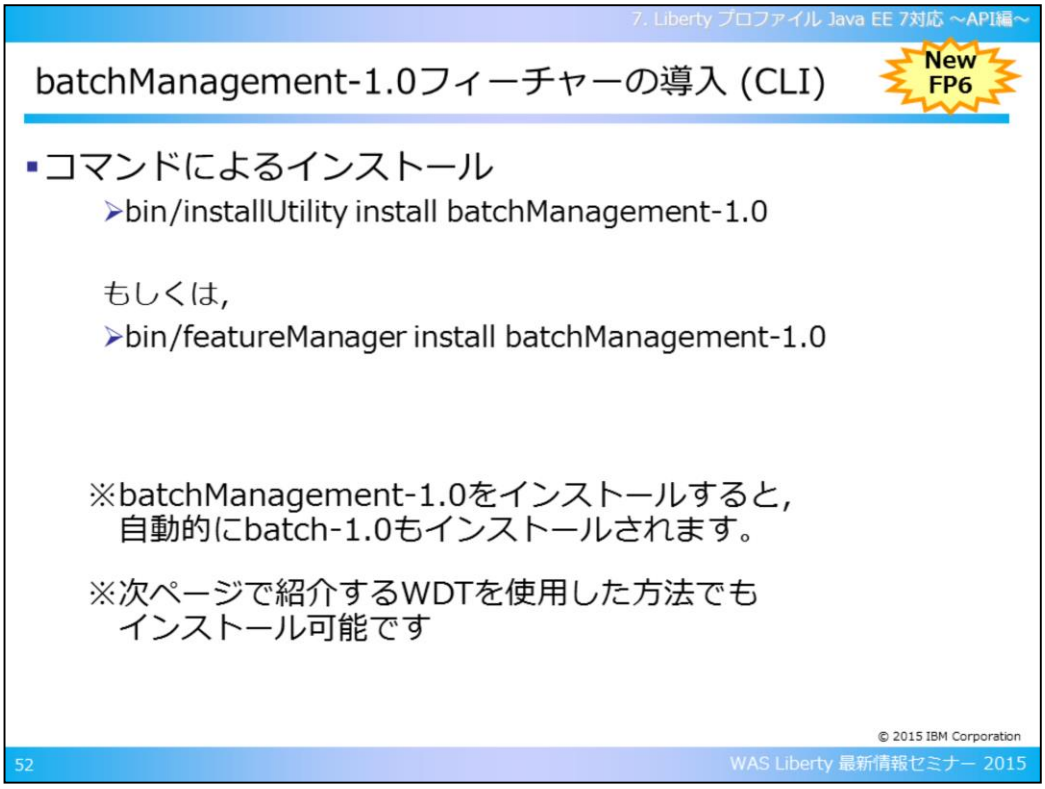

7. Liberty プロファイル Java EE 7対応 ～API編～
Liberty Profileの設定(server.xml)
New FP6
Concurrency Utilities for Java EE 機能の有効化
concurrent-1.0のFeatureを追加する
<featureManager>
<feature>concurrent-1.0</feature>
</featureManager>
ManagedExecutorServiceを構成する
<managedExecutorService jndiName="concurrent/execSvc" />
デフォルトで構成されたManagedExecutorServiceインスタンス(DefaultManagedExecutorService)を利用することも可能
アプリケーションからは java:comp/DefaultManagedExecutorServiceでlookupする
<managedScheduledExecutorService>や <managedThreadFactory>を使用する場合も同様に定義する
デフォルトで構成されたインスタンスは以下のJNDI名でlookupする
java:comp/DefaultManagedScheduledExecutorService
java:comp/DefaultManagedThreadFactory
© 2015 IBM Corporation
47
WAS Liberty 最新情報セミナー 2015

