

WebSphere Application Server V9

Liberty 基盤設計セミナー

導入 / 構成



アジェンダ

□ 導入

- ランタイムの導入方法
 - IBM Installation Manager / アーカイブ展開
- 実働環境への導入作業の流れ
 - ランタイム・Java / Fix / フィーチャー
- BluemixにおけるLibertyの導入

□ 構成

- 構成要素
 - ディレクトリ構造
 - 使用可能な環境変数
 - 構成ファイル (server.xml他、include可能、などの特徴)
- 構成 の おまけ機能
- 「構成」の運用を考える
 - サーバー構成をアーカイブする
 - ディレクトリ構成運用を考える
 - サーバー構成ファイルの運用を考える
- 構成の流れ と 構成手順
 - 1. スタンドアロン構成
 - 2. シンプル・クラスター構成
 - 3. 高可用性構成

導入

導入時に考えること

何を？



ランタイム



Fix



フィーチャー



Java

どこから？



メディア

ご契約中の
PAサイト

Fix Central

Fix Central

WASdev

Liberty
Repository



ローカルリポジトリ
(LARS / Local)

どうやって？



IIM
(IBM Installation Manager)



アーカイブ展開
(zip/jar)



installUtility
コマンド

LARS : Liberty Asset Repository Service

© 2016 IBM Corporation

Libertyランタイムを導入する場合に、何を？どこから取得？どうやってインストール？など、考えるポイントがあります。

何を？ : ランタイム自身、Fix、フィーチャー、ランタイム稼働に必要なJavaなどがあります。

どこから？ : メディアDVD、PAサイト、Fix Centralがあります。これとは別にWASdev(Liberty Repository)というWebサイト、オフラインアクセス可能なローカルリポジトリ(LARS/Local)から取得します。

どうやって？ : IIM(IBM Installation Manager)、アーカイブ展開(zip/jar)、installUtilityコマンドを利用します。

Liberty Repositoryとは

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp_repository.html

Liberty Repositoryは、Libertyに関連する様々なアセットがダウンロード可能なリポジトリです。

基本的には、WASdev(要インターネットアクセス)を指します。

オフラインで使用する場合には、LARSやLocal Repositoryを事前に構築しておくことで対応できます。

取得可能なアセットは、アドオン、管理スクリプト、構成スニペット、フィーチャー、オープンソース統合、製品、製品サンプル、ツールがあります。

【参考】 WASdev サイト

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

Download WAS Liberty

Download Liberty in Eclipse
Install Eclipse plugins to develop, deploy, and debug applications using WebSphere Liberty. Download and manage Liberty installations from Eclipse. Free for development.
GET IT NOW!

Download the Liberty runtime
Download the WebSphere Liberty runtime to install in build environments, to deploy from the command line, and to develop applications in other IDEs. Free for development.
GET IT NOW!

Want to try the Beta?
The July Liberty beta includes updates to the CDI 1.2 feature, including Weld Probe.
GET IT NOW!

GA版

Beta版

Eclipse上のWDT

GA版の自己展開圧縮 ZIPファイル

Beta版の自己展開圧縮 ZIPファイル

WAS Liberty with Java EE 7 Web Profile and IBM Java SDK 8

ASSET TYPE: PRODUCT

WAS Liberty V16.0.0.2 with Java EE 7 Web Profile and IBM Java SDK 8. The lightweight WAS Liberty is production-ready and designed for developers. This ZIP file is Java EE 7 Web Profile certified. To simplify getting started, this package also includes a copy of IBM Java.

IBM Java同梱のアーカイブも取得可能

6

© 2016 IBM Corporation

WASdevのサイトには以下のURLでアクセスできます。

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

前スライドであげたLibertyに関連する様々なアセットがダウンロード可能です。

フィーチャーに関しては、Libertyの導入ディレクトリのbin以下にあるinstallUtilityコマンドを使用してインストールする必要があります。

また、GA版のランタイムだけでなく、月1の頻度でリリースされるBeta版も取得可能です。

Beta版には、新機能が含まれますので、いち早く確認することが可能です。

ランタイムの導入方法

□ 本番・検証用にランタイムを使用する場合

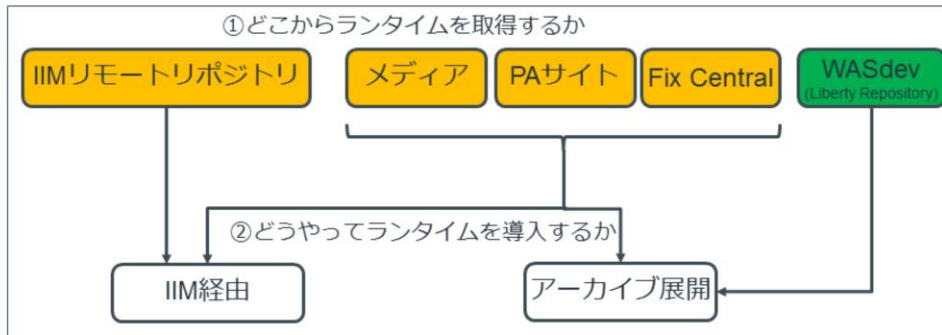
=> **ライセンス版**を導入

※開発用途にも利用可

※評価検証は60日間に限り無償

□ 本番・検証用にランタイム移行するつもりが無い場合(開発のみ)

=> **開発版**をWASdevから導入



7

© 2016 IBM Corporation

ランタイムを導入する際のモチベーションには2つあると考えています。

ひとつは、本番環境や検証環境をサーバーに構成する場合です。

導入したランタイムはそのまま本番・検証で利用しますので、ライセンス版が必要になります。

取得元は、IIMリモートリポジトリ、メディアDVD、PAサイト、Fix Centralです。

導入方法は、IIM経由およびアーカイブ展開です。

ふたつめは、開発用に導入したランタイムを本番・検証環境にそのまま移行しない場合です。

あくまで開発用途でローカル開発PCでの使用になるため、WASdev(Liberty Repository)からアーカイブ版を取得することになります。

※後のスライドで述べますが、開発用途無償版(BASE_ILAN、サポート無し)を、本番・検証環境として移行することも可能です。その場合、ライセンス化ファイルを適用することでライセンス版にアップグレードすることができます。

取得元は、WASdevやFix Centralです。

導入方法は、アーカイブ展開です。

IIMでインストールするためのイメージを取得した場合は、IIM経由でLibertyを導入します。

アーカイブ形式で取得した場合は、アーカイブ展開で導入することになります。

アーカイブ展開によるLiberty導入パターン

1. 任意のアーカイブを取得

1) Libertyカーネルのみ

※Javaが同梱されるパターンは、4)のみ

※ 5) はライセンスをお持ちの方のみ

Libertyカーネル

2) Libertyカーネル + Liberty独自フィーチャー + Java EE 7 Web Profile

Libertyカーネル

独自フィーチャー

webProfile-7.0

3) Libertyカーネル + Liberty独自フィーチャー + Java EE 7 Full Platform

Libertyカーネル

独自フィーチャー

javaee-7.0

4) Libertyカーネル + Liberty独自フィーチャー + Java EE 7 Web Profile + IBM Java 8

Libertyカーネル

独自フィーチャー

webProfile-7.0

IBM Java 8

5) Libertyカーネル + 全てのフィーチャー

Libertyカーネル



全てのフィーチャー

2. アーカイブを展開することで導入(unzip/java -jar)

8

© 2016 IBM Corporation

アーカイブ展開によるLiberty導入パターンについてです。

手順はシンプルで、任意のアーカイブを取得し、unzipやjava -jarで展開するだけです。

1-4はzip形式で提供されるもので、ランタイム機能のみのLibertyカーネルや、Java EEのフィーチャーやLiberty独自フィーチャーが含まれるものがあります。

5は、jar形式でライセンス版のみ提供されるもので、全てのフィーチャーが導入済のアーカイブになります。

【アーカイブ(zip/jar)の入手元】

開発用途無償版は、

Liberty Repository、または、Fix Centralから入手可能です。

zip形式のアーカイブが全て含まれます。

ライセンス版のjar形式のアーカイブは、

Liberty Core、Base、NDすべての製品エディション用に提供しています。

メディア、PA、Fix Centralから入手可能です。

webProfile-6.0が含まれるjarと、フィーチャーが全て導入された.jarがあります。

※フィーチャー全部入りのjarは、Fix Centralからのみ入手可能です。

【参考】開発版を本番・検証環境に移行する場合

ライセンス化ファイルを適用する必要あり

○ 入手元: PAサイト/Fix Central

- ND用 (wlp-nd-license.jar)
- Base用 (wlp-base-license.jar)
- LibertyCore用 (wlp-core-license.jar)

} Liberty導入ディレクトリで、
以下コマンドで適用(NDの場合)
\$ java -jar wlp-nd-license.jar

□ Fix Central や、Liberty Repositoryから取得・導入した Liberty(BASE_ILAN=サポート無し、開発用途)を移行する場合

- ライセンス化ファイル適用で、Base / ND に移行可能

□ Liberty Core V8.5.5 評価版を移行する場合

- ライセンス化ファイル適用で、Liberty Coreに移行可能

開発用途無償版(BASE_ILAN、サポート無し)として導入したLibertyを、本番・検証環境に移行する際には、ライセンス化ファイルを適用する必要があります。
ライセンス化ファイルの適用により、ライセンス版にアップグレード可能です。

【ライセンス化ファイルの入手元】

ND/Base/Liberty Coreの各エディション用のライセンス化ファイルは、PAサイトもしくはFix Centralから入手可能です。

【ライセンス化ファイルの適用】

Liberty導入ディレクトリで、java -jarコマンドを使用して適用します。

(例: java -jar wlp-nd-license.jar)

【注意点】

Liberty Repositoryなどから入手した開発用途無償版(BASE_ILAN、サポート無し)に対して、Liberty Coreへのアップグレードはできません。

Liberty Core用のライセンス化ファイルは、Liberty Core評価版(V8.5.5)を本番移行する際に使用します。

IIM経由でのLiberty導入パターン

1. メディア、PAサイト、Fix CentralからIIMでのインストールイメージを取得

※IIMリモートリポジトリ(URL)の指定でも可

2. IIMを使用して、Libertyを導入

○フィーチャー全部入り

■使用可能なすべてのフィーチャーが事前導入済

○任意のフィーチャー個別選択

■Java EEのフィーチャー

■Liberty独自のフィーチャー(adminCenter-1.0、etc.)

