

Libertyプロファイル構築・運用

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
Webプラットフォーム
蜂谷 美穂

Agenda

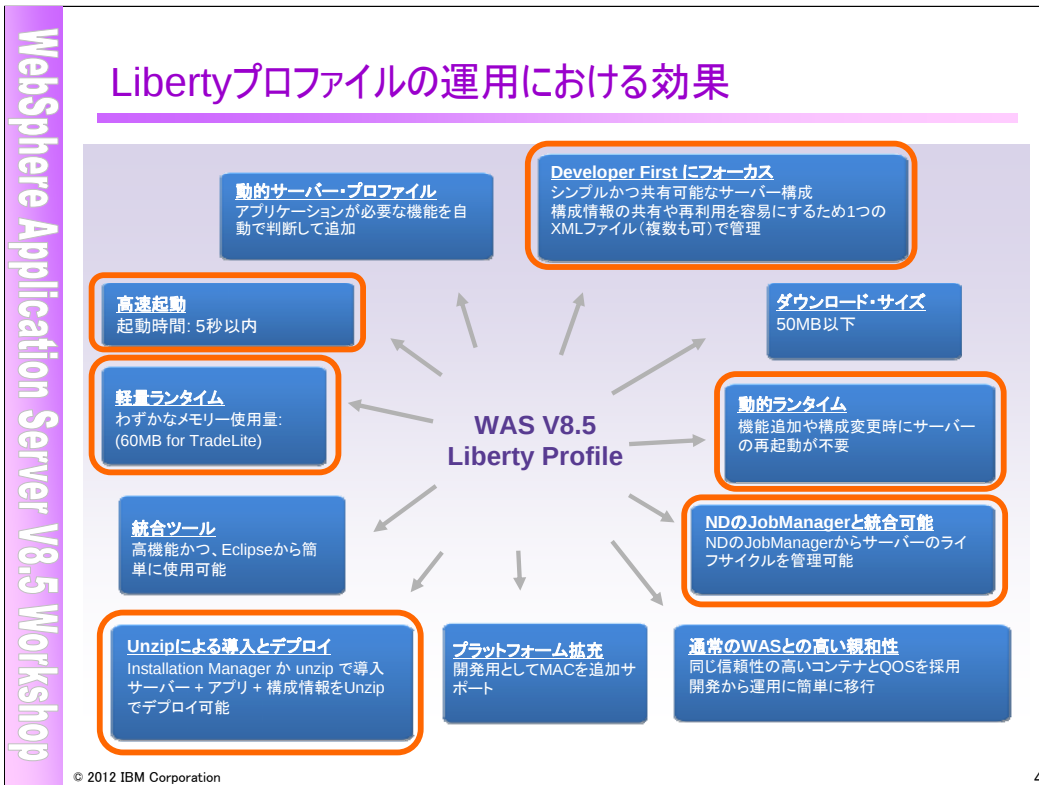
1. 概要
2. トポロジー
3. 導入
4. 構成
5. 運用管理

まとめ

参考文献

1. 概要

Libertyプロファイルの運用における効果
Libertyプロファイルの性能評価結果



Libertyプロファイルは、主に開発中のテストに使用されることを想定したWebアプリケーション実行環境ですが、要件によっては本番環境で使用することも可能です。そして、本番環境での構築や運用といった局面においてもさまざまな効果があります。

「高速起動」: Libertyプロファイルを使用してサーバーの起動時間が短縮されることにより、本番環境におけるメンテナンス時間を極力少なくすること、あるいはメンテナンス時間内でサーバー再起動以外のメンテナンス作業に多くの時間を割り当てることができます。

「軽量ランタイム」: Libertyプロファイルはアプリケーションの実行に必要な実行環境のみを提供するため、メモリー使用量を最小限に抑えることができ、サーバー上の限られたリソースを有効に利用することができます。

「Unzipによる導入とデプロイ」: Libertyプロファイルは、ランタイム環境やサーバー構成、アプリケーション情報などを全てまとめてzip形式にパッケージングすることができ、それをUnzip展開するだけで別ノードへそのまま導入することができるため、本番環境の構築や移行を容易に短時間で行うことができます。

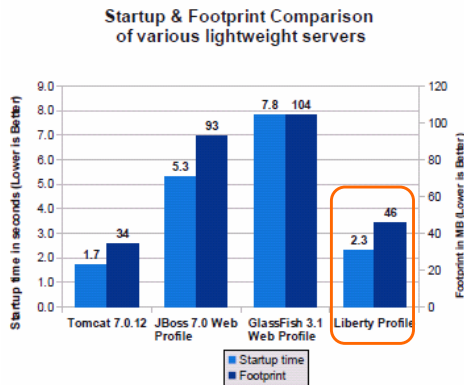
「Developer First にフォーカス」: Libertyプロファイルの構成情報は1つ(複数可)のXMLファイルで管理されているため、構成変更時の前後比較が容易になったり、管理対象となる構成ファイルが少なくなるなど、運用管理の負荷を軽減させることができます。

「動的ランタイム」: Libertyプロファイルに対する機能追加や構成情報の変更にはサーバーの再起動が不要であるため、本番環境においてサービスを停止させることなく構成変更作業を行うことができます。

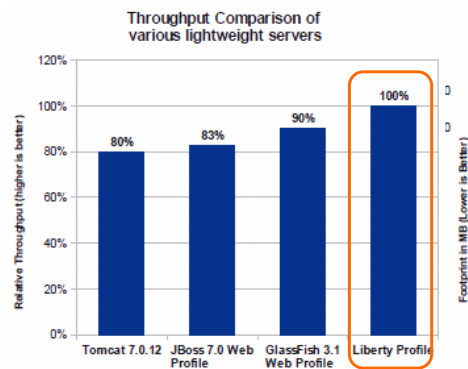
「NDのJobManagerと統合可能」: LibertyプロファイルをJobManagerに登録すれば管理対象とすることができるため、Libertyプロファイルへの作業をジョブとして実行したり、JobManagerにおいてLibertyプロファイルのサーバーのライフサイクルを管理することができます。

Libertyプロファイルの性能評価結果

起動時間とメモリ使用量の比較



スループットの比較



「WAS V8.5 Performance Report Version 1.0」(2012年8月13日作成)より抜粋

© 2012 IBM Corporation

5

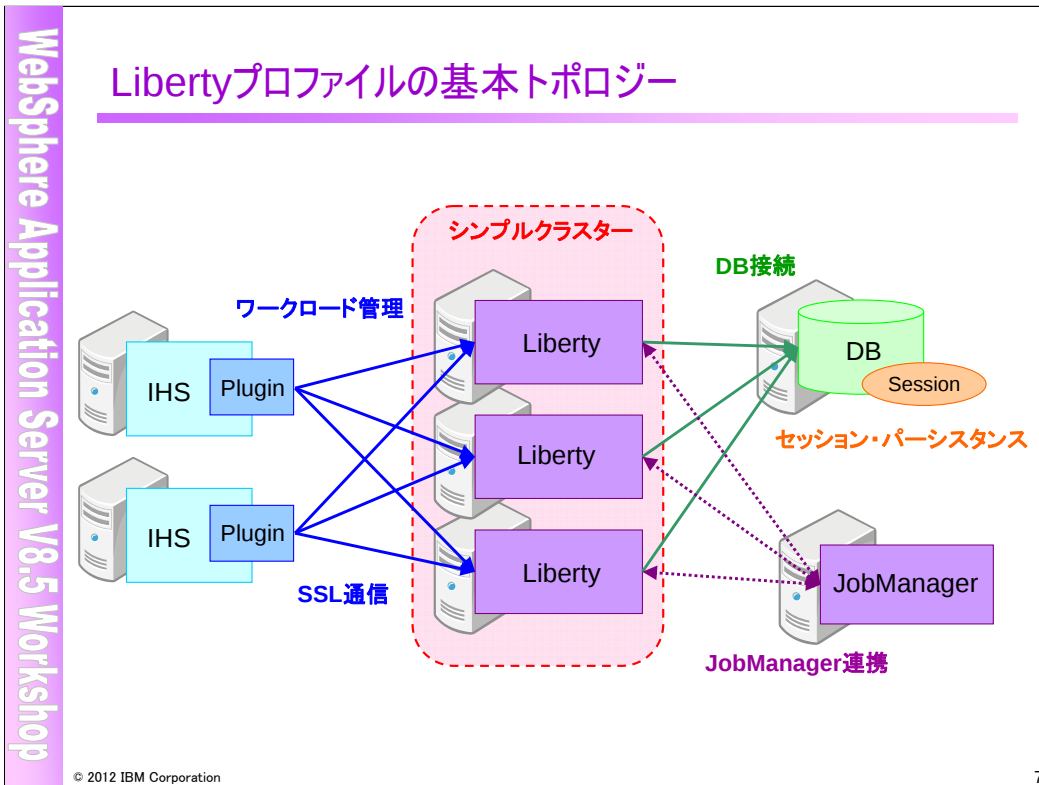
これらのグラフは、Libertyプロファイルと同等な機能を提供するLightweightなWebアプリケーション実行環境とLibertyプロファイルの性能を評価した結果です。

左は起動時間とメモリ使用量を比較したグラフで、薄い青色の棒グラフが起動時間、濃い青色の棒グラフがメモリ使用量を表しています。

右はスループットを比較したグラフで、Libertyプロファイルのスループットを100%とした場合の割合を表しています。

2. トポロジー

Libertyプロファイルの基本トポロジー
JobManagerによる統合管理



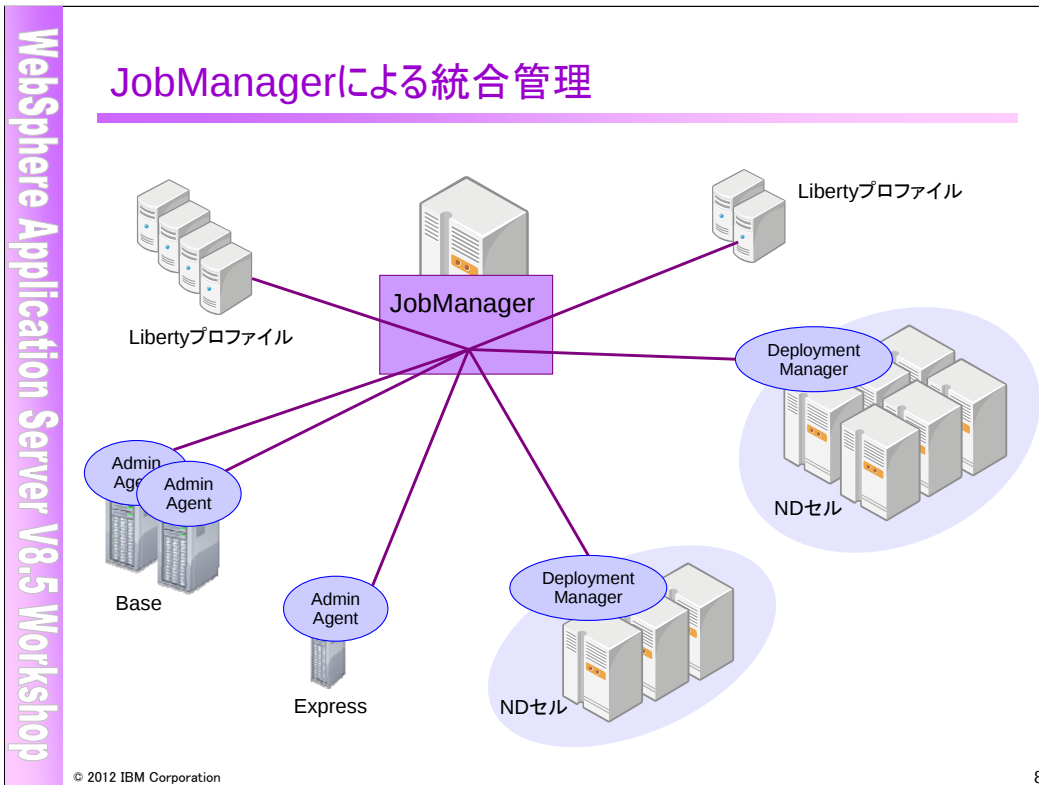
Libertyプロファイルの基本トポロジーは、フルプロファイルの基本トポロジーとほぼ同じです。