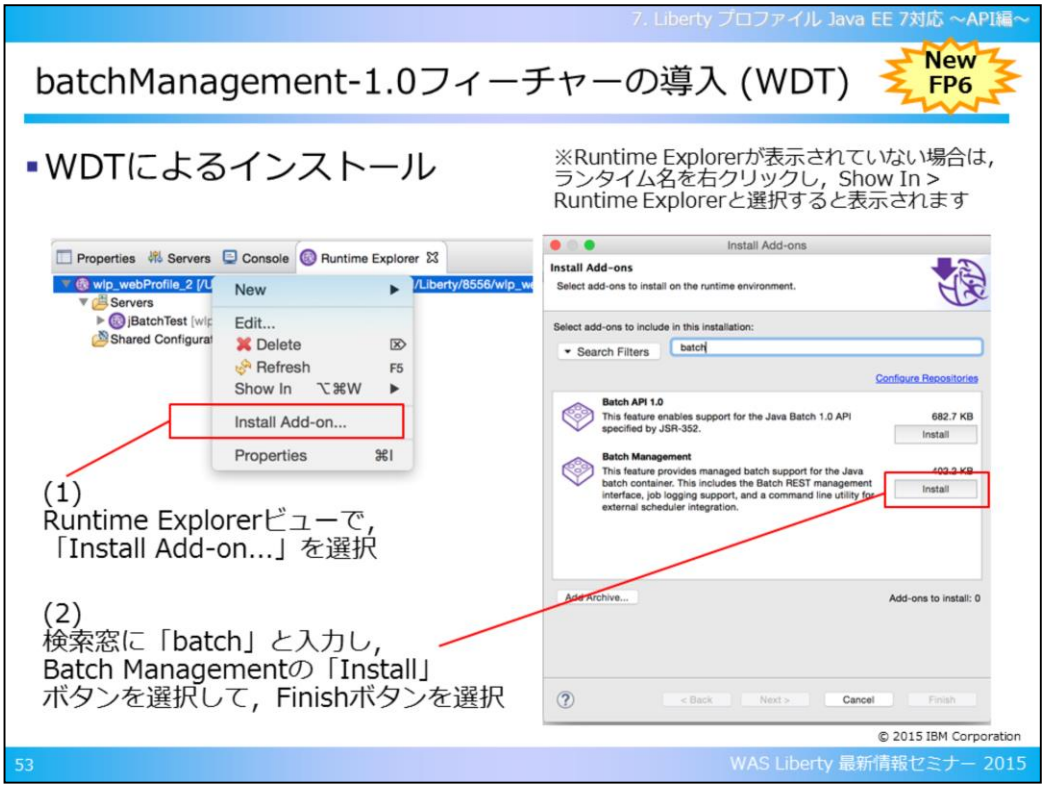

7. Liberty プロファイル Java EE 7対応 ～API編～
Liberty Profileの設定(server.xml) 続き
New FP6
タスクの実行スレッドに伝播させたいコンテキストがある場合は下記いずれかの方法でContextを指定する
<managedExecutorService jndiName="concurrent/execSvc">
<contextService>
<jeeMetadataContext/>
<classloaderContext/>
<securityContext/>
</contextService>
</managedExecutorService>
<contextService id="contextSvc">
<jeeMetadataContext/>
</contextService>
<managedExecutorService jndiName="concurrent/execSvc"
contextServiceRef="contextSvc"/>
© 2015 IBM Corporation
48
WAS Liberty 最新情報セミナー 2015