IIM経由でLibertyを導入する場合は、IIMでリポジトリの登録を行います。その方法は2つあります。

メディアやPAサイト、Fix Centralなどから取得したイメージを指定する方法と、リモートリポジトリを指定する方法です。

後者のリモートリポジトリを指定する場合のURLを以下に示します。

- NDエディション

<http://www.ibm.com/software/repositorymanager/V9WASND>

- Baseエディション

<http://www.ibm.com/software/repositorymanager/V9WASBase>

※いずれも、本番もしくは検証(60日間)の利用用途のためのエディションになります。

導入時には、フィーチャーやアドオン、サンプルなどを選択可能です。

フィーチャーについては、全部入りもしくは個別選択が可能です。

【参考】IBM HTTP Server 及び Plug-in の導入

テスト、本番環境でのLibertyサーバーの利用においては、DMZに IHS/Plug-in を配置し、後段のセキュア・ゾーンにLibertyサーバーを配置するトポロジが一般的です。以下、 IHS/Plug-in の入手・導入方法です。

□ 入手方法

- メディア、PAサイトからのみの提供
- 導入には、IIMとIBM JDK V8も必要

□ 導入方法

- IIMによる導入方法のみ提供
- IIM導入後、IHS、Plug-in、IBM JDK V8の3つをIIMのリポジトリに登録して導入を実施

□ Fix適用

- Fix Centralからダウンロード後、IIMでFixを適用
- 導入サーバがインターネットに接続可能な場合は、IIMで直接Fixを入手して適用可能

IHSやPlug-inを導入する際には、IIM経由での導入が必須となります。また、IBM Javaの導入も必須となります。

Libertyの導入時と同様に、IIMでリポジトリの登録を行います。その方法は2つあります。

メディアやPAサイトから取得したイメージを指定する方法と、リモートリポジトリを指定する方法です。

リモートリポジトリは以下です。

<http://www.ibm.com/software/repositorymanager/V9WASSupplements>

導入後のFix適用は、IIM経由で行います。

導入方法選択の指針

□IIM経由

- Fix適用をツールで管理したい
- バージョンのロールバックをやりたい
- IBM JavaやIHSが必要な場合、IIMの利用が必要
- ランタイム、Java、IHSなどをまとめて導入したい

□アーカイブ展開

- ランタイムを手軽に導入したい
- 長期利用ではなく、短期的利用が主で、すぐに削除する
- Javaは別途用意する必要がある
(webProfile + IBM Java 8のアーカイブは有り)

12

© 2016 IBM Corporation

Libertyの導入方法は、IIM経由とアーカイブ展開の2パターンがあります。
本スライドでは、Libertyを導入する際の指針について説明します。

満たしたい要件にあった方を選択してください。

【IIM経由で導入する】

単一のツールで、必要となるリポジトリの指定によりFix適用や、ロールバックが行えます。このため、Libertyの導入だけでなく、その後のFix適用も含めてツールで管理したい場合に有効です。

また、IBM JavaやIHS、Plug-inなどが必要な場合、これらはIIM経由での導入が必須となります。このため、Libertyを含めて単一のツールのみで導入できるメリットがあります。

【アーカイブ展開】

アーカイブを展開するだけであるため、手軽に、スピーディーに導入したい場合に有効です。

例えば、一部のユーザーに対して実施するトライアル(Beta期間)など短期間での利用の場合にも有効です。

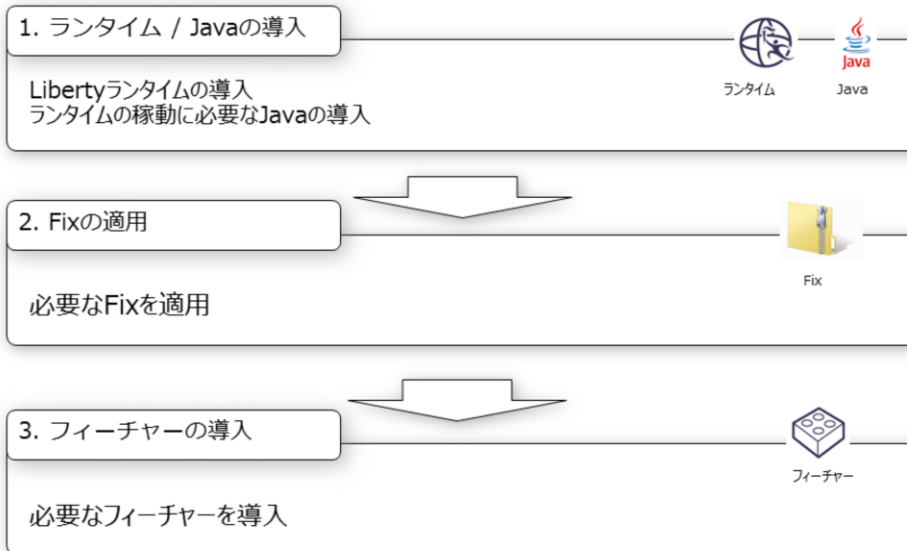
※ただし、IBM Javaは別途導入する必要があります。IHSやPlug-inなどが必要な場合も別途IIM経由での導入が必要になります。(webProfileフィーチャーと共にIBM Java 8が同梱されたアーカイブはあります)

【参考】導入方法による差異 (IIM経由 or アーカイブ展開)

	Installation Manager	アーカイブ展開
エージェントなしでの導入	No	Yes
任意のFixPackレベルに直接上げられるか？	Yes	Yes
FixPackの上書き導入	Yes	No
Interim fixの上書き導入	Yes	Yes
FixPack/iFixのロールバック	Yes	No
製品のエディションのアップグレード	Yes	Yes
組み込み SDK 提供	Yes	Yes
developer toolsとの統合	No	Yes
フィーチャーの追加	Yes	Yes
Liberty Repository の統合	Yes	Yes
z/OS パッケージの提供	Yes	No
パッケージ Minification のサポート	Yes (生成されたイメージは新しいアーカイブとして扱われ、IIMの管理対象にはなりません)	Yes

http://www.ibm.com/support/knowledgecenter/SSAW57_liberty.com.ibm.websphere.wlp.nd.doc/ae/twlp_inst_top.html

実働環境(テスト・本番環境)への導入関連作業の流れ



実働環境へのLiberty導入関連作業についての基本的な流れは以下です。

1. ランタイム および Javaの導入
2. Fixの適用
3. フィーチャーの導入

最初に、Libertyを導入する方法によって、後続の手順が変わってきます。

導入関連作業の流れー 1-1. ランタイムの導入

1-1. ランタイムの導入



ランタイム

Libertyランタイムの導入

■ 以下の選択肢があります。

◆ **IBM Installation Manager (IIM)**での導入

- メディア、PAサイト、Fix Central から、導入イメージを取得し、IIMでLibertyを導入

※IIMでの導入時に、フィーチャー全部入りを選択可能

◆ **アーカイブ展開**での導入

- メディア、PAサイト、Fix Central から、アーカイブを取得し、展開導入
- Liberty Repositoryから、アーカイブを取得し、展開導入。

※Fix Centralから、フィーチャー全部入りの.jarを入手可能

15

Liberty

【1-1.ランタイムの導入】

2パターンがあります。

- IIM経由での導入
- アーカイブ展開での導入)

それぞれフィーチャー全部入りの状態で導入することも可能で、最初に行っておくことで後の作業が楽になります。

※アーカイブ展開の場合、フィーチャー全部入りの.jarがありますが、ライセンスをお持ちの場合のみに取得可能なオプションとなります。

導入関連作業の流れー 1-2. Javaの導入

1-2. Javaの導入



ランタイムの稼働に必要なJavaの導入

- Libertyランタイムの稼働の前提となるJavaの導入も必要です。
選択肢として以下があります。
 - ◆ **IIM**でLibertyを導入した場合
 - メディア、PAサイト、Fix Central から、IIMでLiberty稼働用のIBM Javaを導入
 - ◆ **アーカイブ展開**でLibertyを導入した場合
 - Liberty Repositoryから、IBM Java 8 付属のLibertyアーカイブをダウンロードし、展開導入 (Windows / Linux のみ)
 - ◆ 自前で用意・導入する
 - AIX / Linuxについては、個別インストールも可能
 - Oracle JDK利用などの場合は、自前で用意

【1-2.Javaの導入】

Libertyを稼働させるために必要となるJavaの導入です。

導入方法は3パターンあります。

- IIM経由
- アーカイブ展開(webProfileのみ)
- 自前のJavaを利用

Libertyは、IBM Javaだけでなく、Oracle JDKも利用可能です。(ただしJavaのサポートが必要な場合は、IBM Javaを使用してください。)

導入関連作業の流れー 2. Fixの適用

2. Fixの適用

必要なFixを適用



Fix

- Fix CentralからFixをダウンロードします。
- 導入した方法に合わせてFixを適用します。
 - ◆ **IIM**でLibertyを導入した場合
 - FixPack/Interim fix共にIIMを使用してFixを適用
 - ◆ **アーカイブ展開**でLibertyを導入した場合
 - FixPackの場合は、導入時と異なるディレクトリに展開し、必要な設定（※1）を移行
 - Interim fixの場合は、jar展開で導入（その他注意点はfixに含まれるreadmeに従う）

※1 WLP_USER_DIRとWLP_OUTPUT_DIR 変数相当のディレクトリ下 が対象（変数については「構成」の章）詳細は以下参照
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_inst_fixpack_zip.html

【2.Fixの適用】

Fixの適用には、2パターンあります。

- ・ IIM経由
- ・ アーカイブ展開

IIM経由でLibertyを導入した場合は、
Fixpack/Interim fix共にIIM経由でのFix適用が必須になります。

アーカイブ展開でLibertyを導入した場合は、
元の導入ディレクトリとは異なるディレクトリに新しいFixpackを展開します。
すなわち、新規にLibertyを導入する形式になりますので、既存の構成を再利用する場合には必要な設定の移行が必要です。
(後半の「構成」セッションを参照)

導入関連作業の流れー 3. フィーチャーの導入

3. フィーチャーの導入



フィーチャー

必要なフィーチャーを導入

IIM でLibertyを導入した場合

- IIMで、フィーチャーの追加導入が可能

IIM / アーカイブ展開 共通

- フィーチャーの追加導入方法
 - ◆ installUtility コマンド
 - ◆ 例) \$ bin/installUtility install adminCenter-1.0
- ① コマンド実行すると、
- どこからもってくる？
 - ◆ Liberty Repository
 - インターネットアクセス可の場合
 - ◆ LARS
 - インターネットアクセス不可の場合
 - 別途セットアップが必要だが、複数サーバーで共有可能
 - ◆ Local Repository
 - サーバーのファイルシステム上に置いたもの
 - そのサーバーでのみ利用可能
- ② デフォルトでは、Liberty Repositoryから取得
- ②' オプション指定で、LARS or Local Repositoryも選択可

【3.フィーチャーの導入】

Libertyの導入時に、フィーチャー全部入りを選択することで、本手順を省力化することが可能です。

フィーチャーを追加導入する際には、2つのパターンがあります。

- ・ IIM経由での追加導入
- ・ コマンドによる追加導入

IIM経由での追加導入は、

IIM経由でLibertyを導入した場合のみに利用できます。導入時と同様のウィザードで、フィーチャーの追加導入が可能です。

コマンドによる追加導入は、

IIM経由、アーカイブ展開のいずれの方法でLibertyを導入した場合でも利用できます。

スライド上に示しているinstallUtilityコマンドで任意のフィーチャーを指定し、導入することが可能です。

取得元は、3つあります。

- ・ Liberty Repository (デフォルト、インターネットアクセスが必要)
- ・ LARS (別途、用意が必要)
- ・ Local Repository (別途、用意が必要)