Libertyプロファイルに関しても、同じアプリケーションを複数のノードや複数のサーバー上で稼働させるために並列に配置することができます。ただし、WASのNDエディションで提供されている高機能クラスターとは異なり、Libertyプロファイルでは、ExpressエディションやBaseエディションでもサポートされているシンプルクラスターを構成することになります。

Libertyプロファイルの前段にIHSおよびプラグインを配置することで、IHSから複数のLibertyプロファイルへのリクエストの割り振りを行うことができます。IHSからLibertyプロファイルへの割り振りでは、セッション・アフィニティやサーバー・フェールオーバーなどのワークロード管理を行うこともでき、SSLを使用した通信を行うこともできます。

Libertyプロファイルから後段のDBサーバーに対して、フルプロファイルと同様にデータソースを使用した接続を行うこともできます。また、セッション情報の永続化のためにセッション・パーシスタンスの構成もサポートしています。

運用管理面では、複数のLibertyプロファイルをJobManagerから統合的に管理し、ジョブを実行することができます。JobManagerと連携したトポロジーについては、次ページをご参照ください。



JobManagerを使用すると、フルプロファイルのNDセルやExpress/Baseエディションのサーバーを統合管理するのと同様に、複数のLibertyプロファイルも統合的に集中管理することができます。これにより、Libertyプロファイルのサーバーに対する運用作業も個々に行うのではなく、フルプロファイルの他サーバーと同様の方法で、ジョブとしてまとめて行うことができます。

3. 導入

導入前提

パッケージ機能によるインストール

JobManagerによるインストール

パッケージ機能を利用したインストール・パターン

Libertyプロファイル導入全体の流れ

導入前提

サポートしているJDK

Java6以上であればどのJDKもサポート

ただし...

最小サポート・レベル

IBM JDK : 6.0.9 and 626 SR1
Oracle JDK : Java6 SR26

分散プラットフォーム

32bit/64bitのJava共にサポート

z/OSプラットフォーム

64bitのJavaのみサポート

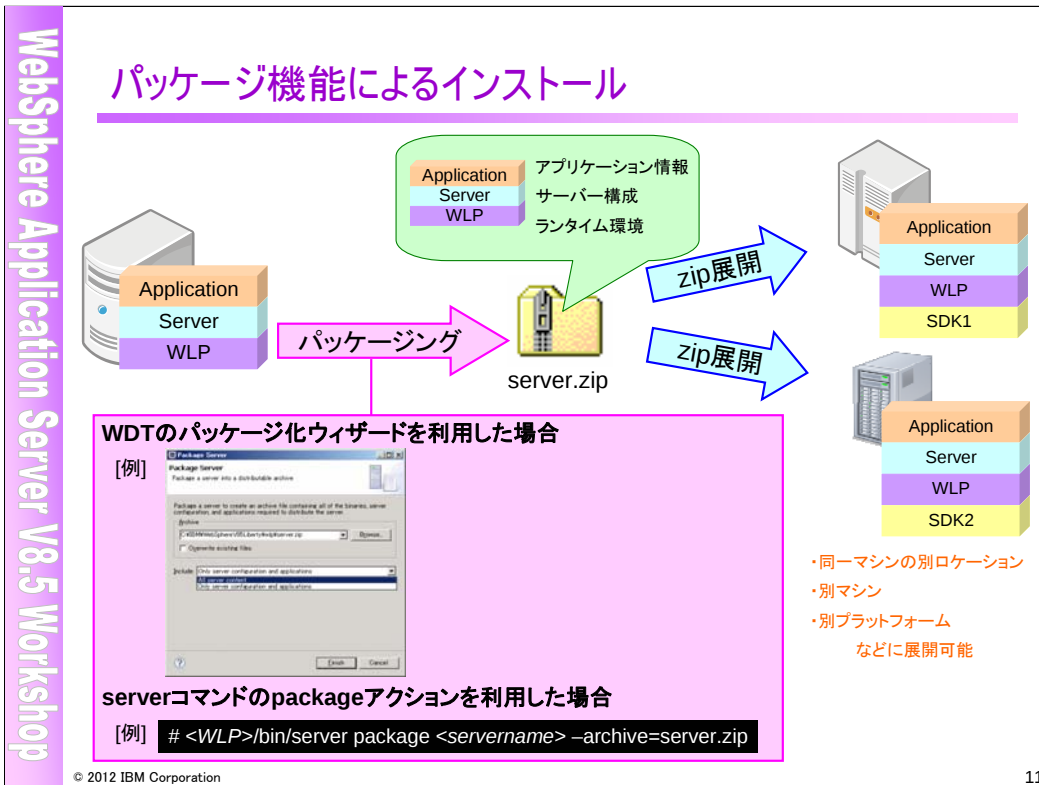
インストール方法

- IIM (IBM Installation Manager) ※「Libertyプロファイル概要・開発」セッション参照
- 自己展開圧縮ファイル(Jarファイル)実行
- WDT (WebSphere Application Server Developer Tools) からダウンロード

- **パッケージ機能によるインストール** ※当セッション参照
ランタイム環境や構成情報を含んだ圧縮ファイルの作成と別ノードでの展開
- **JobManagerによるインストール**
Libertyプロファイル・リソースの作成とジョブでのインストール実行

Libertyプロファイルを稼働させるためのJDKは、Java6以上であればどの種類のJDKでもサポートされます。ただし、いくつかの制約があります。JDKの種類によっては最小サポート・レベルが決められており、IBM JDK は6.0.9 and 626 SR1、Oracle JDK はJava6 SR26 となっています。また、分散プラットフォームでは32bitと64bitのJavaを共にサポートしていますが、z/OSプラットフォームでは64bitのJavaのみをサポートしています。

Libertyプロファイルのインストール方法として、「Libertyプロファイル概要・開発」セッションでは、IIM (IBM Installation Manager) でのインストール、自己展開圧縮ファイル(Jarファイル)を実行することによるインストール、WDT (WebSphere Application Server Developer Tools) からダウンロードすることによるインストールの3種類を紹介しました。当セッションでは、Libertyプロファイルの新規インストールというよりも、一度構築した環境を別のロケーションに導入/移行する場合などに利用できるパッケージ機能によるインストールとJobManagerによるインストールの2種類を紹介します。

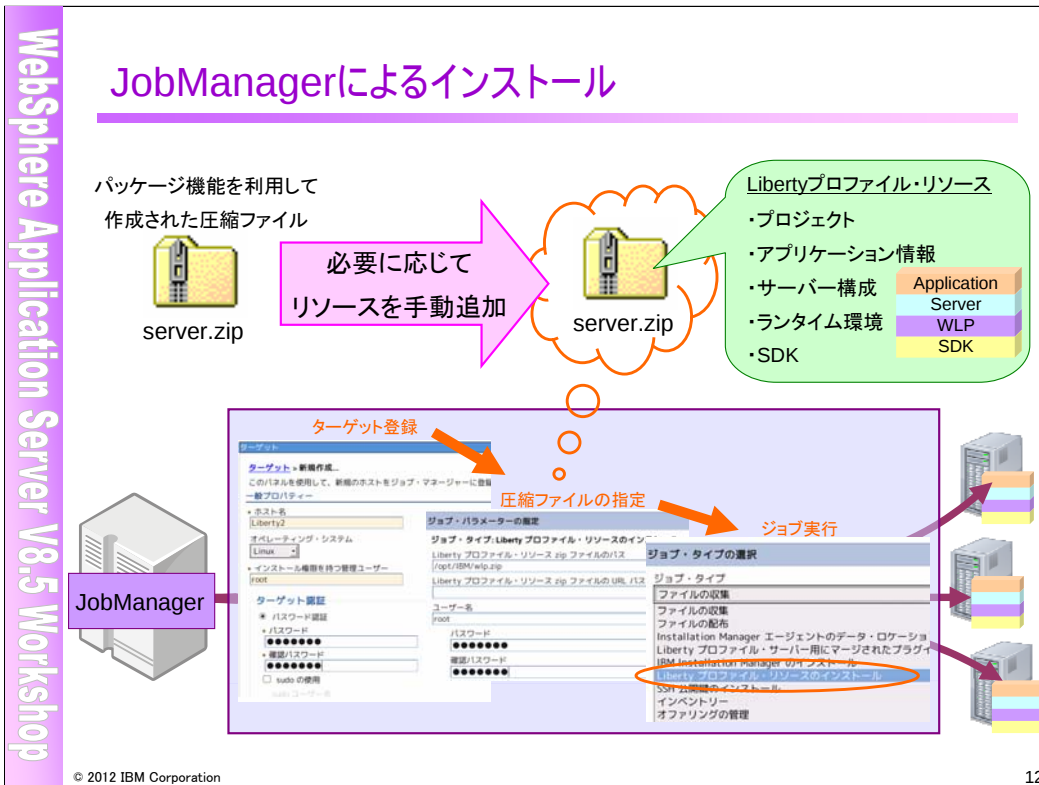


11

パッケージ機能によるインストールとは、一度構築したランタイム環境、サーバー構成、アプリケーション情報を全て1つのzip形式の圧縮ファイルにパッケージングし、それをzip展開することによってLibertyプロファイルを導入するという方法です。

パッケージングを行う機能はLibertyプロファイルが提供しており、WDTのパッケージ化ウィザードを利用したGUIベースで実行することも、serverコマンドのpackageアクションを利用したコマンドベースで実行することもできます。

zip展開先としては、パッケージングを行った同一マシンの別ロケーションでも、別マシン上でも、別プラットフォームのマシン上でも可能です。別プラットフォームのマシンにzip展開する場合、ディレクトリ名やファイル名の大きい文字小文字区別の可否などに関しては考慮する必要があります。