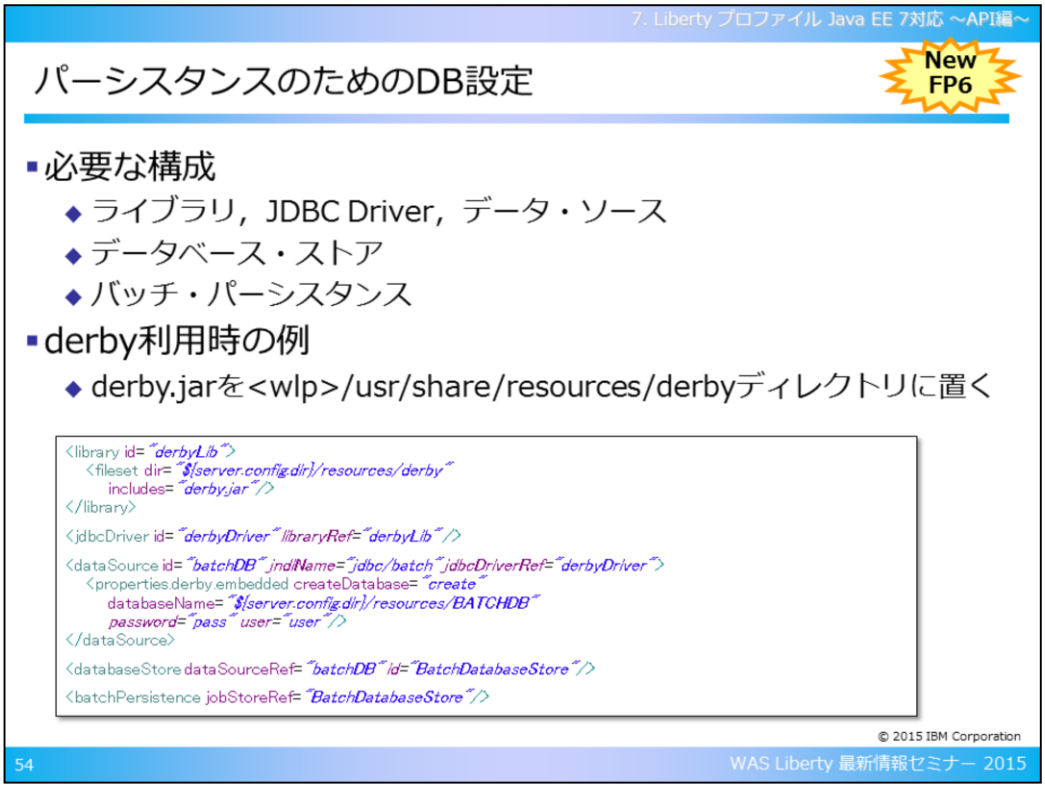

7. Liberty プロファイル Java EE 7対応 ～API編～
構成したManagedExecutorServiceの取得
New FP6
使用するアプリケーションからは，JNDI lookup
またはアノテーションによるインジェクションで取得する
@Resource(lookup="concurrent/execSvc")
ManagedExecutorService executor;
Context cx = new InitialContext();
ManagedExecutorService executor =
(ManagedExecutorService)cx.lookup("java:comp/env/concurrent/execSvc");
Java SEのConcurrency Utilityのクラスとしても利用可能
ManagedExecutorService
→ java.util.concurrent.ExecutorService
ManagedScheduledExecutorService
→ java.util.concurrent.ScheduledExecutorService
ManagedThreadFactory
→ java.util.concurrent.ThreadFactory
© 2015 IBM Corporation
49
WAS Liberty 最新情報セミナー 2015

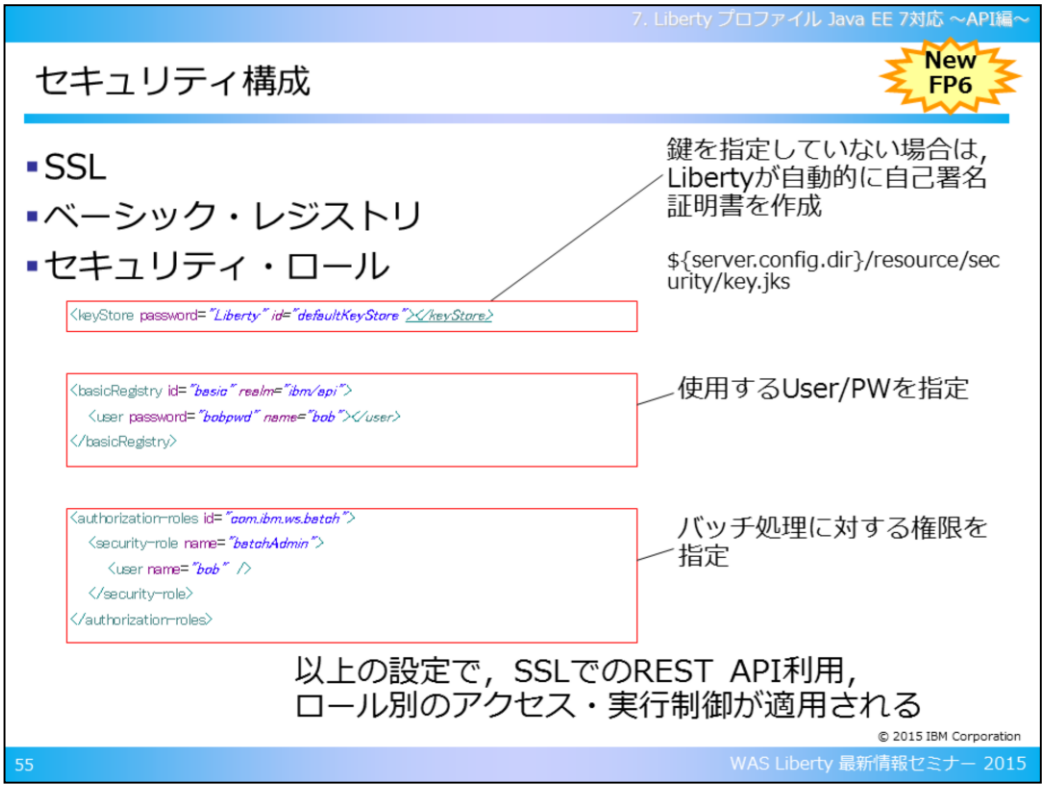

7. Liberty プロファイル Java EE 7対応 ～API編～
Batch Applications for the Java Platform (JSR352)
New FP6
Java Batchの使用にはパーシスタンスのためのDBが必要
DB2
他社DB
derby ※本番非推奨
これらの設定は，
Libertyのサーバー構成ファイル(server.xml)に設定
セキュリティ構成が必要
SSLの設定
ジョブのサブミットやスタート/ストップなどで使用するREST APIをSSL化
バッチの操作，ジョブへのアクセス・実行制御の設定
JavaBatchのロール・ベース・セキュリティ (認可ロール)
batchAdmin / batchSubmitter / batchMonitor
バッチの操作，ジョブ・インスタンスへのアクセス方法
REST API，JobOperatorインタフェース
(事前セットアップ: ユーザーレジストリ，認可ロールの設定)
© 2015 IBM Corporation
50
WAS Liberty 最新情報セミナー 2015