【参考】 installUtilityとは

□ Libertyにアセットの導入、削除、検索ができるコマンド

□ コマンドの構文 `installUtility action [options]`

- download
 - リポジトリからアセットをローカルのファイルシステムに保存
- find
 - リポジトリからアセットを検索
- install
 - リポジトリにあるアセットをLibertyにインストール
- testConnection
 - リポジトリに接続できるか確認
- uninstall
 - Libertyに導入済のアセットをアンインストール
- viewSettings
 - リポジトリの設定を確認
 - リポジトリの設定は、`${wlp.install.dir}/etc/repositories.properties` ファイルに定義

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_command_installutility.html

【参考】Liberty Repositoryからのアセット導入方法

導入方法 アセット	installUtility コマンド	featureManager コマンド	configUtility コマンド	Installation Manager	WDT (WebSphere Developer tools)	WASdev site (Liberty Repository)
アドオン	✓	✓		✓	✓	✓
管理スクリプト						✓
構成スニペット			✓		✓	✓
フィーチャー	✓	✓		✓	✓	
オープンソース 統合	✓			✓	✓	✓
製品				✓	✓	✓
製品サンプル	✓			✓	✓	✓
ツール					✓	✓

【参考】LARS と Local Repository

ローカルリポジトリとして以下を構成して利用できます

□LARS (Liberty Asset Repository Service)

- Liberty Repository相当のものをオンプレミス環境に構築可能
- 社内環境など、Internet越しにLiberty Repositoryにアクセスできない環境において有用
- 別途導入・構築する必要あり <https://github.com/WASdev/tool.lars>

□Local directory-based Repository (Local Repository)

- ローカルマシンのファイルシステムに直接アセットを配置
- ファイルシステムへのアクセスになるので、そのマシン上でのみ利用可能
- installUtility downloadコマンドで作成可能

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/twlp_ins_installUtilitydl.html

導入後の確認方法

□バージョン確認 (productInfo version)

```
$ ./productInfo version
製品名: WebSphere Application Server
製品バージョン: 16.0.0.2
製品エディション: ND
```

□導入済フィーチャーの確認 (productInfo featureInfo)

```
$ ./productInfo featureInfo
adminCenter-1.0 [1.0.0]
apiDiscovery-1.0 [1.0.0]
appClientSupport-1.0 [1.0.0]
appSecurity-1.0 [1.1.0]
appSecurity-2.0 [1.0.0]
```

} フィーチャー一覧

□適用済みのiFixを確認 (productInfo version --ifixes)

22

© 2016 IBM Corporation

Liberty導入後の確認方法についていくつか示します。

- ・製品名・バージョン・エディションを確認

bin/productInfo version

- ・導入済のフィーチャー一覧を確認

bin/productInfo featureInfo

- ・適用済iFixを確認

bin/productInfo version --ifixes

まとめ: 導入方法選択の指針 (再掲)

□IIM経由

- Fix適用をツールで管理したい
- バージョンのロールバックをやりたい
- IBM JavaやIHSが必要な場合、IIMの利用が必要
- ランタイム、Java、IHSなどをまとめて導入したい

□アーカイブ展開

- ランタイムを手軽に導入したい
- 長期利用ではなく、短期的利用が主で、すぐに削除する
- Javaは別途用意する必要がある
(webProfile + IBM Java 8のアーカイブは有り)

23

© 2016 IBM Corporation

(再掲)

Libertyの導入方法は、IIM経由とアーカイブ展開の2パターンがあります。
本スライドでは、Libertyを導入する際の指針について説明します。

満たしたい要件にあった方を選択してください。

【IIM経由で導入する】

単一のツールで、必要となるリポジトリの指定によりFix適用や、ロールバックが行えます。このため、Libertyの導入だけでなく、その後のFix適用も含めてツールで管理したい場合に有効です。

また、IBM JavaやIHS、Plug-inなどが必要な場合、これらはIIM経由での導入が必須となります。このため、Libertyを含めて単一のツールのみで導入できるメリットがあります。

【アーカイブ展開】

アーカイブを展開するだけであるため、手軽に、スピーディーに導入したい場合に有効です。

例えば、一部のユーザーに対して実施するトライアル(Beta期間)など短期間での利用の場合にも有効です。

※ただし、IBM Javaは別途導入する必要があります。IHSやPlug-inなどが必要な場合も別途IIM経由での導入が必要になります。(webProfileフィーチャーと共にIBM Java 8が同梱されたアーカイブはあります)

Bluemix (PaaS) におけるLiberty導入

□Liberty for Java

- .warをデプロイするだけでJava EEアプリが稼働
- 他サービス(DB/分析/Watson/etc.)のバインド

□Libertyコンテナ

- DockerHubで提供しているLibertyイメージを使用して構成
- Libertyに限らず独自イメージをpushして、コンテナ構成も可能
※導入するSW/MWなどのライセンスの要否は別途ご確認ください

□WAS for Bluemix

- Liberty環境が構成済のVMインスタンスを提供
- SSHログインも可能で、OS設定等もいじれる柔軟性

上記のいずれも、数クリックで構築が可能。数十秒～数分以内

24

© 2016 IBM Corporation

IBMのクラウド環境であるBluemixでは、3パターンの形でLiberty環境を構築できます。

以下に紹介するBluemixサービスを活用することで、クラウド上で、迅速に開発・検証・サービス提供を行うことが可能です。

・ Liberty for Java

Cloud FoundryベースのJavaビルドバックでLibertyを採用しています。Webアプリ(.war)をデプロイするだけで、Liberty上稼働するサービス提供が可能です。

またアプリサーバーだけでなく、DBや分析サービス、その他様々なBluemix上のサービスと容易に連携が可能であり、必要となるコンポーネントを組み合わせることでアプリ開発を進めることができます。

・ Libertyコンテナ

LibertyのDockerイメージを使用して、Bluemix上でDockerコンテナ稼働させることができます。

複数台構成、可用性構成なども柔軟に設定できます。

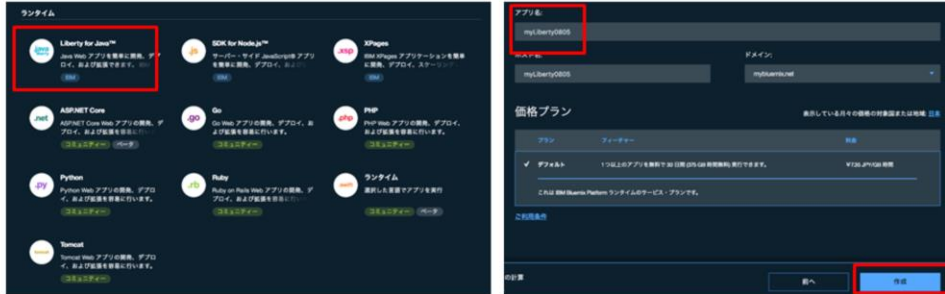
・ WAS for Bluemix

Liberty環境が構成済のVMインスタンスを提供します。SSHのログインも可能で、OS設定も変更でき柔軟性が高いBluemixサービスです。

また、Libertyだけでなく、traditional WASも選択可能であるため、要件にあったランタイムの検証にも活用できます。

Bluemix: Liberty for Java の導入

1. Bluemixランタイム一覧から「Liberty for Java」を選ぶ
2. 任意のアプリ名(ユニーク値)を指定し「作成」をクリック



- デフォルトで、Java EE 7 Web Profileフィーチャーが有効化
BluemixでサポートされるLibertyフィーチャー
○ <https://new-console.ng.bluemix.net/docs/runtimes/liberty/libertyFeatures.html>
- ローカル環境にある既存Libertyサーバーのデプロイも可能
(コマンド実行例) `$ cf push <yourappname> -p wlp/usr/servers/defaultServer`
○ <https://new-console.ng.bluemix.net/docs/runtimes/liberty/index.html>

25

© 2016 IBM Corporation

【Liberty for Java】

デフォルトでは、Java EE 7 Web Profileフィーチャーが有効化されたランタイム環境が生成されます。

Java EE 7 Full Platformのフィーチャーも使用可能です。使用可能なフィーチャーは以下を参照ください。

<https://new-console.ng.bluemix.net/docs/runtimes/liberty/libertyFeatures.html>

BluemixのLiberty for Javaランタイムは、アプリをデプロイするだけで、すぐに外部に向けてサービス提供が可能です。

更に、サーバー構成ファイルなどローカルの開発PCで用意したLiberty環境をBluemix上にデプロイすることも可能です。

詳しくは以下を参照ください。

<https://new-console.ng.bluemix.net/docs/runtimes/liberty/index.html>

Bluemix: Libertyコンテナ の導入

1. Bluemixコンテナ一覧から「ibmliberty」を選ぶ
2. 任意のDockerイメージを選択
(イメージ毎に含まれるフィーチャー・セットが異なる)
 - latest / javaee7 / webProfile6 / webProfile7
3. 単一 or スケーラブル(複数) を選択
4. 単一構成の場合、コンテナ名を指定し「作成」をクリック
 - メモリ、ストレージのサイズ
 - パブリックIPアドレスの割当て
 - パブリック・ポートの指定 } 任意
- スケーラブル構成の場合、コンテナ・グループ名を指定し「作成」をクリック
 - メモリ、ストレージのサイズ
 - インスタンス数
 - HTTPポートの指定
 - 自動リカバリーの使用可能化 } 任意

26

© 2016 IBM Corporation

【Libertyコンテナ】

IBMは、DockerHubにLibertyのDockerイメージを公開しています。これはどなたでも開発用途に無償で利用可能なイメージです。

Bluemix上のLibertyコンテナは、公開しているLibertyのDockerイメージを使用し、Dockerコンテナを構成します。

Bluemix上ですぐに本番利用して頂くことが可能です。

ユーザーは、Libertyイメージ一覧から、任意のフィーチャーセットのLibertyを選択することでコンテナ生成が可能です。

この他、スケーラブル(複数)構成やそれに伴うインスタンス数の指定なども行えます。

Bluemix: WAS for Bluemix の導入

1. Bluemixサービス一覧から「WebSphere Application Server」を選ぶ
2. 任意のサービス名を指定
3. プランを選択 (Liberty Core / Base / ND)
※Libertyが使用可能なのは、Liberty CoreとND。Baseは不可。
4. 「作成」クリック後、traditional or Libertyを選択