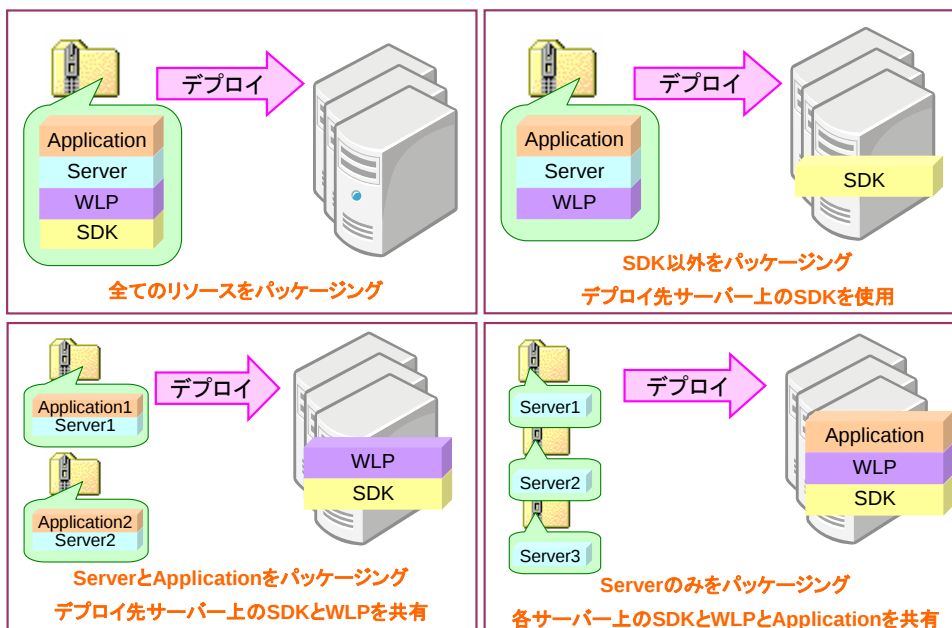


JobManagerによるインストールでは、事前にLibertyプロファイル・リソースと呼ばれるzip形式の圧縮ファイルを作成しておき、それをターゲットとして登録したサーバーに対してジョブとして実行することで、ファイルが展開され、インストールが行われます。

Libertyプロファイル・リソースとは、プロジェクト、アプリケーション情報、サーバー構成、ランタイム環境、SDKの情報がパス名やグループ化などルールに則ったディレクトリ構造で配置されたzipファイルです。手動で新規にzipファイルを作成して必要なリソースを追加していくことも可能ですが、パッケージ機能を利用してzipファイルを作成し、そこに必要に応じてリソースを追加していくことでLibertyプロファイル・リソースを作成することもできます。

Libertyプロファイル・リソースの中のプロジェクトとは、リソースのコンテナ（オプション）です。関連するリソースを同じプロジェクトの下でグループ化することによって、管理を容易にし、他のプロジェクトからのリソースとの名前の競合を避けることができます。また、Libertyプロファイル・リソースの中には、Libertyプロファイルのサーバーを実行するためのSDKも追加することができます。

パッケージ機能を利用したインストール・パターン



© 2012 IBM Corporation

13

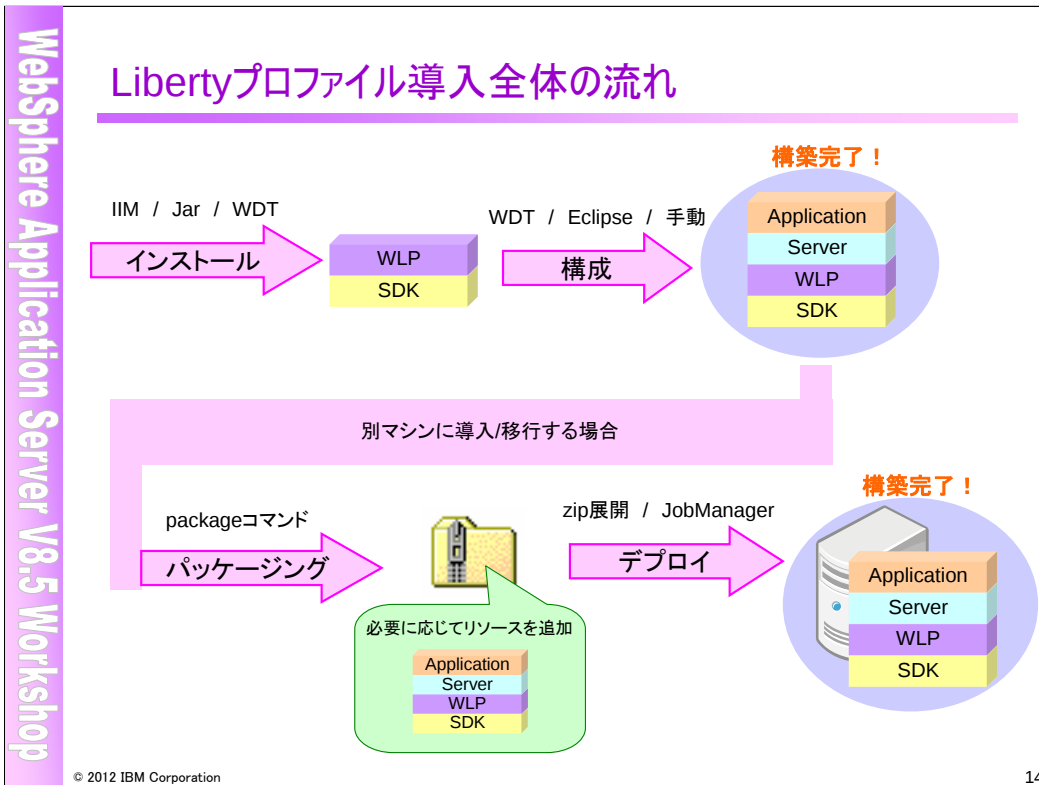
パッケージ機能を利用したインストール・パターンとして、ここでは4つのパターンを紹介します。

1つ目は、全てのリソースをパッケージングし、デプロイ先となるサーバー上で展開してそのまま使用するというパターンです。デプロイ先のサーバーには特に準備や設定は不要です。

2つ目は、SDK以外のリソースをパッケージングし、デプロイ先となるサーバー上のSDKを使用するというパターンです。デプロイ先のサーバーにSDKだけ準備されていれば、Libertyプロファイルのインストール作業やサーバーの構成、アプリケーションのデプロイといった作業は不要です。

3つ目は、サーバー構成とアプリケーション情報をパッケージングし、デプロイ先となるサーバー上のSDKとLibertyプロファイルを使用するというパターンです。デプロイ先のサーバーにSDKとLibertyプロファイルが準備されていれば、さまざまなサーバー構成とアプリケーション情報の組み合わせをデプロイし、SDKとLibertyプロファイルを共有することができます。

4つ目は、サーバー構成のみをパッケージングし、デプロイ先となるサーバー上のSDKとLibertyプロファイルとアプリケーション情報を使用するというパターンです。デプロイ先のサーバーにSDKとLibertyプロファイルとアプリケーション情報が準備されていれば、さまざまなサーバー構成をデプロイしてそれらを共有することができます。



14

Libertyプロファイルを本番環境に導入し、構築する場合の全体の流れをまとめます。

まず初めに、実行環境となるSDKを準備し、IIM/Jar/WDIいずれかの方法によりLibertyプロファイルをインストールします。次に、WDT/Eclipse/手動いずれかの方法によりアプリケーションの実行に必要なサーバーの構成を行い、アプリケーションのデプロイを行います。これで、Libertyプロファイルの導入と構築は完了です。

この構成をそのまま別マシンに導入/移行する必要がある場合は、パッケージ機能によるzipファイルの作成および必要に応じてリソースの追加を行い、それをzip展開/JobManagerいずれかの方法によりターゲットとなるサーバー上に導入します。これで、別マシンへのLibertyプロファイルの導入と構築は完了です。

例えば、開発環境でLibertyプロファイルの環境を全て構築した後に、パッケージ機能を利用して本番環境へ移行すれば、新たに本番環境においてインストールや構成の作業を行う必要がなく、開発環境との設定の差異なども発生することはありません。

4. 構成

構成方法

ランタイムに関する主なフィーチャーおよび構成エレメント

構成例

IHS/プラグイン構成

シンプルクラスター構成

セッション・パーシスタンス構成

セキュリティ構成

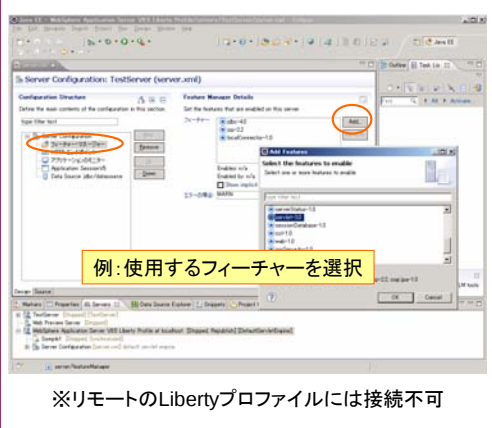
WebSphere Application Server V8.5 Workshop

構成方法

WDTあるいは手動いずれかの方法で構成

WDT

GUIベースでの構成
→ 構成ファイルserver.xmlに反映



※リモートのLibertyプロファイルには接続不可

手動

手動での構成
→ 構成ファイルserver.xml直接編集

server.xml

```
<featureManager>
  <feature>jsp-2.2</feature>
  <feature>localConnector-1.0</feature>
  <feature>例: 使用するフィーチャーを追記</feature>
</featureManager>
```

他にも必要に応じて構成ファイルを追加可能

server.env

環境変数のカスタマイズ
(JRE等を指定)

jvm.options

JVMオプションの
カスタマイズ
(ヒープサイズ等を指定)

bootstrap.properties

ランタイム環境の
初期化に使用

© 2012 IBM Corporation

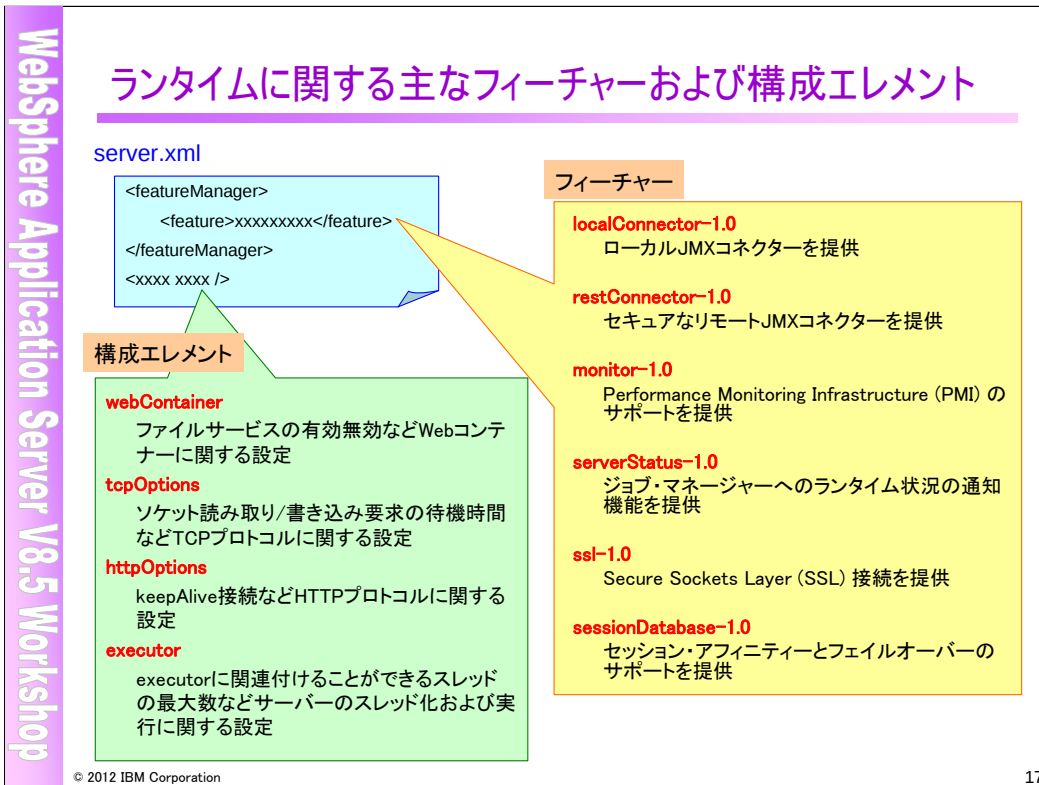
16

Libertyプロファイルの構成や設定は、WDTあるいは手動いずれかの方法で行うことができます。

WDTを使用して構成する場合、GUIベースでの作業になり、使用するフィーチャーや設定項目を選択したり入力したりすることにより、それが構成ファイルserver.xmlに反映されます。WDTはリモートのLibertyプロファイルに接続することができませんので、Libertyプロファイルを導入したマシン上でWDT/EclipseなどGUIベースでの作業が行えない環境では使用することができません。ただ、そのような環境の場合は、WDTが使用できる環境でLibertyプロファイルの構成や設定を行った上で、それらをパッケージ機能を利用してターゲットとなるサーバー上に移行するなどに対応することができます。