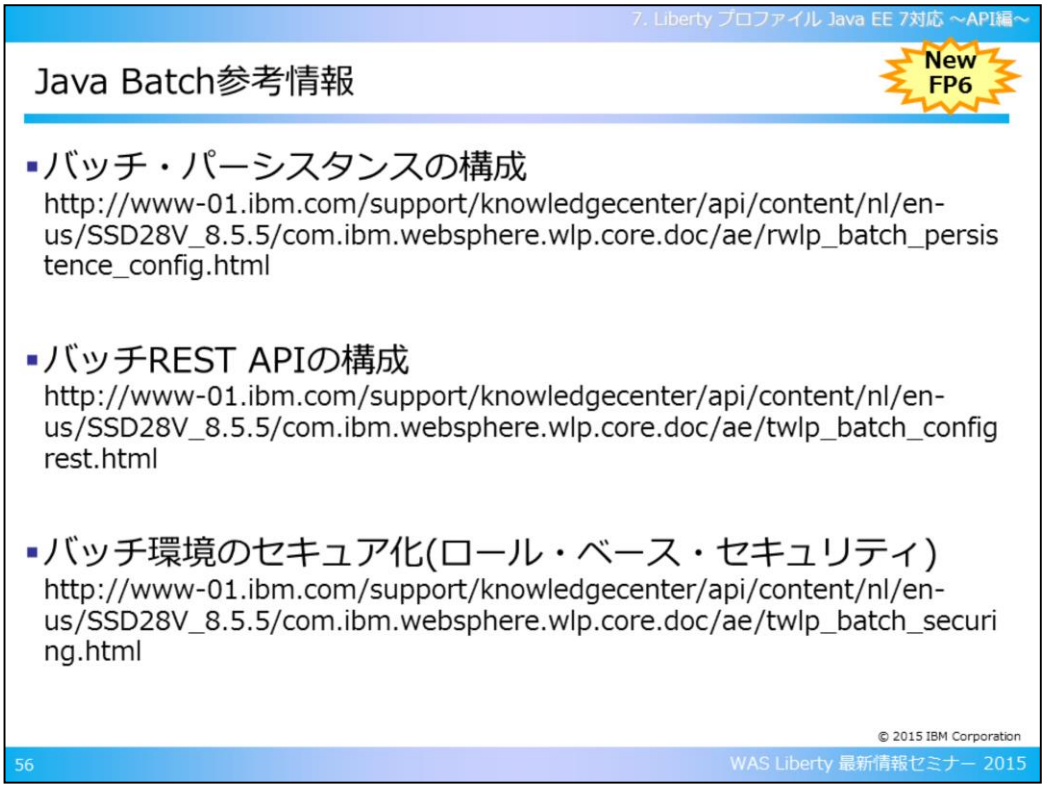

7. Liberty プロファイル Java EE 7対応 ～API編～
Java Batchを使用するためのserver.xmlの構成
New FP6
batchManagement-1.0フィーチャー以下の管理機能を提供
RESTインタフェースを提供しリモートでジョブのサブミットや操作が可能
コマンドライン・ユーティリティを提供
ジョブ・ロギングをサポート
JMS利用による複数サーバーのサポート
server.xmlに設定
<featureManager>
<feature>batchManagement-1.0</feature>
</featureManager>
batch-1.0
JavaBatch(JSR352)のAPIを提供
batchManagement-1.0を有効にすると自動でbatch-1.0も有効化
© 2015 IBM Corporation
51
WAS Liberty 最新情報セミナー 2015