□ 使用可能なフィーチャー

- Liberty Coreプランの場合、Java EE 7 Web Profile
- NDプランの場合、Java EE 7 Full Platform

□ 管理画面がデフォルト有効

- adminCenter-1.0が有効化されており、管理画面へのアクセスが可能

□ VM(OS: RHEL)へのログインが可能

- Libertyの関連ディレクトリにアプリをデプロイしたり、server.xmlの編集など柔軟な操作が可能

【WAS for Bluemix】

プランとして、Liberty Core/Base/NDから選択可能です。

ただし、Libertyが使用可能なのは、Liberty CoreとNDプランのみとなりますのでご注意ください。

Liberty CoreとNDの選択基準のひとつは、利用予定のJava EEフィーチャーです。
Java EE 7 Web Profileのみで良い場合は、Liberty Coreプランを。
Java EE 7 Full Platformが必要な場合は、NDプランを選択します。

※NDプランに関しては、LibertyのVMに加えてIHS用のVMを提供します。複数台のLibertyランタイムでクラスター構成を組む場合には、NDプランを選択します。

BluemixにおけるLibertyのバージョン確認

□プロビジョニング前に確認する方法 (2016/8/5時点)

○Liberty for Java

- <https://new-console.ng.bluemix.net/docs/runtimes/liberty/updates.html>

○Libertyコンテナ

- DockerHubのイメージタグで確認可能
(ひとつ前のFixは、8559)

- <https://hub.docker.com/r/library/websphere-liberty/tags/>

○WAS for Bluemix

- <https://console.ng.bluemix.net/docs/services/ApplicationServerCloud/latestUpdates.html>

※ブラウザのコンテンツ言語を英語で確認ください(最新情報の反映が早い)

Bluemix上の各Libertyのバージョンをプロビジョニング前に確認する方法です。
各ページは、最新情報を得るために、ブラウザのコンテンツ言語を英語にして閲覧するようにしてください。

(注意) あくまで、2016/8/5時点の方法であり、かつBluemixの他のランタイムやサービスが共通であるとは限りません。

参考ページ

□ Liberty for Java

- <https://new-console.ng.bluemix.net/docs/runtimes/liberty/index.html>

□ Libertyコンテナ

- https://new-console.ng.bluemix.net/docs/images/docker_image_ibmliberty/ibmliberty_starter.html
- https://new-console.ng.bluemix.net/docs/containers/container_index.html?pos=2

□ WAS for Bluemix

- <https://console.ng.bluemix.net/docs/services/ApplicationServerCloud/index.html>

Bluemix上の各Libertyに関しては、スライドのリンクを参照ください。

構成

ここからは、どのようにLibertyを構成するか、を解説します

アジェンダ

□ 導入

- ランタイムの導入方法
 - IBM Installation Manager / アーカイブ展開
- 実働環境への導入作業の流れ
 - ランタイム・Java / Fix / フィーチャー
- BluemixにおけるLibertyの導入

□ 構成

- 構成要素
 - ディレクトリ構造
 - 使用可能な環境変数
 - 構成ファイル (server.xml他、include可能、などの特徴)
- 構成 の おまけ機能
- 「構成」の運用を考える
 - サーバー構成をアーカイブする
 - ディレクトリ構成運用を考える
 - サーバー構成ファイルの運用を考える
- 構成の流れ と 構成手順
 - 1. スタンドアロン構成
 - 2. シンプル・クラスター構成
 - 3. 高可用性構成

Libertyサーバーの作成

導入後、構成を行うために、まずはサーバーの作成が必要です。

□ コマンドで作成

○ 以下のコマンドを実行！

<導入ディレクトリ>/wlp/bin/server create server_name

```
C:\Users\IBM_ADMIN>C:\IBM\WebSphere\Liberty\bin\server.bat start TEST
サーバー TEST を起動中です。
サーバー TEST が起動しました。

C:\Users\IBM_ADMIN>C:\IBM\WebSphere\Liberty\bin\server.bat status TEST
サーバー TEST は稼働中です。
```

※ serverコマンドの詳細については、「04 運用設計」のセッションで詳細を解説しています

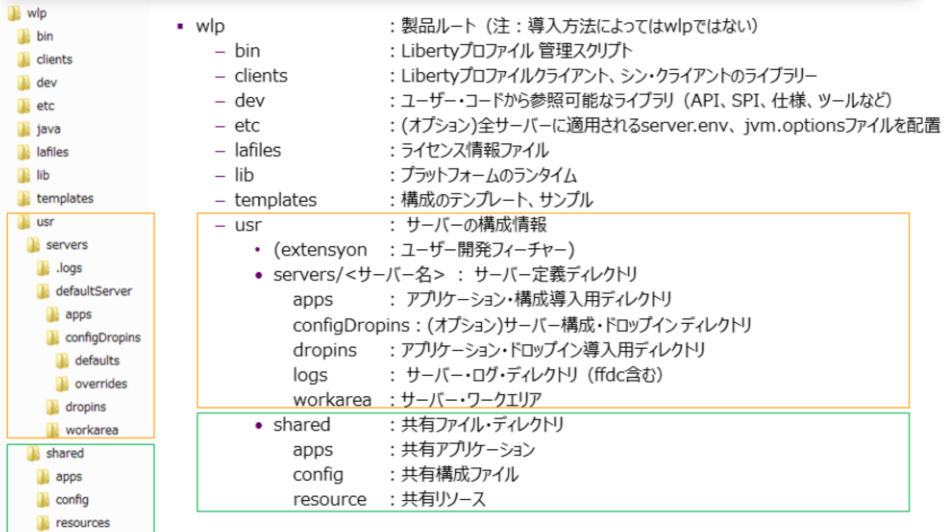


<導入ディレクトリ>/wlp/usr/servers 下に指定した名前でディレクトリが作成されます。

導入が終わったら、構成に先立ってまずは構成の対象となるサーバーを作成する必要があります。

導入すると・・・Libertyプロファイルのディレクトリ構成

以下のようなディレクトリ構成で導入されます。



33

© 2016 IBM Corporation

導入したあとのディレクトリ構成は、上記のようになっています。

大きくは、usr以下に、個別のサーバー設定やアプリケーションが配置されます。

usr下のserver以下には、作成したサーバーの構成情報が格納されます。

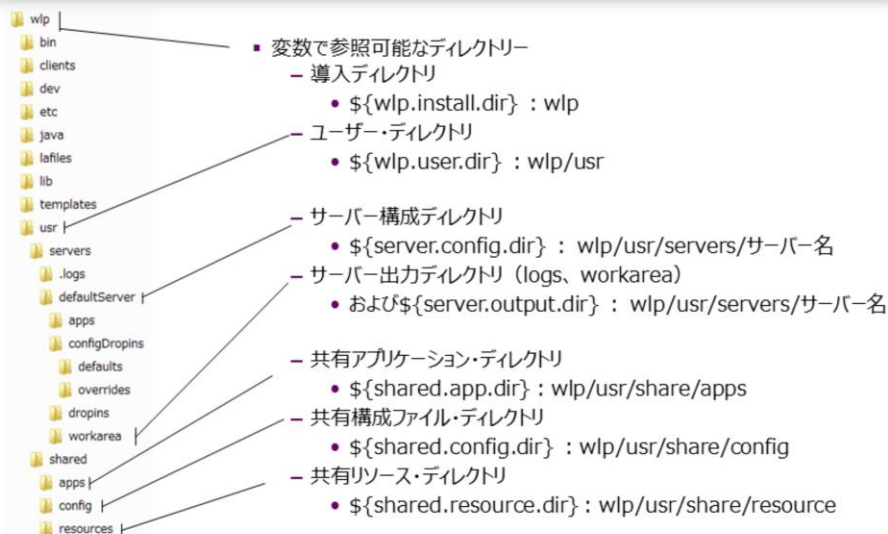
usr下のshared以下には、サーバー内/サーバー間で共有されるアプリケーションやライブラリを配置します。

Liberty プロファイル: ディレクトリーのロケーションおよびプロパティー

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_dirs.html

【参考】Libertyプロファイルで使用可能な事前定義 環境変数

以下のディレクトリを変数に置き換えて指定できます。

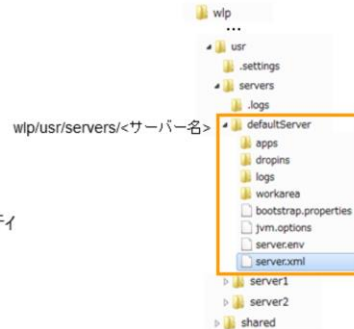


このディレクトリ構成において、主要なディレクトリについては、変数で参照することが出来るようになっています。

Libertyサーバーの構成

(基本的には) 1つの構成ファイル (server.xml) に構成を書きます。

- server.xml [必須]
 - ◆ Libertyのシステム構成ファイル、変更は動的に反映
 - ◆ サーバー構成を作成した際に自動生成
 - ◆ `${server.config.dir}`に配置
 - ◆ デフォルトから変更が必要なもののみ記述する
- サーバー環境定義ファイル[オプション]
 - ◆ bootstrap.properties : サーバーのブートストラップ・プロパティ
 - 初期化用プロパティ、構成ファイル内で使用する変数
 - サーバー初期化時にのみ処理される
 - ◆ server.env: サーバー・プロセスで使用する環境変数
 - JAVA_HOMEの設定など
 - ◆ jvm.options: VM起動時のオプション
 - ヒープサイズの設定、システム・プロパティ、冗長GCなど
 - ✓ `${wlp.install.dir}/etc` および `${server.config.dir}` に配置可能
 - 両方存在している場合は、後者の内容が優先される



35

© 2016 IBM Corporation

それでは、Libertyプロファイルを構成する構成ファイルについてみていきましょう。まず唯一必須となる構成ファイルが server.xmlです。シンプルな構成であればこのserver.xmlのみで構成が完了します。サーバー構成ディレクトリ直下に作成されます。

さらにその他として、追加で作成・利用可能なファイルが3つあります。1つ目が、サーバーの初期化時に読み込まれるbootstrap.propertiesです。構成ファイル内で使用する変数であったり、トレース・ファイル定義だったりを指定することが可能です。サーバー・プロセスが読み込む環境変数を定義する server.envファイルや、JVM起動時オプションを設定することができる jvm.optionsファイルがあります。ヒープサイズの設定やverbose:gcの設定はここで実施します。これらのファイルはサーバー起動時に読み込まれるため動的変更は行われません。

さらに詳細な内容については以下のKnowledge Centerを参照ください。

Specifying Liberty profile bootstrap properties

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_inst_bootstrap.html

Customizing the Liberty profile environment

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_admin_customvars.html

server.xmlの定義

サーバー構成はXML形式で記述します。

- <server>エレメントの子要素として設定を追加
- 以下は最低限必要なエレメント
 - ◆ <featureManager>
 - 利用するフィーチャーを定義
 - ◆ <httpEndpoint>
 - サーバーのホスト名、トランスポートを指定
- システム・デフォルト設定が定義済み
 - ◆ デフォルトを変更する場合のみ記述を追加
 - ◆ 記述がない場合は、デフォルト値が使用される

デフォルトで生成されるserver.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <featureManager>
    <feature>jsp-2.2</feature>
  </featureManager>