手動で構成する場合、構成ファイルserver.xmlを直接編集することになりますので、使用するフィーチャーや設定項目を全て構成エレメントのルールに従って記載することになります。Libertyプロファイルの構成ファイルとしては、server.xmlの1つだけでなく、必要に応じてファイルを作成して追加することができます。追加できるファイルとしては、環境変数のカスタマイズを行うためのserver.env、JVMオプションのカスタマイズを行うためのjvm.options、ランタイム環境の初期化に使用されるbootstrap.propertiesなどがあります。

16



ランタイムに関する主なフィーチャーと構成エレメントをいくつか紹介します。

有効化したいフィーチャーは、server.xml上で<featureManager>エレメントのサブエレメントである<feature>内に記載します。ランタイムに関するフィーチャーの種類としては、localConnector-1.0フィーチャー、restConnector-1.0フィーチャー、monitor-1.0フィーチャー、serverStatus-1.0フィーチャー、ssl-1.0フィーチャー、sessionDatabase-1.0フィーチャーなどがあります。フィーチャーの中には、ある特定のフィーチャーを有効にすると、依存関係のある関連した他のフィーチャーも有効になるというケースもあります。例えば、restConnector-1.0フィーチャーを有効にした場合、製品内部の依存関係により、jaxrs-1.0フィーチャーやservlet-3.0フィーチャー、ssl-1.0フィーチャーも有効になります。このような場合、フィーチャーを削除する際に、意図しない機能まで無効化してしまう可能性があるため注意が必要です。

デフォルトから変更したい設定は、server.xml上に構成エレメントを記載します。構成エレメントの種類は多数あり、その属性やサブエレメントなどについても、InfoCenterにまとめられています。ここでは、構成エレメントとして4種類紹介します。<webContainer>は、ファイルサービスの有効無効などWebコンテナに関する設定を行います。<tcpOptions>は、ソケットで読み取り要求または書き込み要求が完了するのを待機する時間などTCPプロトコルに関する設定を行います。<httpOptions>は、keepAlive接続などHTTPプロトコルに関する設定を行います。<executor>は、executorに関連付けることができるスレッドの最大数などサーバーのスレッド化および実行に関する設定を行います。

参考:

WebSphere Application Server Liberty profile feature dependencies might be different in the developer tools and server runtime environment

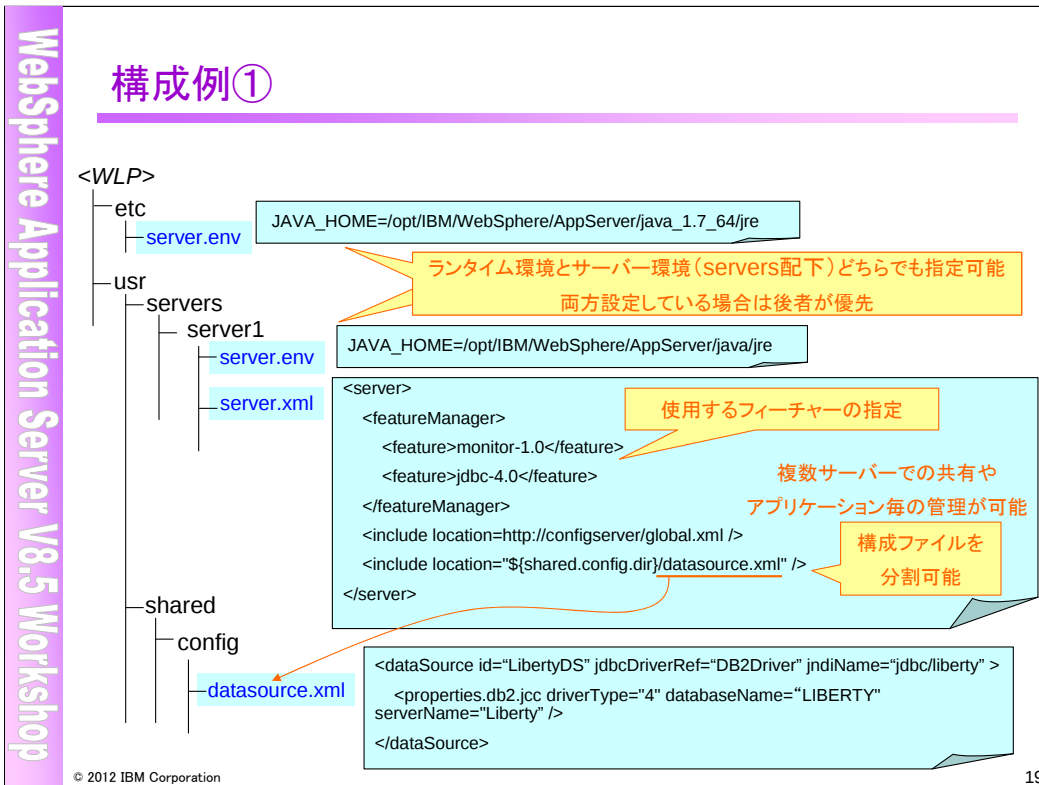
<http://www-01.ibm.com/support/docview.wss?uid=swg21593248>

WAS V8.5 Information Center 「Liberty プロファイル: server.xml ファイルの構成エレメント」

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.wlp.nd.multipatform.doc/autodita/rwlp_metatype_4ic.html

【参考】各フィーチャーの詳細

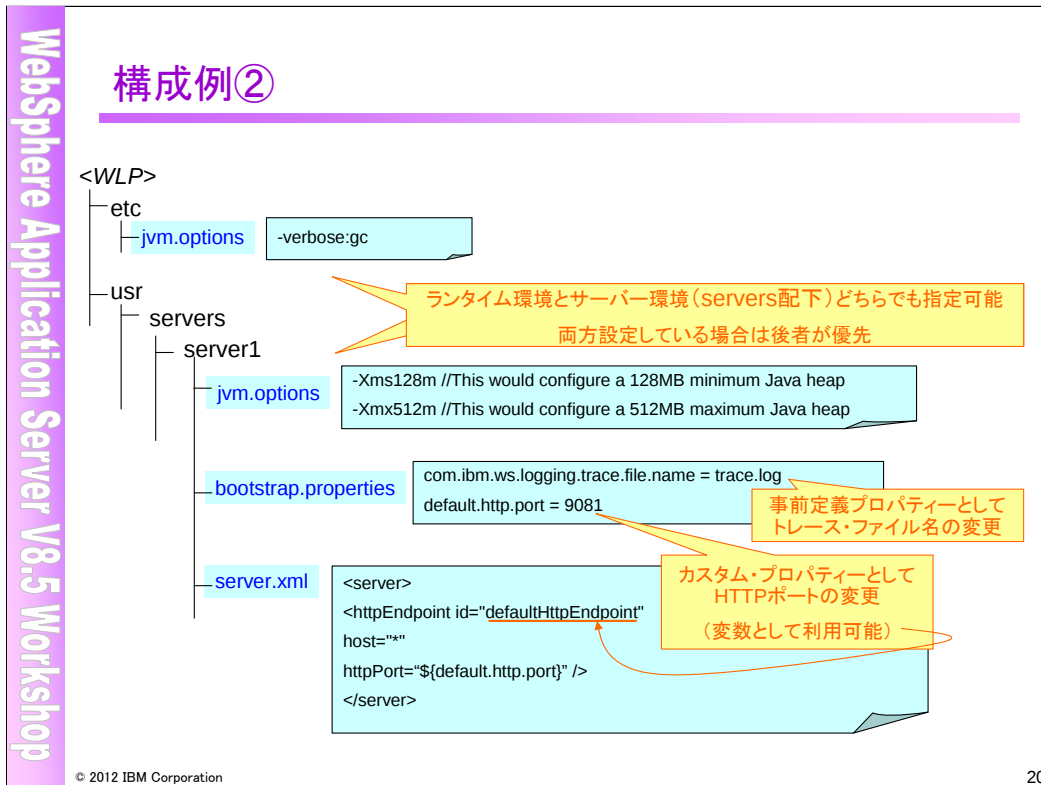
- localConnector-1.0 : ローカルJMXコネクター
 - ◆ ローカルJMXコネクターを提供
 - ◆ 同一ホスト・マシン上で同一ユーザー ID、同一 JDK で実行しているユーザーのみが使用可能
 - ◆ Jconsole などの JMX クライアントやAttach API を使用するその他のJMXクライアントによるローカル・アクセスが可能
- restConnector-1.0 : リモートJMXコネクター
 - ◆ セキュアなリモートJMXコネクターを提供
 - ◆ 任意の JDK を使用してローカルまたはリモートで使用可能
 - ◆ REST ベース・コネクター (RESTプロトコル) 経由の JMX クライアントによるリモート・アクセスが可能
 - ◆ SSL および基本ユーザー・セキュリティ構成が必要 (ssl-1.0フィーチャー組み込み済)
- monitor-1.0 : モニター
 - ◆ LibertyプロファイルでのPerformance Monitoring Infrastructure (PMI) のサポートを提供
- serverStatus-1.0 : サーバー状況
 - ◆ ジョブ構成でリソースとして認識されているデプロイメント・マネージャーおよびジョブ・マネージャーにその状況を自動的にパブリッシュ可能
 - ◆ 認識される状態は Started と Stopped
- ssl-1.0 : SSL
 - ◆ Secure Sockets Layer (SSL) 接続をサポート
 - ◆ セキュア HTTPS リスナーを使用する場合は必ず有効化
 - ◆ 旧バージョンのWASで提供されていたものと同じダミーの鍵ストアとダミーのトラストストアを提供
- sessionDatabase-1.0 : セッション・パーシスタンス
 - ◆ Libertyプロファイルでセッション・アフィニティとフェイルオーバーのサポートを提供



構成例の1つ目を紹介します。

環境変数のカスタマイズを行うserver.envでは、Libertyプロファイルの稼働環境となるJREを指定することができます。server.envは、ランタイム環境としてLibertyプロファイルのetcディレクトリ配下にも、サーバー環境としてLibertyプロファイルのusrディレクトリの各サーバーディレクトリ配下にも配置することができます。両方に配置して同じ内容を設定している場合は、サーバー環境の設定が優先されます。