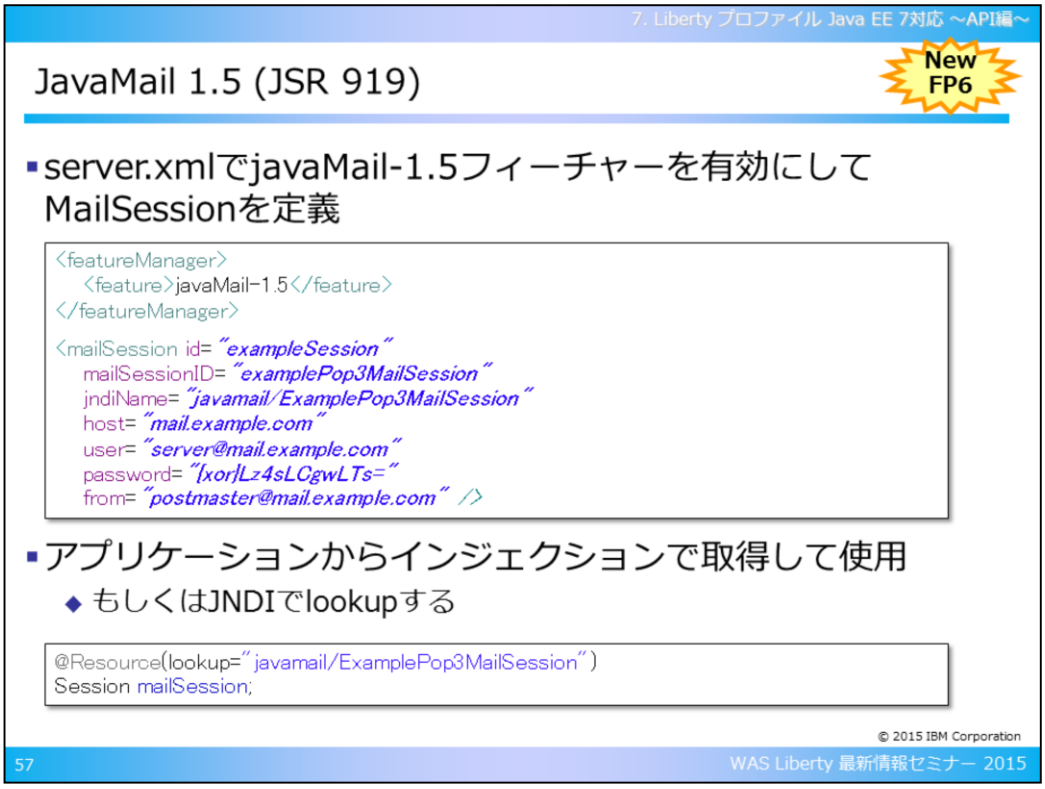

7. Liberty プロファイル Java EE 7対応 ～API編～
batchManagement-1.0フィーチャーの導入 (CLI)
New FP6
コマンドによるインストール
bin/installUtility install batchManagement-1.0
もしくは,
bin/featureManager install batchManagement-1.0
※batchManagement-1.0をインストールすると,
自動的にbatch-1.0もインストールされます。
※次ページで紹介するWDTを使用した方法でも
インストール可能です
© 2015 IBM Corporation
52
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
batchManagement-1.0フィーチャーの導入 (WDT)
New FP6
WDTによるインストール
※Runtime Explorerが表示されていない場合は，ランタイム名を右クリックし，Show In > Runtime Explorerと選択すると表示されます
Properties
Servers
Console
Runtime Explorer
New
Edit...
Delete
Refresh
F5
Show In
Install Add-on...
Properties
⌘I
(1)
Runtime Explorerビューで，「Install Add-on...」を選択
(2)
検索窓に「batch」と入力し，Batch Managementの「Install」ボタンを選択して，Finishボタンを選択
Install Add-ons
Select add-ons to install on the runtime environment.
Select add-ons to include in this installation:
Search Filters
batch
Configure Repositories
Batch API 1.0
This feature enables support for the Java Batch 1.0 API specified by JSR-352.
682.7 KB
Install
Batch Management
This feature provides managed batch support for the Java batch container. This includes the Batch REST management interface, job logging support, and a command line utility for external scheduler integration.
Install
Add Archive...
Add-ons to install: 0
< Back
Next >
Cancel
Finish
© 2015 IBM Corporation
53
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
パーシスタンスのためのDB設定
New FP6
必要な構成
ライブラリ，JDBC Driver，データ・ソース
データベース・ストア
バッチ・パーシスタンス
derby利用時の例
derby.jarを<wlp>/usr/share/resources/derbyディレクトリに置く
<library id="derbyLib">
<fileset dir="${server.config.dir}/resources/derby"
includes="derby.jar" />
</library>
<jdbcDriver id="derbyDriver" libraryRef="derbyLib" />
<dataSource id="batchDB" jndiName="jdbc/batch" jdbcDriverRef="derbyDriver">
<properties.derby.embedded createDatabase="create"
databaseName="${server.config.dir}/resources/BATCHDB"
password="pass" user="user" />
</dataSource>
<databaseStore dataSourceRef="batchDB" id="BatchDatabaseStore" />
<batchPersistence jobStoreRef="BatchDatabaseStore" />
© 2015 IBM Corporation
54
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
セキュリティ構成
New FP6
SSL
ベーシック・レジストリ
セキュリティ・ロール
鍵を指定していない場合は，Libertyが自動的に自己署名証明書を作成
${server.config.dir}/resource/security/key.jks
<keyStore password="Liberty" id="defaultKeyStore"></keyStore>
<basicRegistry id="basic" realm="ibm/api">
<user password="bobpwd" name="bob"></user>
</basicRegistry>
使用するUser/PWを指定
<authorization-roles id="com.ibm.ws.batch">
<security-role name="batchAdmin">
<user name="bob" />
</security-role>
</authorization-roles>
バッチ処理に対する権限を指定
以上の設定で，SSLでのREST API利用，
ロール別のアクセス・実行制御が適用される
© 2015 IBM Corporation
55
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
Java Batch参考情報
New FP6
バッチ・パーシスタンスの構成
http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en-us/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/rwlp_batch_persistence_config.html
バッチREST APIの構成
http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en-us/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/twlp_batch_configrest.html
バッチ環境のセキュア化(ロール・ベース・セキュリティ)
http://www-01.ibm.com/support/knowledgecenter/api/content/nl/en-us/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/twlp_batch_securing.html
© 2015 IBM Corporation
56
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
JavaMail 1.5 (JSR 919)
New FP6
server.xmlでjavaMail-1.5フィーチャーを有効にして MailSessionを定義
<featureManager>
<feature>javaMail-1.5</feature>
</featureManager>
<mailSession id="exampleSession"
mailSessionID="examplePop3MailSession"
jndiName="javamail/ExamplePop3MailSession"
host="mail.example.com"
user="server@mail.example.com"
password="{xor}Lz4sLCgwLTs="
from="postmaster@mail.example.com" />
アプリケーションからインジェクションで取得して使用
もしくはJNDIでlookupする
@Resource(lookup="javamail/ExamplePop3MailSession")
Session mailSession;
© 2015 IBM Corporation
57
WAS Liberty 最新情報セミナー 2015