  <httpEndpoint id="defaultHttpEndpoint"
    host="localhost"
    httpPort="9080"
    httpsPort="9443"/>
</server>
```

サーバー作成時に明示的な指定が無い場合は、
<wlp.install.dir>/templates/defaultServer が
ベースとなって作成されます。

※defaultServerのテンプレートの内容は、導入されて
いるパッケージによって異なります。

server.xmlについて、より詳しく見ていきます。

右にあるXMLファイルが、デフォルトで作成される server.xmlファイルです（※）。この構成ファイルでJSPとServlet稼動するWebコンテナについて定義が完了しています。従来のWASの構成ファイルをご存知の方はそのシンプルさに驚くことと思います。

簡単にXMLの要素を見ていきます。<server>エレメントの下に<featureManager>エレメントがありこの中でJSPが有効化されています。さらに<httpEndpoint>エレメントがありWebコンテナがListenするホスト名とポートが指定されています。

それ以外は全てシステムのデフォルト値が暗黙的に定義されています。デフォルトを変更する場合のみ明示的にserver.xmlに記載します。記載がない場合は、デフォルト値が有効になるため、構成ファイルをシンプルに保つことができます。

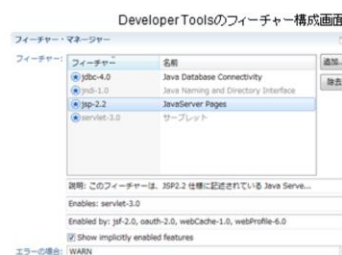
※ server.xmlのデフォルトの設定は、導入したパッケージによって差異があります。

server.xmlの定義： 使用フィーチャーの定義

<featureManager> および <feature> で使用するfeatureを指定します。

- <featureManager> 下の<feature>で
利用するフィーチャーを定義し、有効化
- フィーチャーには依存関係がある
- 前提となるフィーチャーは暗黙的に有効となる
 - ◆ 例) jsp-2.2はservlet-3.0が前提となっている
- 有効となっていないフィーチャーは、
アプリケーションでは使用できない

```
server.xml
...
<!-- Enable features -->
<featureManager>
  <feature>jsp-2.2</feature>
  <feature>jdbc-4.0</feature>
</featureManager>
...
```



37

© 2016 IBM Corporation

<featureManager>の下には、<feature>エレメントを追加して構成していきます。

各<feature>には依存関係が定義されていて、前提となるフィーチャーは暗黙的に有効にされます。たとえば、jsp-2.2に関してはservlet-3.0が前提となっているため、jsp-2.2を有効化すると、暗黙的にservlet-3.0が有効化されます。

server.xmlでは、明示的に有効化したフィーチャーしか記述されませんが、WebSphere Developer Toolsのフィーチャー構成画面（後述）では、暗黙的に有効とされているフィーチャーに関しても表示することが可能です。

フィーチャーの一覧と利用可能なエディションの対応については、以下を参照ください。

Liberty features

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_feat.html

server.xmlの定義: HTTPエンドポイントの定義

<httpEndpoint>でHTTPサーバーとしての定義を行います。

■ <httpEndpoint>

- ◆ サーバーがLISTENするポート、提供するHTTPサーバー・サービスの定義
- ◆ 詳細の設定は、子エレメントで指定可能
 - <accessLogging>
 - アクセス・ログのパス、出力フォーマット、ファイル・サイズ、ファイル数
 - <httpOptions>
 - 読み込み/書き込みタイムアウト、KeepAliveの設定
 - <sslOptions>
 - 使用するSSL定義（SSLレパートリー）の設定
 - <tcpOptions>
 - 非活動タイムアウト

server.xml

```
...
<httpEndpoint
  host="localhost"
  httpPort="9080"
  httpsPort="9443"
  id="defaultHttpEndpoint" />
...
```

Developer Toolsの
HTTPエンドポイント構成画面

38

© 2016 IBM Corporation

次に<httpEndpoint>です。サーバーのWebコンテナーがListenするポートを指定します。

デフォルトではホスト名とポートですが、その他の詳細な設定は、子エレメントで指定していきます。

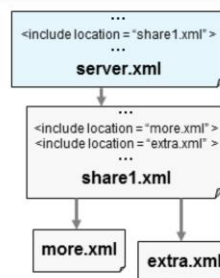
たとえば、Webコンテナーのアクセスログのフォーマットやサイズなどは、<accessLogging>エレメントで指定していきます。他にもHTTPやTCPのタイムアウトなども子エレメントの中で指定していきます。

特にホスト名やポートの明示指定がないと、暗黙的なデフォルト値で起動します。同一ノード内で複数プロセスを起動する場合などは同一のデフォルト値で起動した結果、ポートがバッティングして正常に起動しない、といった事象につばがるケースがありますので、ホスト名、ポートなどの固有情報を明示指定を行うようにしてください。

server.xmlの定義： その他Tips

ファイルのインクルード、変数記述など、柔軟な構成が可能です。

- 柔軟な構成：構成ファイルのインクルードが可能
 - ◆ 複数サーバーで定義を共有する場合、チーム開発などで有効
 - サーバーインスタンスに依存しない定義（DB定義など）
 - ◆ `<include location="dir/file">`で組込むべきファイルを指定
 - dirを相対パスで指定した場合以下の順で検索される
 - ・ `<include>`が書かれた構成ファイルのディレクトリ
 - ・ `.${server.config.dir}`
 - ・ `.${shared.config.dir}`
- 変数を使用した記述（`${〜}`）が可能
 - ◆ 製品事前定義 変数（`${server.config.dir}`等）
 - ◆ `bootstrap.properties`で指定した値
- 時間を指定する属性値には時間単位をつける
 - ◆ 時間単位としては時間（h）、分（m）、または秒（s）が使用可能
 - 一部の設定ではミリ秒（ms）も使用可能
 - たとえば30秒は「30s」と表す
 - 複数の時間単位を使用して「1m30s」というような表記も可能



39

© 2016 IBM Corporation

さらにserver.xmlは、構成ファイルの一部を別XMLファイルに外出しにすることが可能です。

たとえば、データ・ソースへのアクセスするための定義など、サーバー・インスタンスごとに変わらない共通する定義は、別のXMLファイルとして外出しし、`<include>`エレメントでserver.xmlに取り込むことが可能です。このように共通する構成ファイルをコンポーネント化することで管理の容易化を実現できます。またチーム開発などの場合も、サーバー・インスタンスに依存しない共通構成ファイルを配ることで、簡単に構成の共通化を実現することが可能です。

その他、`bootstrap.properties`で定義した変数を構成ファイルに利用することができます。従来のWebSphereではタイムアウトを指定する場合は単位を確認する必要がありましたが、Libertyではタイムアウト関連パラメータは、hやm、sなど単位を指定します。たとえば3分は3mと定義することができますし180sと定義することもできます。

Using include elements in configuration files

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_includes.html

Using variables in configuration files

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_vars.html

Using Ref tags in configuration files

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_refs.html

bsphere.wlp.nd.multipatform.doc/ae/twlp_setup_reftags.html

構成 の おまけ機能

ここからは、構成に関連する便利な機能やTipsを紹介します。

【こんなときどうする?】導入されている製品の情報を調べたい

productinfoコマンドを使用してバージョン情報、フィーチャー一覧、Fix情報などを確認できます。

- productinfoコマンドを使用することで、導入されている様々なLibertyの情報を確認することが出来ます。

productinfo action -[options]

- action として、以下を指定できます。

- compare
 - 別ディレクトリに導入されているLibertyプロファイルと比較して、導入されているifixの差分を表示します。
- featureInfo
 - 導入されているフィーチャーの一覧を出力します。
- validate
 - 導入されているフィーチャーの妥当性を検証します。
- version
 - 製品名とバージョンを表示します。

```
C:\IBM\WAS\WebSphere\defaultServer\wlp\bin>productinfo.bat compare --
apars=P54833 --target=C:\IBM\WAS\WebSphere\liberty\
C:\IBM\WAS\WebSphere\defaultServer\wlpにあるイメージの一部のfixesが
C:\IBM\WAS\WebSphere\libertyにあるイメージに欠落しています。

以下のfixesがC:\IBM\WAS\WebSphere\defaultServer\wlpにあるイメージに存在しますが、
C:\IBM\WAS\WebSphere\libertyにあるイメージに欠落しています。
fix 内の P54833: [8559-wlp-archive-FP61936]
fix 内の P58799: [8559-wlp-archive-FP61936]
fix 内の P59140: [8559-wlp-archive-FP61936]
fix 内の P58097: [8559-wlp-archive-FP61936]

以下のfixesがC:\IBM\WAS\WebSphere\defaultServer\wlpにあるイメージと
C:\IBM\WAS\WebSphere\libertyにあるイメージに存在します。
fix 内の P61836: [8559-wlp-archive-FP61936]
fix 内の P56315: [8559-wlp-archive-FP61936]
すべてのAPARがインストールにあります。
```

```
C:\Users\IBM_ADMIN\IBM\WAS\WebSphere\liberty\bin>productinfo.bat version --verbose
WebSphereApplicationServer.properties:
com.ibm.websphere.productId=com.ibm.websphere.appserver
com.ibm.websphere.productOwner=IBM
com.ibm.websphere.productVersion=16.0.0.2
com.ibm.websphere.productName=WebSphere Application Server
com.ibm.websphere.productInstallType=Archive
com.ibm.websphere.productEdition=IB
com.ibm.websphere.productLicenseType=IPLA
```

```
[ucadmin@uc bin]$ ./productinfo version --verbose
WebSphereApplicationServer.properties:
com.ibm.websphere.productId=com.ibm.websphere.appserver
com.ibm.websphere.productOwner=IBM
com.ibm.websphere.productVersion=16.0.0.2
com.ibm.websphere.productName=WebSphere Application Server
com.ibm.websphere.productInstallType=Archive
com.ibm.websphere.productEdition=BASE_ILAN
com.ibm.websphere.productLicenseType=ILAN
```

41

© 2016 IBM Corporation

Libertyでは導入する方法もバリエーションがあり、このランタイムはどのように導入したのだったか?と思うことがしばしばあるかもしれません。

そのような場合は、このproductinfoコマンドが役に立ちます。他にも、導入されている個別fixの内容を調べたり、ライセンスの状況を調べたり、利用するケースが度々あると思われます。詳細については以下を参照ください。

Liberty: productInfo command

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_productinfo.html

【こんなことも出来ます】jarアーカイブからいきなり起動

パッケージ化されたjarファイルから、解凍せずにそのまま起動できます！

□ 実行可能なJARファイルとしてサーバーがパッケージ化されている事が前提

- アーカイブ生成（windows環境でアーカイブ作成してみました）