構成情報を記載するserver.xmlでは、使用するフィーチャーの指定や各種設定を行います。全ての構成を1つのserver.xml内に記載することもできますが、<include>エレメントを使用することにより、構成ファイルを分割して他のディレクトリやWebサーバーなど外部に配置しておくことができます。例えば、特定のサーバーに固有の変数を含むserver.xmlは各サーバーで保持し、メイン構成用の構成ファイルを個別ファイルにして複数のサーバーで共有するなど、環境に合わせた構成ファイルの構造を作成することができます。また、アプリケーション毎の構成をそれぞれ個別ファイルにすることで、アプリケーション毎に運用やバージョン管理を行うことができます。この例では、server1というサーバーの構成ファイルserver.xmlにおいて<include>エレメントを使用し、データソースに関する設定部分をdatasource.xmlという名称の個別ファイルに分割しています。



構成例の2つ目を紹介します。

JVMのオプションのカスタマイズを行うjvm.optionsでは、ヒープサイズやその他のJVMオプションを指定することができます。jvm.optionsは、ランタイム環境としてLibertyプロファイルのetcディレクトリ配下にも、サーバー環境としてLibertyプロファイルのusrディレクトリの各サーバーディレクトリ配下にも配置することができます。両方に配置して同じ内容を設定している場合は、サーバー環境の設定が優先されます。この例では、ランタイム環境としてverbosegcログ出力の設定を行い、server1という特定のサーバー環境としてヒープサイズの設定を行っています。Libertyプロファイルでは、デフォルトのJVMオプションとして「-Xms4m」「-Xmx488m」「-XX:MaxPermSize=256m (Solarisのみ)」が設定されています。

ランタイム環境の初期化に使用されるbootstrap.propertiesには、事前定義プロパティやカスタム・プロパティを設定することができます。bootstrap.propertiesで設定したカスタム・プロパティは、server.xml内で変数化して使用することができます。

参考:

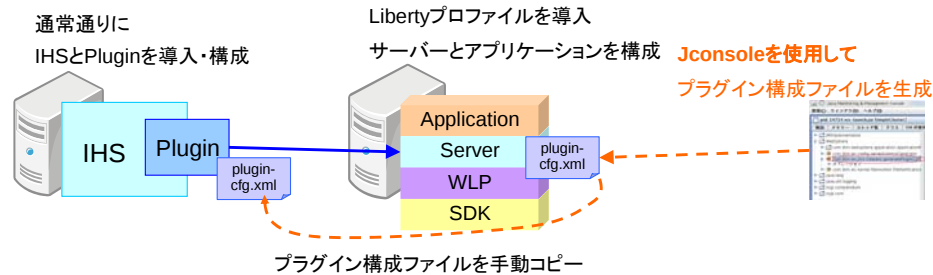
Setting generic JVM arguments in the WebSphere Application Server V8.5 Liberty profile

<http://www-01.ibm.com/support/docview.wss?uid=swg21596474>

IHS/プラグイン構成

使用するフィーチャー : localConnector-1.0

構成方法



フルプロファイルと比較して・・・

- ・プラグイン構成ファイルの生成にはJconsoleを使用 (JobManagerでも可)
- ・プラグイン構成ファイルの伝播は手動コピー

© 2012 IBM Corporation

21

LibertyプロファイルでIHS/プラグイン構成を行う場合、localConnector-1.0というフィーチャーを使用します。

構成方法は、まず、通常通りにIHSのノードに対してIHSとPluginを導入して構成します。そして、LibertyプロファイルのノードにLibertyプロファイルを導入し、サーバーとアプリケーションを構成します。次がフルプロファイルの場合と大きく異なる点で、LibertyプロファイルではJconsoleを使用してプラグイン構成ファイルを生成します。Jconsoleとは、JavaSDKで提供されているJMXに準拠した監視ツールで、JVM上で実行されているアプリケーションのリソースやパフォーマンス情報をモニターすることができます。Jconsoleの実行ファイルは<JDK_Home>/binにあります。プラグイン構成ファイルはLibertyプロファイルのサーバーディレクトリに作成されるので、それをIHSノードに手動でコピーし、プラグインが読み込めるようにします。このプラグイン構成ファイルの生成は、WDTのユーティリティー機能で実行することもできます。

以上の作業によりIHS/プラグイン構成が行われ、クライアントからのリクエストをIHSが受け付け、IHSからプラグインを経由してLibertyプロファイルのサーバーへと割り振りが行われるようになります。

フルプロファイルと比較して、Libertyプロファイルでプラグイン構成ファイルを生成するにはJconsoleを使用するという点や生成されたプラグイン構成ファイルの伝搬は手動コピーで対応しなければならないという点が異なります。プラグイン構成ファイルの生成は、JobManagerを使用してジョブとして実行することもできます。

【参考】IHS/プラグイン構成のserver.xmlの例

server.xml

```
- <server description="new server">
- <!-- Enable features -->
- <featureManager>
  <feature>localConnector-1.0</feature>
  <feature>jsp-2.2</feature>
</featureManager>
  <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" httpsPort="9443" />
  <pluginConfiguration webserverPort="80" webserverSecurePort="443" />
</server>
```

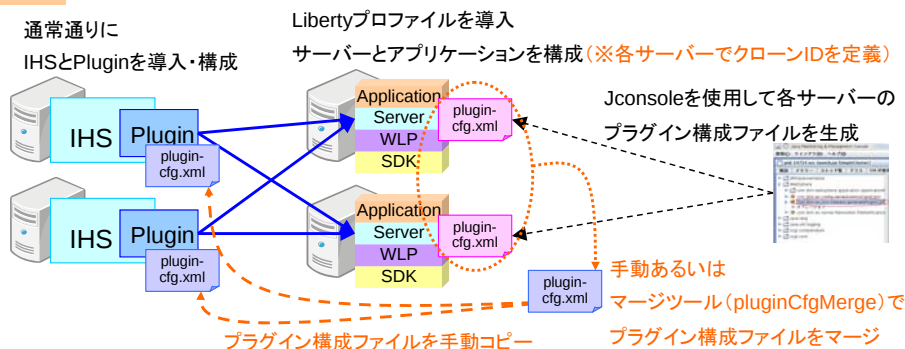
「localConnector-1.0」フィーチャーを使用

<pluginConfiguratioin>
プラグイン構成を設定

シンプルクラスター構成

使用するフィーチャー : なし

構成方法



フルプロファイルと比較して...

- ・Baseエディションのシンプルクラスターと同様にセッション・アフィニティー、サーバーのフェールオーバー、ワークロード管理が可能
- ・シンプルクラスターの台数制限は購入ライセンスに依存

© 2012 IBM Corporation

23

Libertyプロファイルでシンプルクラスター構成を行う場合、特に必要なフィーチャーはありません。

構成方法は、まず、通常通りにIHSのノードに対してIHSとプラグインを導入して構成します。そして、複数のLibertyプロファイルのノードにLibertyプロファイルを導入し、サーバーとアプリケーションを構成します。そのとき、各サーバーのクローンIDを定義する必要があるため、各サーバーのserver.xml上に任意のクローンIDを記載します。次に、各サーバーのプラグイン構成ファイルをそれぞれJconsoleを使用して生成します。プラグイン構成ファイルはLibertyプロファイルの各サーバーディレクトリにそれぞれ作成されるので、それらを手動あるいはWASのフルプロファイルで提供されているマージツール (pluginCfgMergeコマンド) を使用してマージします。最後に、そのマージしたプラグイン構成ファイルをIHSノードに手動でコピーし、プラグインが読み込めるようにします。

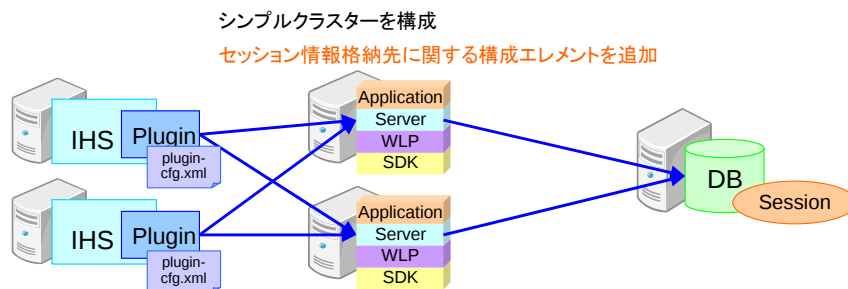
以上の作業によりシンプルクラスター構成が行われ、クライアントからのリクエストを単一あるいは複数のIHSが受け付け、IHSからプラグインを経由して複数のLibertyプロファイルのサーバーへと割り振りが行われるようになります。そして、Libertyプロファイルのサーバーへの割り振りの際には、セッション・アフィニティーやサーバーのフェールオーバーなどのワークロード管理が行われます。

フルプロファイルと比較して、Libertyプロファイルでシンプルクラスター構成を行う場合、プラグイン構成ファイルを生成するにはJconsoleを使用するという点は異なりますが、フルプロファイルのExpress/Baseエディションのシンプルクラスターと同様に、セッション・アフィニティー、サーバーのフェールオーバーなどのワークロード管理を行うことができます。Libertyプロファイルによるシンプルクラスターの台数制限は、WASの購入ライセンスに依存します。

セッション・パーシスタンス構成

使用するフィーチャー : sessionDatabase-1.0

構成方法



フルプロファイルと比較して・・・

- ・Express/Baseエディションのシンプルクラスターと同様にセッションのフェールオーバーが可能
- ・セッション保持方法としてサポートされるのはDB、WXS、XC10のみ
(メモリ間複製はサポート外)

© 2012 IBM Corporation

24

Libertyプロファイルでセッション・パーシスタンス構成を行う場合、sessionDatabase-1.0というフィーチャーを使用します。

構成方法は、シンプルクラスターを構成したら、あとはserver.xml上にセッション情報格納先に関する構成要素を追加するだけです。

以上の作業によりセッション・パーシスタンス構成が行われ、特定のサーバー障害によりメモリ上のセッション情報が失われても、外部保管しているセッション情報からセッションのフェールオーバーを実施することができるようになります。

フルプロファイルと比較して、Express/Baseエディションのシンプルクラスターと同様にセッションのフェールオーバーを行うことができます。ただし、Libertyプロファイルのセッション保持方法としてサポートされるのは、DB、WXS (WebSphere eXtreme Scale)、DataPower XC10のみであり、メモリ間複製はサポートされていません。

【参考】セッション・パーシスタンス構成のserver.xmlの例

bootstrap.properties

cloneId=sc1

server.xml

```
- <server description="new server">
- <!-- Enable features -->
- <featureManager>
- <feature>localConnector-1.0</feature>
- <feature>jsp-2.2</feature>
- </featureManager>
<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" httpsPort="9443" />
<pluginConfiguration webserverPort="80" webserverSecurePort="443" />
<include location="${shared.config.dir}/httpSessionPersistence.xml" />
</server>
```

Includeを使用して
構成ファイルを分割

httpSessionPersistence.xml

```
- <server description="Demonstrates HTTP Session Persistence Configuration">
- <featureManager>
- <feature>sessionDatabase-1.0</feature>
- <feature>servlet-3.0</feature>
- </featureManager>
- <httpEndpoint id="defaultHttpEndpoint">
- <tcpOptions soReuseAddr="true" />
- </httpEndpoint>
- <fileset id="DB2Files" includes="*.jar" dir="${shared.resource.dir}" />
- <library id="DB2Lib" filesetRef="DB2Files" />
- <jdbcDriver id="DB2Driver" libraryRef="DB2Lib" />
- <dataSource id="SessionDS" jdbcDriverRef="DB2Driver" jndiName="jdbc/sessions">
- <properties.db2.jcc driverType="4" databaseName="SAMPLE" serverName="Liberty3" portNumber="50000" user="root"
password="rootadm" />
- </dataSource>
- <httpSessionDatabase id="SessionDB" dataSourceRef="SessionDS" />
- <httpSession storageRef="SessionDB" cloneId="${cloneId}" />
</server>
```