7. Liberty プロファイル Java EE 7対応 ～API編～
参考：Java SE 8対応
© 2015 IBM Corporation
58
WAS Liberty 最新情報セミナー 2015

Libertyプロファイルで利用可能なJDK

- JDKつきZIPファイルにはIBM Java SDK 8が同梱
  - Windows 32bit/64bit, Linux x86 32bit/64bit, Linux PPC 32bit/64bit/64bit LE

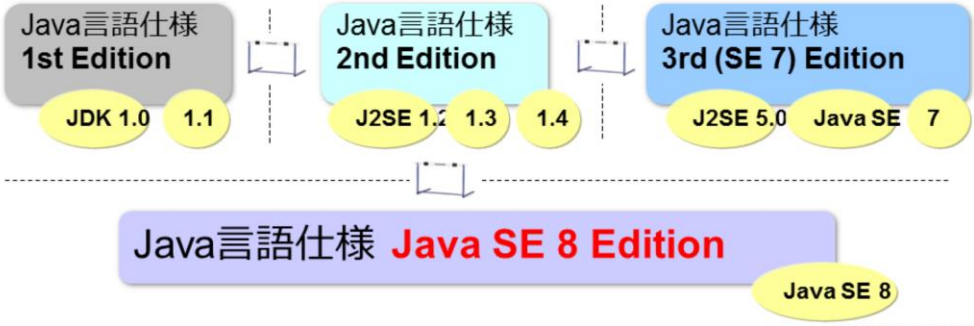


- 製品メディア/PAサイトからダウンロードしたAssemblyには IBM Java SDK 8/7が同梱
  - IIMを使用して導入が可能
  - Libertyプロファイルと同時にしか導入できない（任意のディレクトリには導入できない）
- Java SE 6/7/8仕様に準拠した任意のJRE/JDKが利用可能
  - Oracle JDK
  - OpenJDK