```
c:\>C:\IBM\WebSphere\Liberty\bin\server.bat package TEST --include=minify, runnable --archive=C:\IBM\WebSphere\Liberty\usr\servers\TEST\TEST.jar
サーバー TEST をパッケージ中です。
サーバー TEST で内容を照会しています。
IBM J9 VM バージョン pwa6480sr3-20160428_01 (SR3) (ja_JP) で、TEST (WebSphere Application Server 16.0.0.2/wlp-1.0.13.cl160220160526-2258)を起動しています
[AUDIT] CWWKF0001I: サーバー TEST が起動されました。
[AUDIT] CWWKF0012I: サーバーは次のフィチャーをインストールしました。[servlet-3.0, jsp-2.2, ssl-1.0, jndi-1.0, json-1.0, adminCenter-1.0, distributedMap-1.0,
jaxrs-1.1, resConnector-1.0]
[AUDIT] CWWKF0026I: サーバー TEST は、より小さいパッケージを作成する準備ができました。
[AUDIT] CWWKF0036I: 4.549 秒後にサーバー TEST が停止しました。
サーバー TEST のアーカイブを作成しています。
サーバー TEST のパッケージが C:\IBM\WebSphere\Liberty\usr\servers\TEST\TEST.jar で完了しました。
```

- 起動（Linux環境で起動してみました）

```
[ucadmin@uc bin]$ java -jar /mnt/hgfs/work/TEST.jar
ファイルを/home/ucadmin/wlp/extract/TEST_110221261576740/wlpに抽出しています。
すべての製品ファイルを正常に抽出しました。
IBM J9 VM バージョン pxa6470_27sr3fp30-20160112_01 (SR3 FP30) (ja_JP) で、TEST (WebSphere Application Server 16.0.0.2/wlp-1.0.13.cl160220160526-2258)を起
動しています
[AUDIT] CWWKF0001I: サーバー TEST が起動されました。
[AUDIT] CWWKF0005R: アプリケーションの dropsins をモニター中です。
[AUDIT] CWWKF0012I: サーバーは次のフィチャーをインストールしました。[jsp-2.2, servlet-3.0]
[AUDIT] CWWKF0011I: サーバー TEST は、Smarter Planet に対応する準備ができました。
```

□ 留意点

- アーカイブは、テンポラリディレクトリに回答されてフォアグラウンドで起動します。
- ログは標準出力に出力されます。
- デフォルトで 2 PCIはNGです。（テンポラリディレクトリへのトランザクションログの永続化が不可のため）
- 詳細はこちら参照↓

https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_setup_jarserver.html

42

© 2016 IBM Corporation

Libertyはその軽量さが売りのひとつですが、この機能はまさにそれを象徴しています。Libertyをアーカイブする際に、起動可能指定を行うと、なんとアーカイブから直接起動できてしまいます。

とはいえあくまでもテンポラリのディレクトリに解凍した上で起動するわけですが、この機構のため、この場合は利用可能な機能に制限が出てきます。

こちらも詳細については以下を参照ください。

Running a Liberty server from a JAR file

https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_setup_jarserver.html

【こんなことも出来ます】設定のドロップインフォルダ (1/2)

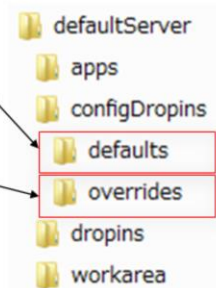
サーバー設定の部分的・動的な上書きが可能です。

- **特定のディレクトリ**に、変更したい設定を記述した設定ファイルを配置することで、server.xmlに手を加えずにサーバー設定を**動的に部分的に上書き**することができます。

- 以下の2つの方法を提供しています。

1. server.xml に定義されていない項目のデフォルト値を変更する
2. server.xml の定義を上書きする

※ overrides > server.xml > defaults > 暗黙的デフォルト値の順に適用されます。



■ 手順

- ◆ 以下のディレクトリを作成します。
 - `usr/servers/server_name/configDropins/overrides` → 設定上書き用
 - `usr/servers/server_name/configDropins/defaults` → デフォルト値設定用
- ◆ 設定ファイルを目的に応じて上記ディレクトリに配置します。
- ◆ 動的に配置が検出され設定が適用されます。

http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_dropins.html

43

© 2016 IBM Corporation

また、server.xmlの変更はモニターされて自動で再ロードされますが、なんとserver.xmlの内容はそのままに、変更したい部分のみを切り出して構成ファイルを作成し、特定のディレクトリに配置することで、一部分のみを動的に上書きする、という事も可能です。

この特定のディレクトリ、というのは、defaultsと、overridesの2種類用意されており、どちらにファイルを配置するかで効き方が変わってきます。

overridesに配置した場合は、単純にserver.xmlの内容を上書きします。

defaultsに配置した場合は、server.xmlに記述されていない設定のデフォルト値を変更することになります。

server.xmlに設定されていない設定項目は、暗黙的にデフォルト値で動作することになりますが、これらの優先順位は以下のようになります。

overrides > server.xml > defaults > 暗黙的デフォルト値

【こんなことも出来ます】設定のドロップインフォルダ (2/2)

■ 例えば・・・

通常のserver.xmlはこの内容ですが・・・

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <featureManager>
    <feature>jsp-2.2</feature>
  </featureManager>

  <httpEndpoint id="defaultHttpEndpoint"
    host="localhost" httpPort="9080" httpsPort="9443"/>

</server>
```

TEST
apps
configDropins
defaults
overrides
dropins

この↓内容のファイルをoverridesに配置すると・・・

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <httpEndpoint id="defaultHttpEndpoint"
    httpPort="9081"/>
</server>
```

配置された内容を検知して動的にListenポートが変更されます。

```
[16/07/29 11:33:43.900 JST] 00000037 com.ibm.ws.tcpchannel.internal.TCPChannel
  始まれ、現在、ホスト {27.0.0.1 (IPv4: 127.0.0.1)}、ポート 9080 の要求を listen しています。 I CWWK00219I: TCP チャネル defaultHttpEndpoint が開
[16/07/29 11:34:58.991 JST] 00000050 com.ibm.ws.config.xml.internal.ServerXMLConfiguration A CWWK0093A: 構成ドロップイン・リソースを処理中です
  : C:\IBM\WebSphere\Library\usr\servers\TEST\configDropins\overrides\port_9081.xml
[16/07/29 11:34:58.998 JST] 00000050 com.ibm.ws.config.xml.internal.ConfigValidator W CWWK0011W: 構成の検証が失敗しました。 httpEndpoint 構成の defaultHttpEndpoint
  インスタンスで矛盾する設定が見つかりました。
  プロパティ httpPort に矛盾する値があります。
  値 9080 が file://C:\IBM\WebSphere\Library\usr\servers\TEST\server.xml で設定されています。
  値 9081 が file://C:\IBM\WebSphere\Library\usr\servers\TEST\configDropins\overrides\port_9081.xml で設定されています。
  プロパティ httpPort に 9081 に設定されます。
```

■ こんなときに便利！

- ◆ パフォーマンステストなどで、一時的にスレッド数を変更したい
- ◆ server.xml の include の代わりに、開発・テスト・本番など、環境によって異なる設定を入れ込みたい

44

© 2016 IBM Corporation

上記は実行例です。

この機能は、例えばパフォーマンステスト時に一時的なパラメータを変更してチューニングの効果を確認したい、という場合や、開発・テスト・本番など、環境によって異なる設定を入れ込みたい場合に便利です。

特にパフォーマンステスト実施時には、テストケースとserver.xmlファイルを関連付けることで、このケースはどのようなパラメータで実施したか、というエビデンスにもなりますので非常に便利です。

【こんなときどうする？】通信をSSLで暗号化したい

securityUtilityでSSL証明書の生成を行い、SSLフィーチャーを組み込みます。

- 以下のコマンドでSSL証明書を生成します。

```
securityUtility createSSLCertificate --server=<サーバー名> --password=<パスワード>
```

```
C:\IBM\WebSphere\liberty\Archive\8556\wlp\bin>securityUtility createSSLCertificate --server=defaultserver --password=password
成功: C:\IBM\WebSphere\liberty\Archive\8556\wlp\bin\server\defaultServer\resources\security\key.kesを作成中です
サーバー defaultserver の SSL 証明書を生成しました。証明書は SubjectDN CN=X0000X.ibm.com, OU=defaultserver, O=ibm, C=us で作成されました。
SSL を使用可能にするには、server.xml に次の行を追加してください。
```

```
<featureManager>
<feature>ssl-1.0</feature>
</featureManager>
<keyStore id="defaultKeyStore" password="XorjLz4sLCgwLTs" />
```

注意！証明書のデフォルトの有効期限は365日！ **重要**

- 上記出力のとおり、server.xmlを編集します。

- SSLでアクセスしてみると...



http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.doc/ae/twlp_sec_create_certificate.html

45

© 2016 IBM Corporation

また、Libertyサーバーとクライアント間の通信を、HTTPSにしたい、という場合は上記の手順をとる事で簡単に構成することが可能です。

ただし、詳細なSSLオプションを設定する必要がある場合は、以下を参照してください。

ここでひとつ大事なポイントがあります。デフォルトで作成した証明書は、有効期限は365日、1年になっています。きちんと認識して運用するか、長めに設定しておくか、留意してください。（尤も、本番運用時には自己署名証明書ではなく厳密な管理を行うことと思いますが・・・）

Liberty プロファイル: SSL 構成の属性

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_ssl.html

【こんなことも出来ます】オリジナルのフィーチャーの開発

独自の、オリジナルのフィーチャーも開発して組み込みます！

- 作業の流れは以下のようになります。
 1. OSGiバンドルを作る
 2. jarを作成する
 3. フィーチャーマニフェストファイルを作成する
 4. バンドルを`${wlp.user.dir}/extension/lib` に配置
 5. フィーチャーマニフェストファイルを`${wlp.user.dir}/extension/lib/features`に配置
 6. (ローカライズしている場合) `${wlp.user.dir}/extension/lib/features/l10n` にローカライズファイルを配置する

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/twlp_feat_example.html

さらに、Libertyではオリジナルのフィーチャーも作成し、組み込むことが可能です。

【こんなことも出来ます】 Libertyサーバーにファイルを転送する

LibertyサーバーのMbean操作によってサーバーとのファイル転送を行うことが出来ます！

- restConnectorフィーチャーを有効にすると、以下2つのMbeanが利用可能になります。

- FileService

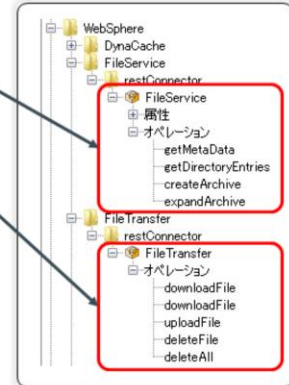
- ディレクトリのリスト、ファイルのメタデータおよびアーカイブの作成・解凍を行う。(WLP_HOME以下のみ対象)

- FileTransfer

- ファイルのダウンロード、アップロード、削除が可能。
(ソース/ターゲットファイルの指定、オプション指定でファイル进行操作)

- これによって、アプリケーションの置き換え(dropinsへの配置 or server.xmlの置き換えと任意のディレクトリへの配置)や設定変更を行うことが可能です。

- ◆ Collective対応のファイル転送mbeanも用意されており、これを利用することでmemberが存在していないcollective内のホストへもファイルを転送する事が出来ます。



Jconsoleで表示されるファイル転送系mbean

https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/cwlp_file_transfer.html

Libertyには、ファイルの送受信を行う機能もあります。restConnectorフィーチャーを利用することで、ファイル操作関連のMbeanと、ファイル転送関連のMbeanが有効になります。

これによって、リモートのLibertyに構成ファイルを転送してそのまま構成変更してしまったり、することが出来ます。

さらに、Collective対応のmbeanも用意されていますので、この機能を利用した環境構築も不可能ではありません。(この方法は運用のセッションで紹介しています。)

【こんなことも出来ます】設定サンプルがほしい – configUtility コマンド

configUtilityコマンドで、設定サンプルを参照しながら簡単にサーバー構成が可能です。

- 設定のサンプルをリポジトリから取得することができます。
- 以下のようなことができます。

◆ find

- Liberty環境に適用可能な featureをLibertyRepositoryから検索します。

◆ install

- featureをインストールします。
- 右は例ですが、設定が必要な箇所が
`<!-- TODO: -->`
 で示されています。



出力例

```
C:\Users\IBM_ADMIN>C:\IBM\Websphere\Liberty\Archive\85506\wlp\bin\configUtility.bat install
remoteAdministration

検索された構成スニペットをダウンロードしています...