「sessionDatabase-1.0」フィーチャーを使用

bootstrap.propertiesで定義した
プロパティを変数として使用

セキュリティ構成

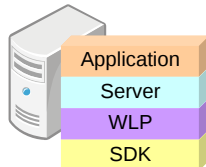
使用するフィーチャー : ssl-1.0 / appSecurity-1.0

構成方法

Libertyプロファイルを導入

サーバーとアプリケーションを構成

使用するセキュリティ機能に応じた構成エレメントを追加



<主なセキュリティ機能>

- ・SSL通信
- ・管理セキュリティ
- ・アプリケーション・セキュリティ
- ・ユーザー・レジストリー
- ・証明書作成
- ・LTPA

フルプロファイルと比較して・・・

- ・フルプロファイルのセキュリティ・フィーチャーのサブセットのみがサポート
(Libertyプロファイルではできないことの詳細は次ページ参照)

Libertyプロファイルでセキュリティ構成を行う場合、ssl-1.0やappSecurity-1.0というフィーチャーを使用します。

ssl-1.0フィーチャーを有効にすると、Secure Sockets Layer (SSL) 接続がサポートされます。セキュアHTTPSリスナーを使用する場合は、このフィーチャーを有効にする必要があります。セキュアHTTPSリスナーは、ssl-1.0フィーチャーが有効にならない限り始動されません。このフィーチャーが使用不可の場合、HTTPSリスナーは停止されます。

appSecurity-1.0フィーチャーを有効にすると、サーバー・ランタイム環境およびアプリケーションのセキュリティがサポートされます。詳細は、「Libertyプロファイル概要・開発」セッションをご参照ください。

Libertyプロファイルで利用できる主なセキュリティ機能として、SSL通信、管理セキュリティ、アプリケーション・セキュリティ、ユーザー・レジストリー、証明書作成、LTPAなどが挙げられます。構成方法は、それぞれのセキュリティ機能に応じた構成エレメントをserver.xmlに追加します。

Libertyプロファイルでは、フルプロファイルのセキュリティ・フィーチャーのサブセットのみがサポートされています。フルプロファイルと比較してLibertyプロファイルではできないことの詳細は、次ページをご参照ください。

【参考】Libertyプロファイルではできないこと

「WebSphere Application Server V8.5 Information Center」より抜粋

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.wlp.doc/topics/rwlp_sec_diff.html

- Java EE セキュリティーはサポートされません。
- パブリック API および SPI の中には、サポートされないものがあります。各 Liberty プロファイル API の Java API 文書は、インフォメーション・センターのプログラミング・インターフェース (API) のセクションに詳しく示されています。また、サーバー・イメージの /dev/ibm-api/javadoc ディレクトリの下に JAR ファイルとして入手することもできます。
- カスタム・ユーザー・レジストリーはありません。
- 水平伝搬は行われません。
- SecurityAdmin MBean はサポートされません。そのため、認証キャッシュのクリアのようなメソッドは使用できません。
- Java Authorization Contract for Container (JACC) はサポートされません。
- Java 2 Connector (J2C) のプリンシパル・マッピング・モジュールはサポートされません。
- Java Authentication SPI (JASPI) はサポートされません。
- マルチ・セキュリティー・ドメインはサポートされません。
- サーバーのセキュリティー・インフラストラクチャーの一部であるセキュリティー監査サブシステムはありません。

【参考】SSL通信のserver.xmlの例

最もシンプルな基本SSL構成(デフォルト設定を使用)

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>
```

「ssl-1.0」フィーチャーを使用

```
<keyStore>
```

SSL 暗号化のために使用されるセキュリティ証明書のリポジトリ

起動時に証明書作成

```
[12/07/26 0:28:51:234 JS1] 0000001a com.ibm.ws.kernel.feature.Internal.FeatureManager
A CWWKF00111: サーバー SimpleCluster1 は、Smarter Planet に対応する準備ができました。
[12/07/26 0:28:53:268 JS1] 00000012 com.ibm.ws.ssl.config.WsKeyStore
A CWWK10003A: SSL 証明書の作成に 2.032 秒かかりました。SSL 鍵ファイル: /opt/IBM/WLP/mlp/usr/0
.../SimpleCluster1/resources/security/key.jks
[12/07/26 0:28:52:320 JS1] 00000012 com.ibm.ws.tcpchannel.internal.TCPChannel
I CWWK002191: TCP チャンネル defaultHttpEndpoint-ssl が開始され、現在、ホスト * (IPv6)、ポート
9443 の要求を listen しています。
```

カスタマイズしたSSL構成

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>

<keyStore id="myKeyStore" password="{xor}DFoKyp="
  location="{server.config.dir}/mykeystore.p12"
  type="PKCS12"/>
<keyStore id="myTrustStore" password="{xor}DFoKyp="
  location="{server.config.dir}/mytruststore.p12"
  type="PKCS12"/>
<ssl id="mySSLConfig" keystoreRef="myKeyStore"
  trustStoreRef="myTrustStore"/>

<httpEndpoint id="defaultHttpConfig">
  <sslOptions sslRef="mySSLConfig"/>
</httpEndpoint>
```

```
<sslOptions>
```

トランスポートのSSLプロトコル構成を設定

```
<sslDefault>
```

SSLサービスのデフォルト構成レパートリー

【参考】管理セキュリティのserver.xmlの例

管理ユーザー1人だけの設定

`<quickStartSecurity>`

単純な管理セキュリティを設定

```
<quickStartSecurity userName="bob"
  userPassword="{xor}Lz4sLCgwLTs"/>
<keystore id="DefaultKeyStore"
  password="{xor}DFoKyp="/>
```

管理ユーザー複数人の設定

```
<administrator-role>
  <user>fred</user>
  <group>administratorsGroup</group>
</administrator-role>
```

`<administrator-role>`

サーバー管理者ロールを割り当てられた
ユーザー/グループを設定

【参考】ユーザー・レジストリーのserver.xmlの例

基本XMLレジストリー

```
<server>
  <featureManager>
    <feature>appSecurity-1.0</feature>
  </featureManager>

  <basicRegistry realm="basicRealm">
    <user name="bob" password="{xor}CDo9Hgw=" />
    <group name="group1">
      <member name="bob"/>
    </group>
  </basicRegistry>
</server>
```

<user> <group>
ユーザー・レジストリー内のユーザーとパスワードを設定

LDAPレジストリー

```
<server>
  <featureManager>
    <feature>appSecurity-1.0</feature>
  </featureManager>

  <ldapRegistry host="ccwin12.austin.ibm.com"
    port="389" baseDN="o=ibm,c=us"
    ldapType="IBM Tivoli Directory Server" />
</server>
```

<ldapRegistry>
LDAPユーザー・レジストリー構成を設定

SAFレジストリー

```
<featureManager>
  <feature>zosSecurity-1.0</feature>
</featureManager>

<safRegistry id="saf"/>
```

【参考】証明書作成のコマンドの例

securityUtilityコマンド

- ・「encode」パラメータ : Libertyプロファイルのプレーン・テキストの暗号化
 - ・「createSSLCertificate」パラメータ : 自己署名証明書の作成
- <引数>
パスワード、証明書有効期限、証明書のサブジェクトおよび発行者DNを指定可能

自己署名証明書作成のコマンド実行結果

```
C:\IBM\WebSphereV85Liberty\wlp\bin>securityUtility.bat createSSLCertificate --server=TestServer --
password=password
鍵ストア C:\IBM\WebSphereV85Liberty\wlp\usr\servers\TestServer\resources\security\key.jks を作成中です

サーバー TestServer に SSL 証明書が作成されました

SSL を使用可能にするには、server.xml に次の行を追加します。

<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>
<keyStore id="defaultKeyStore" password="{xor}Lz4sLCgwLTs=" />

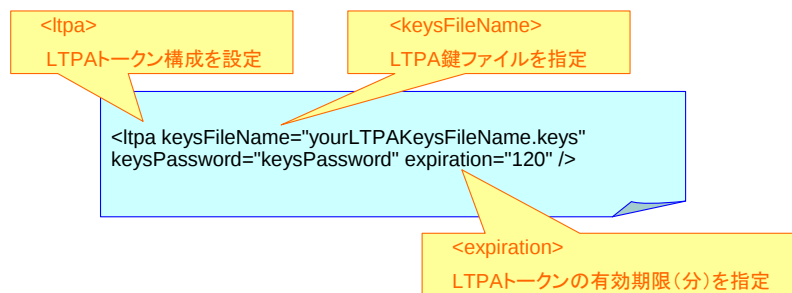
C:\IBM\WebSphereV85Liberty\wlp\bin>
```