© 2015 IBM Corporation

Java SE 8における言語仕様の変革

- 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



© 2015 IBM Corporation

Project Lambda

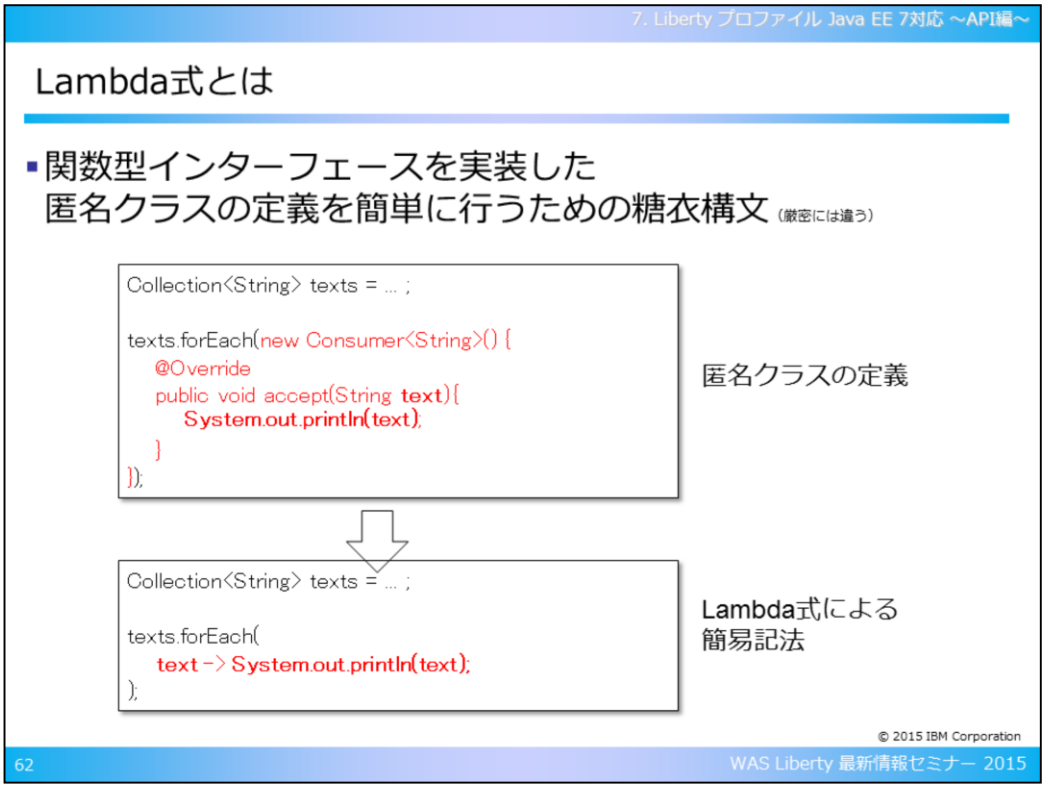
- Lambda式
- 型推論
- Method Reference
- Default Method
- Stream API



```
( 引数 ) -> { 処理 }
```

Lambda式とは，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」

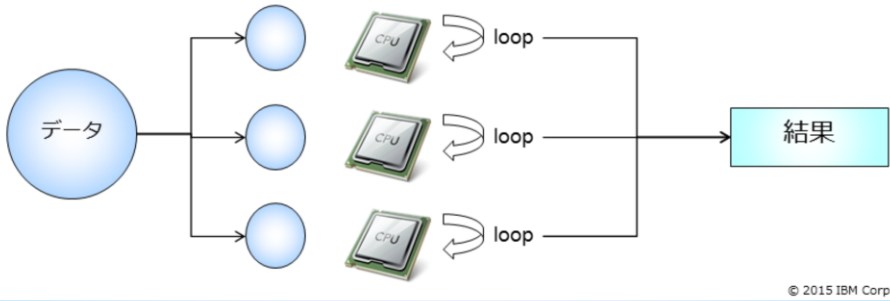
© 2015 IBM Corporation


7. Liberty プロファイル Java EE 7対応 ~API編~
Lambda式とは
関数型インターフェースを実装した
匿名クラスの定義を簡単に行うための糖衣構文 (厳密には違う)
Collection<String> texts = ... ;
texts.forEach(new Consumer<String>() {
@Override
public void accept(String text){
System.out.println(text);
}
});
匿名クラスの定義
Collection<String> texts = ... ;
texts.forEach(
text -> System.out.println(text);
);
Lambda式による
簡易記法
© 2015 IBM Corporation
62
WAS Liberty 最新情報セミナー 2015

外部イテレーターとStreamによる並行処理

- Lambda式とStream APIを利用すると
  簡単に並列処理を記述できる

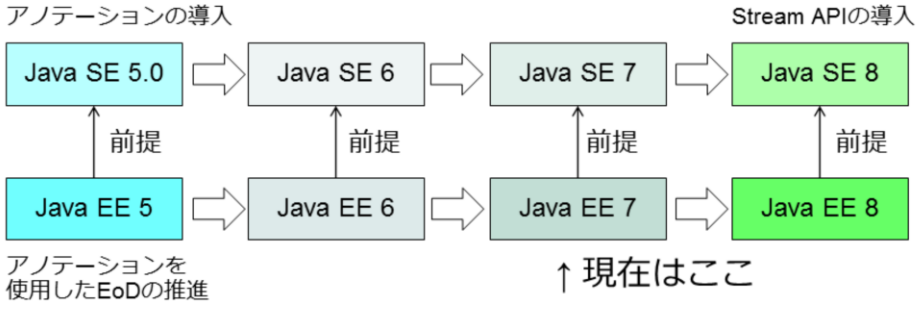
```
// 最もSalaryの高い非Managerをかえす
public static Person getHighestNonManagerEarner(List<Person> list) {
    return list.stream().parallel()
                .filter(p -> !p.isManager())
                .reduce(dummy, (p1, p2) -> (p1.getSalary() > p2.getSalary())? p1 : p2);
}
```