<server description="Remote Administration Sample Configuration">
  <!-- NOTE: This file is for reference only. -->
  <!-- Enable restConnector-1.0 feature -->
  <featureManager>
    <feature>restConnector-1.0</feature>
  </featureManager>

  <!-- Simple administrative security configuration. -->
  <!-- TODO: Set the security configuration for Administrative access -->
  <quickStartSecurity userName="${adminUser}" userPassword="${adminPassword}">

  <!-- TODO: Set the SSL keystore password -->
  <keyStore id="defaultKeyStore" password="${keystorePassword}">

  <!-- TODO: Set HTTP Endpoint attributes -->
  <httpEndpoint id="defaultHttpEndpoint"
    host=""
    httpPort="9080"
    httpsPort="9443"/>

  <!-- TODO: Use readDir and writeDir to specify directories that remote
  clients are allowed to have read and write access. There can be
  multiple readDir and writeDir elements. Replace writePath and readPath
  variables with your choice of locations or remove them if not needed. -->
  <remoteFileAccess>
    <writeDir>${writePath}</writeDir>
    <readDir>${readPath}</readDir>
  </remoteFileAccess>
</server>
```

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_configutil.html

48

© 2016 IBM Corporation

便利情報が続きますが、server.xmlの設定の再、設定サンプル/テンプレートをダウンロードして参照することができます。

configUtilityというコマンドを利用して、installすると、サンプルが出力されますので、これをベースにして、`<!-- TODO: -->`で示された部分を編集することで設定を行うことができる便利コマンドです。

Liberty profile: configUtility command

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_configutil.html

【こんなことも出来ます】スキーマを確認したい – ws-schemagen.jar ツール

- 以下のコマンドで、Libertyプロファイルの設定ファイル（server.xml）のスキーマファイル出力することができます。

```
java -jar ws-schemagen.jar [options] outputFile
```

- オプションとして、以下が指定できます。

- ◆ **--encoding=charset**
 - 出力ファイルのキャラクタセットを指定します。
- ◆ **--locale=language**
 - 言語指定が可能です。
 - ISO-639の2文字の言語コードと、ISO-3166のCountryコードを、
" "で繋いで指定します。

出力例

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ext="http://www.ibm.com/xmlns/sdd/schema/annotation/ext">
  <xsd:complexType name="com.ibm.ws.security.authentication.tai.interceptor">
    <xsd:annotation>
      <xsd:documentation>トラスト・アソシエーション・インターセプターを定義します。 </xsd:documentation>
      <xsd:appinfo>
        <ext:label>トラスト・アソシエーション・インターセプター </ext:label>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:complexType>
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="library" type="com.ibm.ws.classloading.sharedlibrary">
      <xsd:annotation>
        <xsd:documentation>共有ライブラリー構成の ID の参照。 </xsd:documentation>
        <xsd:appinfo>
          <ext:label>共有ライブラリー </ext:label>
        </xsd:appinfo>
      </xsd:annotation>
    </xsd:element>
  </xsd:choice>
</xsd:schema>
```

Libertyプロファイルはいろいろな便利ツールが用意されていますが、さらにはserver.xmlファイルのスキーマファイルまで出力するコマンドが用意されています。

Liberty プロファイル: コマンド・プロンプトから Liberty プロファイル構成スキーマを生成

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_configutil.html

【こんなことも出来ます】アクセス先DBのDDLを生成したい - ddlGen ユーティリティ

データベースへアクセスする設定がされている場合、DDLを生成することができます。

- データベースにアクセスするフィーチャーを利用している場合、DDL（Data Definition Language）を生成します。
- 前提
 - ◆ ddlGen ユーティリティが検索するサーバーのパスを環境変数 WLP_USER_DIR で指定しておきます。
 - ◆ ddlGenユーティリティを実行する前に、対象のサーバーを起動しておく必要があります。
 - ◆ localConnector フィーチャーを有効化する必要があります。

```
ddlGen generate <server_name>
```

```
C:\Users\IBM_ADMIN>C:\IBM\WebSphere\Liberty\bin\ddlGen.bat generate batch_disp1
CWWKD0107I: 要求された DDL は、次のディレクトリに生成されました:
C:\IBM\WebSphere\Liberty\usr\servers\batch_disp1\ddl
```

まだまだ続きます。

DBアクセスを行うフィーチャーを利用している場合には、アクセス先DBのDDLを生成するというユーティリティまで用意されています。

「構成」の運用を考える

構成運用の重要機能：サーバー構成をアーカイブする

サーバーのアーカイブを作り、導入イメージやバックアップとして利用できます。

□ 以下のコマンドでアーカイブを作成

```
<導入ディレクトリ>/wlp/bin/server package server_name ¥
--archive= アーカイブファイル名.[ zip | jar ] ¥
--include= [ all | usr | minify ] [ runnable ]
```

□ --include オプションで、アーカイブに含める内容の選択が可能です。

- All
 - Wlp以下すべてをアーカイブに含めます。
- Usr
 - Libとusr以下を含めます
- Minify
 - 稼動に必要なもののみ選択的にアーカイブします。
- runnable
 - この指定をすることで、アーカイブから直接起動できるようになります。

各オプション指定での出力されたアーカイブのサイズサンプル

TEST_all.zip	476,373 KB
TEST_mini.zip	93,155 KB
TEST_mini_run.zip	46,806 KB
TEST_usr.zip	160,201 KB

Libertyの構成に関する運用を考えるにあたっては、サーバーのパッケージ機能が非常に重要です。

作成したアーカイブを新規サーバーのテンプレートとしたり、バックアップとしたりすることで、運用を効率的に行うことが出来るようになります。

詳細については以下を参照ください。

Packaging a Liberty server from the command line

https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_package_server.html

【こんなときどうする】アーカイブの形式はzip ? jar ?

- Unixにおいてはserver packageコマンドでアーカイブする際に、jarを指定した場合とzipを指定した場合では、(unzipコマンドの実装に依存して) 解凍時のファイルパーミッションに違いが生じます。
 - zipの場合はパーミッションが保管されませんので注意が必要です！

jar

```
[wasadmin@oms94 jar]$ java -jar templateServer.jar /work/jar/
Extracting files to /work/jar/wlp
Successfully extracted all product files.
[wasadmin@oms94 jar]$ ls -la /work/jar/wlp/bin/server
```

-rwxrwxr-x

zip

```
[wasadmin@oms94 zip]$ unzip templateServer.zip
Archive: templateServer.zip
  creating: wlp/
      :
[wasadmin@oms94 jar]$ ls -la /work/zip/wlp/bin/server
```

-rw-rw-r--

■ ダウンロードしたファイルおよびアーカイブを使用した Liberty のインストールとアンインストール

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_inst.html

アーカイブを扱う際のTipsです。

Jarファイルを解凍した場合は、ファイルのパーミッションはそのまま保管されますが、zipの場合は保管されませんので、解凍後もう一手間必要になります。ご注意ください。

ディレクトリ構成運用を考える

□ ポイント1：アプリケーション配置ディレクトリをどう選択すべきか？

○ 選択肢

1. server.xmlで明示指定
2. dropinsディレクトリに配置し動的更新

○ 検討の指針

- 本番環境など厳密な定義を必要とする場合は「1」、開発段階の簡易デプロイには「2」を！
- 詳細は、「04. 運用管理」参照

□ ポイント2：どこがユーザーがメンテナンスすべき部分か？

○ 検討の指針

- ログ、構成、アプリケーションが主要な対象
 - ログ、構成はサーバーディレクトリ(\$server.config.dir)に含まれる
 - アプリケーションはserver.xmlから明示指定（もしくはdropins）、いずれの場合もソフトウェア構成管理ツールで別管理を行う
- アーカイブ展開でLibertyをインストールしている場合は、Fix適用時に抜き出す必要があるため、これらの可搬性を意識する
 - 詳細は、次ページ「ポイント3：Fix適用時のディレクトリ構成を参照」
- 構成変更全体に関しては、「04. 運用管理」参照

Libertyでは、ディレクトリ構成をどのように監視していくかという運用も検討すべき事項の一つになります。幾つかポイントを挙げます。

1つ目のポイントとしては、アプリケーションを配置するディレクトリをどのように管理するか、という点です。Dropinに動的に配置して動的にロードする、という方法は、主に開発時などに開発環境で非常に便利な使い方になるかと思います。

一方で、明示的に配置場所を指定して、自動的なロードを行わないといった選択肢もあります。実働環境などでは、不用意にアプリケーションの置き換えが行われない用にこちらの方法を採った方が良いでしょう。

2つ目のポイントは、Libertyを構成するディレクトリのうち、どこがメンテナンスの対象になる部分か、という点です。ログファイルの書き出し先やアプリケーションの配置先などを明確にしておく必要があります。

特にアーカイブ導入を行っているケースで、Fix適用時などに問題になります。この点は、次ページでも詳細を解説します。

ディレクトリ構成運用を考える

□ ポイント3 : Fix適用時 (1/2)

○ 検討の指針

- IIMで導入を行っている場合は、Fix適用はIIMに管理を任せる
- アーカイブ展開導入を行っている場合は、以下に留意する

○ アーカイブ展開導入している場合のFix適用手順

1. jar, zipいずれの場合も、**既存ディレクトリとは異なる新しいパスに導入(解凍)を行う**
 - Jarは自己解凍実行形式なので、java -jar コマンドで解凍 / Zipは任意のアーカイブで解凍
2. **旧パスから新パスに以下の必要なファイルをコピー**
 - \${wlp.user.dir} (wlp/usr) : 共用リソース含むサーバー構成ファイルの格納場所
 - \${server.output.dir} (wlp/usr/servers/サーバー名) : サーバーによって生成されるログファイルなどの出力場所 (デフォルトだとWLP_USER_DIR に含まれる。)