指定サーバーのディレクトリ配下に
証明書作成

【参考】LTPAのserver.xmlの例

LTPA構成

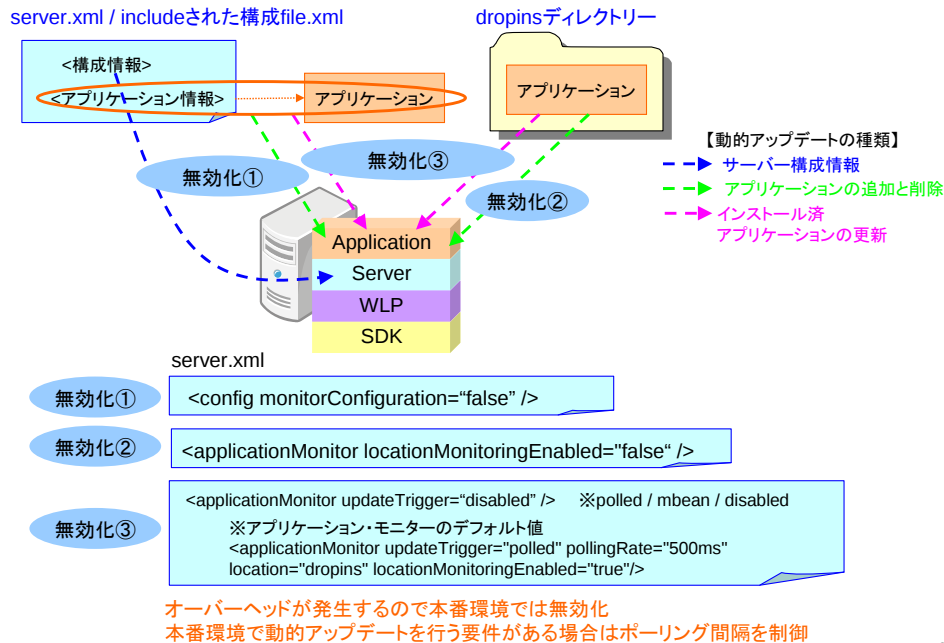
分散環境のSSO とセキュリティーを暗号化によりサポート



5. 運用管理

- 動的アップデートの制御
- モニター機能
- モニター・データ
- ログ出力
- Dump出力
- JobManager連携
- その他の運用管理項目

動的アップデートの制御



© 2012 IBM Corporation

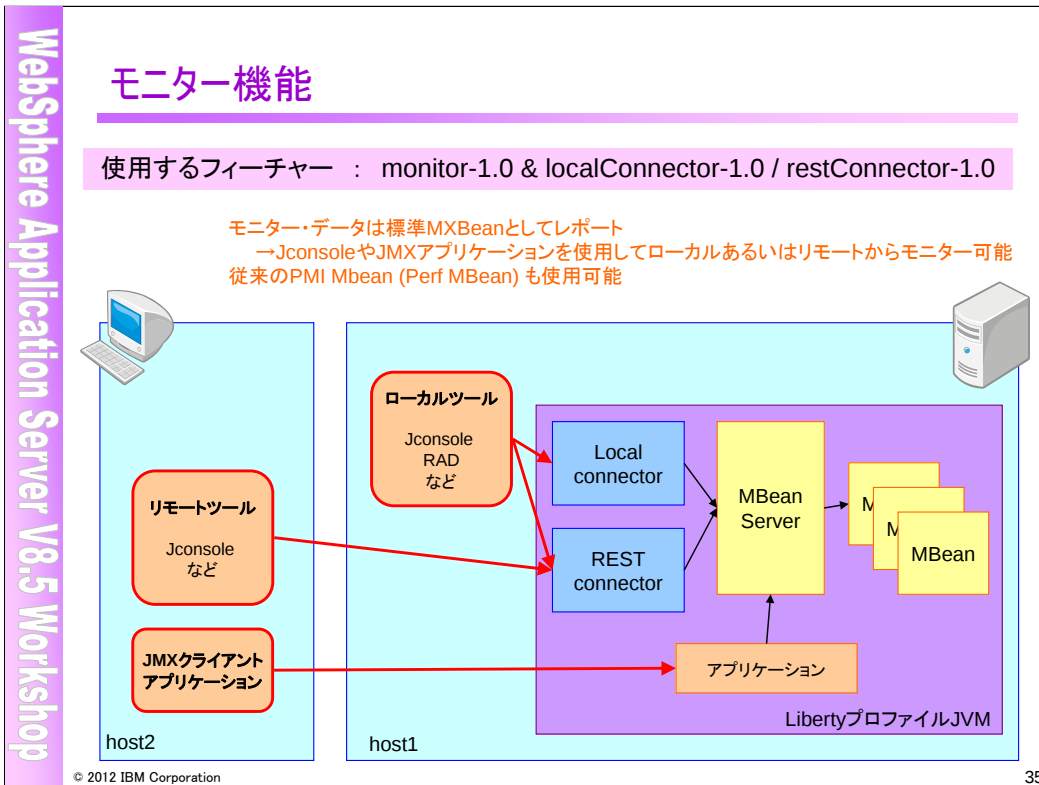
34

Libertyプロファイルは、サーバー構成やアプリケーション情報を動的にアップデートすることができますが、その機能を必要に応じて制御することができます。

動的アップデートの種類としては、サーバー構成情報、アプリケーションの追加と削除、インストール済アプリケーションの更新、の3つがあります。サーバー構成情報は、server.xmlあるいはincludeされた構成ファイルから動的アップデートを行うことができます。アプリケーションの追加と削除およびインストール済アプリケーションの更新は、server.xmlあるいはincludeされた構成ファイルやアプリケーション配置ディレクトリー、dropinsディレクトリーから動的アップデートを行うことができます。これらの動的アップデート機能に関して、それぞれの方法で無効化あるいはモニターのポーリング間隔の変更を行うことができます。

構成情報をモニターする構成サービスやアプリケーション・モニターによるポーリングはオーバーヘッドが発生するので、本番環境では無効化します。本番環境で動的アップデートを行う要件がある場合は、ポーリング間隔を長く設定するなどして、できるだけオーバーヘッドが少なくなるように対応します。

。



Libertyプロファイルのサーバー・ランタイム環境をモニターするためにはmonitor-1.0フィーチャーを使用します。そして、ローカルツールからモニター・データを確認するためにはlocalConnector-1.0フィーチャー、リモートツールからモニター・データを確認するためにはrestConnector-1.0フィーチャーを使用します。

monitor-1.0フィーチャーを有効にするとモニターが開始され、モニター・データは標準MXBeanとしてレポートされます。モニター用のMXBeanは、「WebSphere:type=JvmStats」(JVM)、「WebSphere:type=ServletStats,name=*」(スレッドプール)、「WebSphere:type=ThreadPoolStats,name=DefaultExecutor」(Webアプリケーション)です。また、従来のPMI Mbean (Perf MBean)も使用できますので、必要に応じてモニター・データを確認することができます。

localConnector-1.0フィーチャーでは、ローカルJMXコネクタが提供されます。このコネクタはJVMに組み込まれ、同一ホスト・マシン上で同一ユーザー ID、同一JDKで実行しているユーザーのみが使用することができます。Jconsoleなどの JMXクライアントやAttach API を使用するその他のJMXクライアント、RADなどによるローカル・アクセスが可能になります。

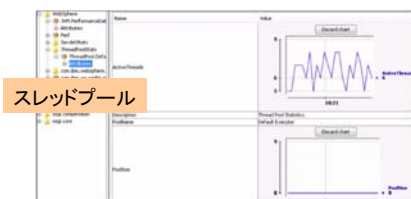
restConnector-1.0フィーチャーでは、セキュアなJMXコネクタが提供されます。このコネクタは、任意のJDKを使用してローカルまたはリモートで使用することができます。RESTベース・コネクタ経由のJMX クライアントによるリモート・アクセスが可能になります。SSLおよび基本ユーザー・セキュリティ構成が必要になります。

【参考】モニター・データ

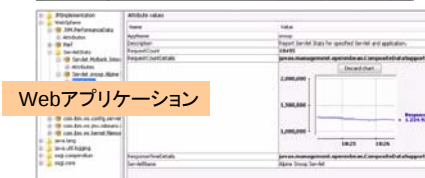
JconsoleでJVMに接続 → MBeanの各属性をクリックするとモニター・データが確認可能



- Heap: 現行JVMに使用されているヒープ・サイズ
- FreeMemory: 現行JVMに使用可能な空きヒープ
- UsedHeap: 現行JVMの使用済みヒープ
- ProcessCPU: JVMプロセスで使用されたCPUのパーセンテージ
- GcCount: JVMの始動以降にGCが行われた回数
- GcTime: GC時間の合計累算値
- UpTime: JVM の始動以降の時間(ミリ秒単位)



- スレッドプール名
- 要求を処理中のアクティブ・スレッド数
- スレッドプールのサイズ



- アプリケーション名
- サーブレット名
- 要求カウント数
- 平均応答時間(ナノ秒単位)

ログ出力

主なログ

<WLP>/usr/servers/server1/logs配下

console.log 基本的なサーバー状況およびオペレーション・メッセージ (タイムスタンプなし)、verbosegcログ出力

<例> [監査] CWWKE0001: サーバー TestServer が起動されました。
[監査] CWWKZ0058: アプリケーションの dropins をモニター中です。

messages.log System.out / System.err およびその他のメッセージ (タイムスタンプと発行元スレッドIDあり)

<例> [12/07/06 23:16:29:078 JST] 00000001 .ibm.ws.kernel.launch.internal.platform.FrameworkManagerImpl A
CWWKE0001: サーバー TestServer が起動されました。
[12/07/06 23:16:32:406 JST] 0000000f com.ibm.ws.app.manager.internal.monitor.DropinMonitor A
CWWKZ0058: アプリケーションの dropins をモニター中です。
CWWKF0008: フィーチャー更新が 1.969 秒で完了しました。

trace_<timestamp>.log 現行トレース構成で決定された詳細レベルのトレース・メッセージ (タイムスタンプあり)

<WLP>/usr/servers/server1/logs/ffdc配下

ffdc_<timestamp>.log 初期障害データ・キャプチャー機能 (FFDC) のメッセージ
(通常は要求されたオペレーションの失敗に関連する診断データの指定域ダンプ含)

ロギング・プロパティ

「server.xml」あるいは「bootstrap.properties」で設定 (詳細は次ページ)

server.xml

```
<logging maxFileSize=xx
logDirectory=xxxxx
/>
```

bootstrap.properties

```
com.ibm.ws.logging.max.file.size=xx
com.ibm.ws.logging.log.directory=xxxxx
```

© 2012 IBM Corporation

37

Libertyプロファイルの主なログは、console.log、messages.log、trace_<timestamp>.log、ffdc_<timestamp>.logなどがあり、デフォルトではLibertyプロファイルの各サーバーディレクトリ配下のログディレクトリに出力されます。

console.logには、基本的なサーバー状況およびオペレーション・メッセージが出力されます。タイムスタンプは出力されません。JVMオプションでverbosegcを有効にした場合のログは、console.logに出力されます。

messages.logには、SystemOutやSystemErrおよびその他のメッセージが出力されます。タイムスタンプと発行元スレッドIDが出力されますので、障害時の問題判別の際などは、こちらのログを確認します。

trace_<timestamp>.logには、現行トレース構成で設定された詳細レベルのトレースやメッセージが出力されます。タイムスタンプは出力されます。

ffdc_<timestamp>.logには、初期障害データ・キャプチャー機能 (FFDC) のメッセージが出力されます。通常は、要求されたオペレーションの失敗に関連する診断データの指定域ダンプが含まれます。

ログファイルの出力先や最大サイズや出力内容などの詳細は、ロギング・プロパティとしてserver.xmlあるいはbootstrap.propertiesで設定することができます。

【参考】ロギング・プロパティ

server.xmlで 設定する属性	bootstrap.propertiesで 設定するプロパティ	説明
logDirectory	com.ibm.ws.logging.log.directory	FFDC を含むすべてのログ・ファイルのディレクトリーを設定します。
maxFileSize	com.ibm.ws.logging.max.file.size	ログ・ファイルの許容最大サイズ(MB)。これを超えるとローテートされます。これを無効にするには、値を 0 に設定します。
maxFiles	com.ibm.ws.logging.max.files	最大ファイル・サイズが有効な場合に、この設定を使用して、保持する各ログ・ファイルの数を決定します。
consoleLogLevel	com.ibm.ws.logging.console.log.level	console.log ファイルに入れるメッセージの細分度を制御します。有効な値は INFO、AUDIT、WARNING、ERROR、および OFF です。デフォルトのレベルは AUDIT です。
messageFileName	com.ibm.ws.logging.message.file.name	メッセージ・ログのデフォルト名は messages.log です。このファイルは常に存在し、System.out と System.err に加えて、INFO とその他 (AUDIT、WARNING、ERROR、FAILURE) のメッセージが含まれます。このログには、タイム・スタンプと発行元スレッド ID も含まれます。
traceFileName	com.ibm.ws.logging.trace.file.name	追加トレースまたは詳細トレースが有効な場合にのみ作成されます。
traceSpecification	com.ibm.ws.logging.trace.specification	トレース・ストリングを使用して、選択的にトレースを有効にします。デフォルトは *=-info=enabled です。
traceFormat	com.ibm.ws.logging.trace.format	トレース・ログのフォーマットを制御します。Liberty プロファイルのデフォルトのフォーマットは ENHANCED です。フルプロファイルのように BASIC と ADVANCED のフォーマットも使用できます。

Dump出力

サーバーのスナップショット

【取得方法】

dump アクション実行(サーバー稼働中/停止中でも実行可能)

[例] # <WLP>/bin/server dump TestServer --archive=TestServer.dump.zip

サーバーのスナップショット情報を含む圧縮ファイル作成

サーバー TestServer のダンプが

C:\IBM\WebSphereV85Liberty\wlp\usr\servers\TestServer\TestServer.dump.zip で完了しました。