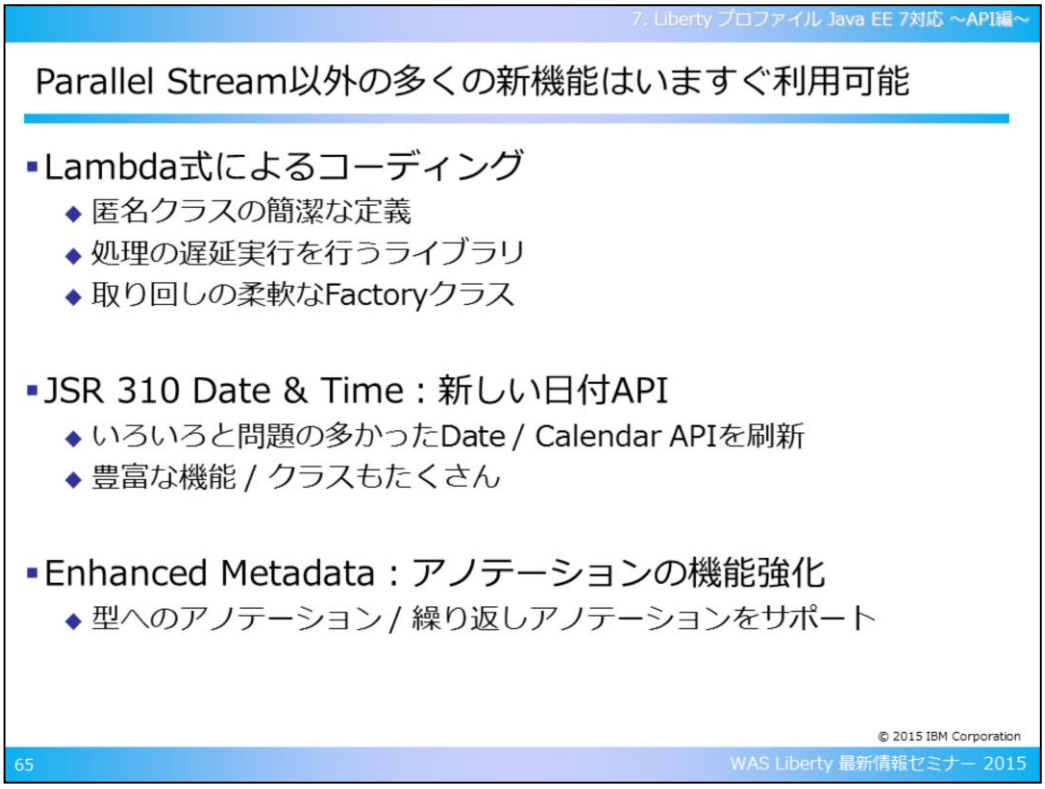
© 2015 IBM Corporation

Java SEとJava EEの関係

- Java EE環境で並行実行を利用するには，もう少しかかる
  - Stream APIが内部で使用するスレッドの起動が，Java EEコンテナでは問題を起こす
  - 将来のJava SE/EEでの対応を待つ必要がある（Java EE Concurrency Utilitiesで対応される予定）



© 2015 IBM Corporation


7. Liberty プロファイル Java EE 7対応 ～API編～
Parallel Stream以外の多くの新機能はいますぐ利用可能
Lambda式によるコーディング
匿名クラスの簡潔な定義
処理の遅延実行を行うライブラリ
取り回しの柔軟なFactoryクラス
JSR 310 Date & Time：新しい日付API
いろいろと問題の多かったDate / Calendar APIを刷新
豊富な機能 / クラスもたくさん
Enhanced Metadata：アノテーションの機能強化
型へのアノテーション / 繰り返しアノテーションをサポート
© 2015 IBM Corporation
65
WAS Liberty 最新情報セミナー 2015

まとめ

- Java EE 7仕様は，モダンなアプリケーションの実装にも対応できる優れた仕様です
- J2EE仕様/古いフレームワークで実装されたアプリケーションを再設計するベストなタイミングです
- LibertyプロファイルはJava EE 7/Java SE 8に対応し，最新のプログラミングモデルによるアプリケーション実装をサポートします

© 2015 IBM Corporation


7. Liberty プロファイル Java EE 7対応 ～API編～
THANK
YOU
© 2015 IBM Corporation
67
WAS Liberty 最新情報セミナー 2015