【取得情報】

サーバー構成

ログ情報

デプロイ済みアプリケーションの詳細

※サーバー稼働中の場合は下記情報も取得

サーバー内の各 OSGi バンドルの状態、

スレッド情報、ヒープサイズ、OS、

ネットワーク状況などのランタイム環境設定 など

Javacore / HeapDump / SystemDump

【取得方法】

(例)IBM JDK の場合

「jvm.options」に以下を追記して「kill -3 <PID>」(Windowsの場合はCtrl+Break)

-Xdump:heap

← HeapDump取得

-Xdump:java+heap+system

← Javacore&HeapDump&SystemDump取得

© 2012 IBM Corporation

39

Libertyプロファイルでは、サーバーのスナップショットを取得する機能が提供されています。取得方法は、serverコマンドのdumpアクションを使用します。dumpアクションはサーバー稼働中でも停止中でも実行することができ、実行すると、そのサーバーのスナップショット情報を含む圧縮ファイルがLibertyプロファイルの各サーバーディレクトリ配下に作成されます。取得情報は、サーバー構成、ログ情報、デプロイ済みアプリケーションの詳細などです。サーバー稼働中にdumpアクションを実行した場合は、それに加えて、サーバー内の各 OSGi バンドルの状態、登録されたOSGiサービスの情報、スレッド情報、JVM、ヒープサイズ、オペレーティング・システム、ネットワーク状況などのランタイム環境設定などの詳細な情報が取得できます。このサーバーのスナップショット情報は、WDTのユーティリティー機能で取得することもできます。

また、Javacore/HeapDump/SystemDumpなどの一般的なDumpも取得することができます。取得方法は、jvm.optionsにそれぞれに必要なオプションを指定した上で、Unixの場合は「kill -3」コマンドを実行することで、Windowsの場合はサーバーをCtrl+Breakで停止することで、各DumpがLibertyプロファイルの各サーバーディレクトリ配下に出力されます。

参考:

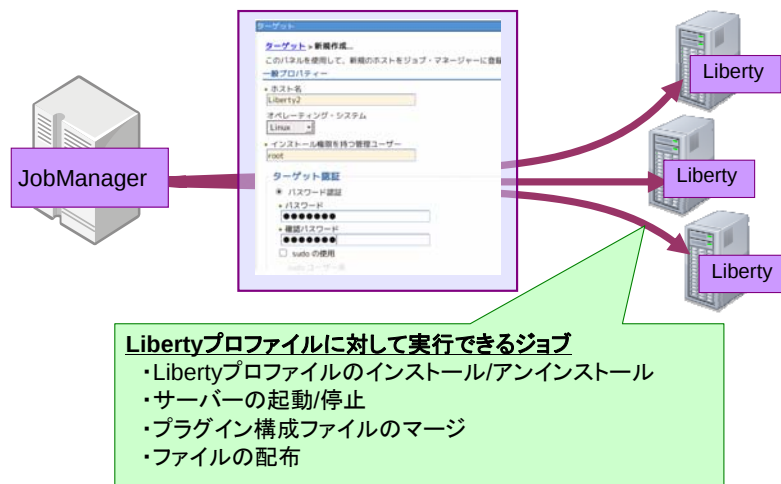
How to generate javacores, heapdumps and system cores for the WebSphere Application Server V8.5 Liberty profile

<http://www-01.ibm.com/support/docview.wss?uid=swg21597830>

JobManager連携

構成方法

フルプロファイルと同様にLibertyプロファイルをホスト・ターゲット登録
(管理コンソール、wsadminのregisterHostコマンド、などを使用)



JobManagerを使用すると、フルプロファイルのNDセルやExpress/Baseエディションのサーバーを統合管理すると同様に、複数のLibertyプロファイルも統合的に集中管理することができます。

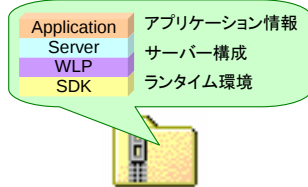
LibertyプロファイルをJobManagerと連携させるには、フルプロファイルの場合と同様で、Libertyプロファイルのサーバーを管理コンソールやwsadminのregisterHostコマンドなどを使用してJobManagerのホスト・ターゲットとして登録します。

JobManagerからLibertyプロファイルに対して実行できるジョブは種類が限られており、Libertyプロファイルのインストール/アンインストール、サーバーの起動/停止、プラグイン構成ファイルのマージ、ファイルの配布があります。

その他の運用管理項目

バックアップ / リストア

パッケージ機能を使ってアーカイブ化



監視

Javaプロセス監視
(PIDファイルなし)

serverStatus-1.0フィーチャーを使用した
JobManagerでの稼働監視



Fix適用

(2012年9月時点でFixPack未リリース)

<適用方法>

既存環境を事前に保管しておき
FixPackが適用された(修正が含まれた)状態のLibertyプロファイルのjarを
新たにインストールして既存環境を適用する形式になる予定

※iFix適用方法は次ページ参照

© 2012 IBM Corporation

41

その他にも、本番環境ではさまざまな運用管理項目がありますが、Libertyプロファイルにおける対応については、以下の通りです。

バックアップ/リストアに関しては、Libertyプロファイルで提供しているパッケージ機能を使用してアーカイブ化することで、ランタイム環境からアプリケーション情報まで全ての情報を容易にバックアップ取得することができ、zip展開することで容易にリストアすることができます。

監視に関しては、フルプロファイルで一般的に行われているのと同様に、Javaプロセス監視を行うことができます。LibertyプロファイルではPIDファイルは存在しないため、PIDファイルの監視を行うことはできません。また、serverStatus-1.0フィーチャーを使用すると、LibertyプロファイルのサーバーはJobManagerに対して自動的にランタイム状況を通知することができるようになりますので、それで稼働状況を監視することもできます。

Fix適用に関しては、2012年9月時点でV8.5のFixPackはまだリリースされていません。適用方法に関しては、既存構成を事前に保管しておき、FixPackが適用された状態の(V8.5.0.1相当の修正を含む)Libertyプロファイルのjarがリリースされるので、それを新しくインストールし、事前に保管しておいた既存環境を適用するという形式になるようです。フルプロファイルの場合のようなFix適用のためのツールは不要であり、既存の環境にFixを適用するという形式ではないようです。

【参考】iFix適用方法

iFix適用方法

1. サーバー停止「server stop <servername>」
2. Jarファイルで提供されるiFixモジュール展開(Libertyプロファイルのインストール先指定)
3. 適用後は「--clean」オプションをつけてサーバー起動「server start <servername> --clean」
(※)「--clean」オプションによりworkareaディレクトリ(<WLP>/usr/servers/<servername>/workarea)にあるOSGiキャッシュやplatformキャッシュを削除(再作成)

iFix適用後のサーバー起動時のメッセージ出力 (iFixに関連するフィーチャーを有効にしている場合のみ出力)
「[監査] CWWKF0014W: サーバーには次のテスト修正がインストールされています。PMxxxxx。」

fixesディレクトリにxmlファイル自動作成 (fix適用の履歴)

```
<WLP>/lib/fixes配下
2012/08/02 13:39 <DIR> .
2012/08/02 13:39 <DIR> ..
2012/08/02 13:39 1,508 8.5.0.0-WS-WAS_WLPArchive-TFPM69790_8.5.0.20120731_1104.xml
```

iFix削除方法

1. サーバー停止「server stop <servername>」
2. 以下のファイルを手動で削除
 - ・<WLP>/lib/com.ibm.ws.jndi.1.0.0.20120731-1104.jar (iFix適用によって追加されたファイル)
 - ・<WLP>/lib/fixes/8.5.0.0-WS-WAS_WLPArchive-TFPM69790_8.5.0.20120731_1104.xml
3. 削除後は「--clean」オプションをつけてサーバー起動「server start <servername> --clean」

まとめ

WebSphere Application Server V8.5 Workshop

まとめ

セッションの内容

概要	トポロジー	導入	構成	運用管理
運用における効果 性能評価結果	基本トポロジー JobManagerトポロジー	導入前提 パッケージ機能 JobManager 導入全体の流れ	構成方法 フィーチャー/構成エレメント IHS/プラグイン シンプルクラスター セッション・パーススタンス セキュリティ	動的アップデートの制御 モニター機能 ログ Dump JobManager連携 その他の運用管理項目

Libertyプロファイルの本番環境への適用

開発段階では積極的にLibertyプロファイルの使用を検討するのに対して
本番環境への適用にあたってはLibertyプロファイルの制約も理解した上で検討

<Libertyプロファイルを本番環境で利用するのに有効なケース>

- ・これまでTomcatで本番運用していた場合
- ・サーバーリソースが少ない場合
- ・動的アップデートや短時間サーバー起動の要件がある場合
- ・ベンダーによる長期サポートが求められる場合
などなど

© 2012 IBM Corporation
44

当セッションでは、Libertyプロファイルの概要、トポロジー、導入、構成、運用管理に関して説明し、提供される機能の使用方法や制約などについて紹介しました。

Libertyプロファイルは、開発段階では積極的に使用することを検討するのに対し、本番環境への適用にあたっては、当セッションで紹介したような制約も理解した上で検討する必要があります。Libertyプロファイルで提供されていないトランザクション管理やWebサービスなどの機能は使用せず、運用要件も満たしている場合は、Libertyプロファイルを有効に利用することができます。

Libertyプロファイルを本番環境で利用するのに有効なケースとして、いくつか例を挙げます。

まず、これまでTomcatで本番運用していた場合です。Tomcatで実現されている機能は基本的にLibertyプロファイルでも利用可能であり、TomcatからLibertyプロファイルに移行するためのアプリケーション・マイグレーション・ツールも提供されています。また、サーバーリソースが少ない場合も有効です。Libertyプロファイルが使用するメモリやディスクは、アプリケーションの実行に必要な最低限の容量に抑えることができますので、限られたサーバーのリソースを効率的に利用することができます。そして、Libertyプロファイルの大きな特徴ともいえる動的アップデートや短時間サーバー起動などの要件がある場合も、Libertyプロファイルを有効に利用することができます。フリーでサポートのないLightweightなアプリケーション・サーバーやJDKが多く存在する中、WASの安価なExpressエディションを購入し、同梱されているLibertyプロファイルで運用すれば、LibertyプロファイルやJDKのサポートを得ることができます。

参考文献

参考文献

- 「WebSphere Application Server V8.5 Information Center」
 - ◆ http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.home.doc_wasinfo_v8r5/welcome_ic_home.html
- 「System Requirements for WebSphere Application Server v8.5 – Liberty」
 - ◆ <http://www.ibm.com/support/docview.wss?rs=180&uid=swg27028175>
- 「WebSphere Application Server Liberty profile feature dependencies might be different in the developer tools and server runtime environment」
 - ◆ <http://www.ibm.com/support/docview.wss?uid=swg21593248>
- 「Setting generic JVM arguments in the WebSphere Application Server V8.5 Liberty profile」
 - ◆ <http://www.ibm.com/support/docview.wss?uid=swg21596474>
- 「How to generate javacores, heapdumps and system cores for the WebSphere Application Server V8.5 Liberty profile」
 - ◆ <http://www.ibm.com/support/docview.wss?uid=swg21597830>

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、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、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様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様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、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、WebSphere、およびz/OS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。

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。