ポイント3つ目は、Fix適用の方法です。これがどのようにディレクトリ構成と関わるか、というと、IIMで導入した場合は既存の導入ディレクトリに履歴を保持しながら被せる様に適用可能であるのに対して、アーカイブからの導入は、新規導入と同じようなイメージで別ディレクトリに導入する事になってしまうわけです。(次ページの図参照)

この場合、ポイント2で解説した、ユーザー資産が保管されるディレクトリを、Fix適用(=新規導入)した別ディレクトリに移す、もしくは参照させる方法が必要になります。

そのままファイルをコピーするというのもひとつの方法ではありますが、環境変数経由でこれらのディレクトリを指定することによって、Libertyとは別のディレクトリにログ等のユーザー管理資産を配置する事が可能になります。

こうしておくと、Fix適用時にもファイルコピーなどを行うことなく、新規導入した方のLibertyランタイムの環境変数から参照して、設定なども補完する事が出来ます。

ディレクトリ構成運用を考える

□ ポイント3 : Fix適用時 (2/2)



これらのディレクトリは、以下の環境変数にパスを指定しておくことで、デフォルトから変更可能です。つまり、Libertyの導入パス外に設定しておくことで、Fix適用時に影響を受けないようにすることも可能です。

- ・WLP_USER_DIR
- ・WLP_OUTPUT_DIR

上記は、前頁で説明したFix適用時のディレクトリ指定にイメージ図です。

サーバー構成ファイルの運用を考える

□ server.xmlの構成情報をどう持つか？

- ポイントは、複数サーバー構成とした場合の個別情報の分離
- サーバー単位に1つずつ専用のファイルを用意しても良いが、共通設定部分は1つのファイルを共用した方が変更効率がよく、編集ミスも少なくできる（と考えられる）

□ 利用可能な機能

- server.xml の `<include location="dir/file">`
- configDropins で個別部分を上書き

□ 構成情報分離の例

- 例1. 複数サーバー環境でサーバ固有情報を分離
 - サーバ固有情報を別ファイル化
 - 各サーバー毎に別のファイルを配置
 - 項目例：`<httpEndpoint>`情報
- 例2. アプリチームで管理する情報を分離
 - 別ファイルに分離し管理をアプリチームに任せる
 - 複数サーバーでも同一ファイルを配置
 - 記述する項目は調整しておく

次は、サーバー構成ファイルの運用についてです。

サーバー構成も、include指定することで構成を別ファイルに分離する事が出来たり、configDropinsフォルダへの配置による動的更新など幾つかの便利な機能がありました。

これらを有効活用しつつ、運用のあるべき姿をどう実現していくか、というのを考える必要があります。

例えば、テスト環境を複数持つ場合は環境ごとに異なる部分をincludeで別だしするとか、チーム単位で編集可能な項目を別ファイルに分けて定義して、分散管理を行うとか、様々な方法が考えられます。

構成の流れ と 構成手順

ここからは、実際にトポロジーのセッションで紹介したトポロジー例を、どのように構成していくか、流れと手順を説明します。

【再掲】 トポロジー選択指針

各トポロジーの機能的な制約や構成・運用管理の負荷、
実現したいサービスレベルの要件に応じてトポロジーやシステム構成を検討すべき！

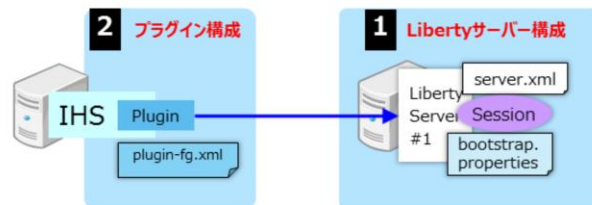
トポロジー	トポロジーの説明	考慮事項
スタンドアロン構成	同じアプリケーションを稼働させていても各Libertyサーバーは完全に独立している構成	①セッション共有や負荷分散が不可 ②構築/構成は容易 ③運用管理は個々に実施 ④障害発生時の利用ユーザーにはエラー
シンプル・クラスター構成	クラスタリング機能(Collective)を使わずにLibertyサーバー間で負荷分散やセッション情報共有を行う構成	①セッション共有や負荷分散が可能 ②構築/構成は比較的容易 ③運用管理は個々に実施 ④障害発生時の利用ユーザーはフェイルオーバーによりサービス継続可能で縮退運用
高可用性構成 (NDエディション限定)	クラスタリング機能(Collective)を使ってLibertyサーバー間で負荷分散やセッション情報共有を行う可用性の高い構成	①セッション共有や負荷分散が可能 ②構築/構成は比較的困難 ③運用管理は一元的に集中管理が可能 ④障害発生時の利用ユーザーはフェイルオーバーによりサービス継続で非縮退運用 ⑤業務提供用Libertyサーバー(Collective Member)以外に管理用Libertyサーバー(Collective Controller)も必要 ⑥動的スケーリングや無停止でのリリースなど煩雑な運用管理の自動化や高いサービスレベルの維持が可能

トポロジーセッションで紹介したトポロジー例を再掲します。

スタンドアロン、シンプル・クラスター、高可用性構成、の3つを挙げて解説します。

1. スタンドアロン構成

「スタンドアロン構成」構成の流れ



Libertyプロファイルでスタンドアロン構成をとる場合の流れを説明します。

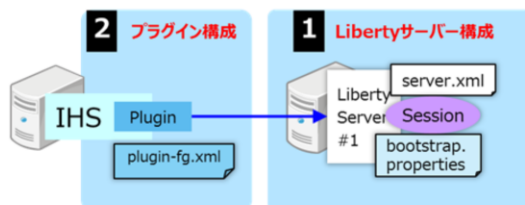
まず、フルプロファイルと同様、通常通りにLoad Balancer/IHS/PluginといったLibertyサーバー前段のコンポーネントを導入します。

以降の大まかな作業の流れは以下の通りです。

- (1) Libertyサーバーを稼働させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、サーバーの設定を行います。
- (2) プラグインを構成します。

これらの各ステップに関する詳細な手順は次ページ以降でご説明します。

スタンドアロン構成方法 (1) Libertyサーバー構成



1. Libertyプロファイルを導入

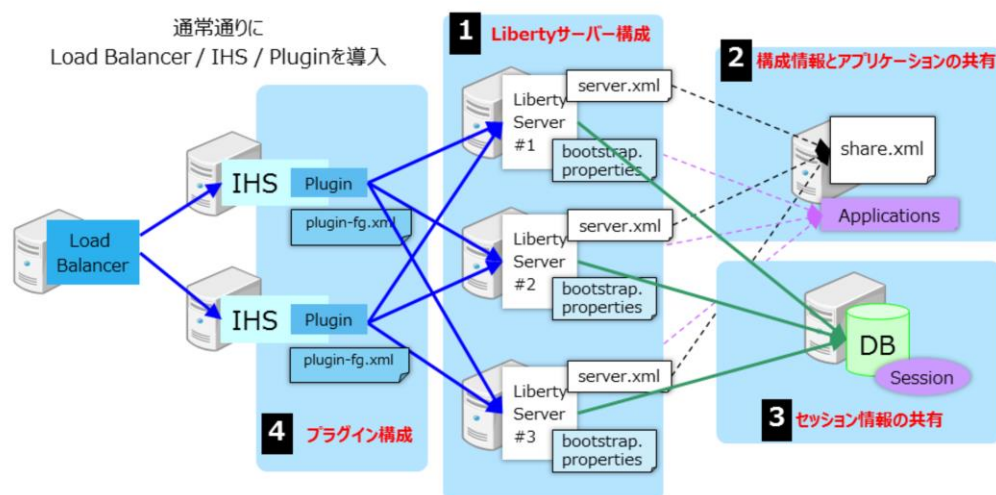
2. plugin-cfg.xmlファイルを生成して配置

こちらは、Libertyサーバーを稼働させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、サーバー固有の設定をする手順の例です。

1. Libertyサーバー上に Libertyプロファイルを導入します。
2. Libertyサーバーを作成します。
3. `server.xml`に対して必要な構成を行います。
4. WDTやmbeanのオペレーションを実行することで`plugin-cfg.xml`を生成します。
5. `plugin-cfg.xml`をIHSサーバー上の規定の場所に配置します。

2. シンプル・クラスター構成

「シンプル・クラスター構成」構成の流れ



64

© 2016 IBM Corporation

Libertyプロファイルでシンプル・クラスターを構成する場合の構成方法についてご説明します。

まず、フルプロファイルと同様、通常通りにLoad Balancer/IHS/PluginといったLibertyサーバー前段のコンポーネントを導入します。

以降の大まかな作業の流れは以下の通りです。

(1) Libertyサーバーを稼働させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、サーバー固有の設定をします。

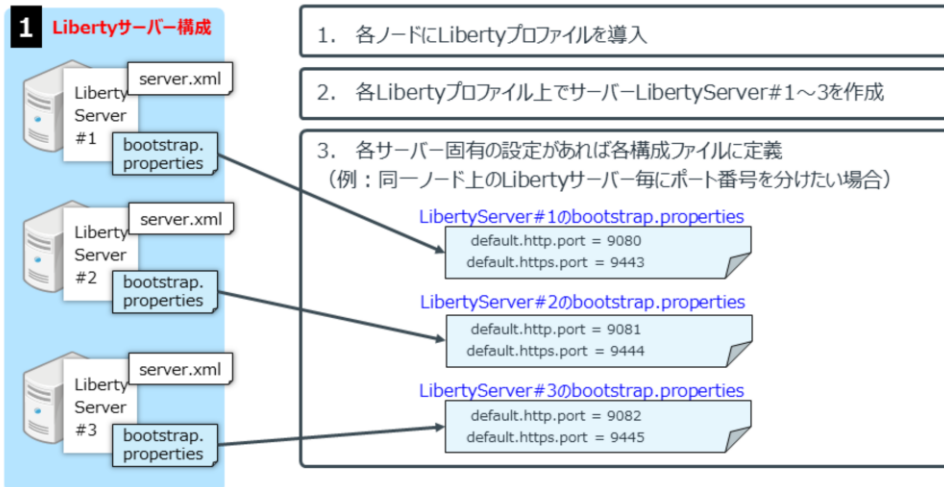
(2) 構成変更時の管理対象を最小化するために、Libertyサーバー共通となる構成情報とアプリケーションを共有させるように構成します。

(3) 各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成します。

(4) セッション情報を取り扱うアプリケーションの場合は、Libertyサーバー同士でセッション情報を共有するように構成します。

これらの各ステップに関する詳細な手順は次ページ以降でご説明します。

シンプル・クラスター構成方法 (1) Libertyサーバー構成



65

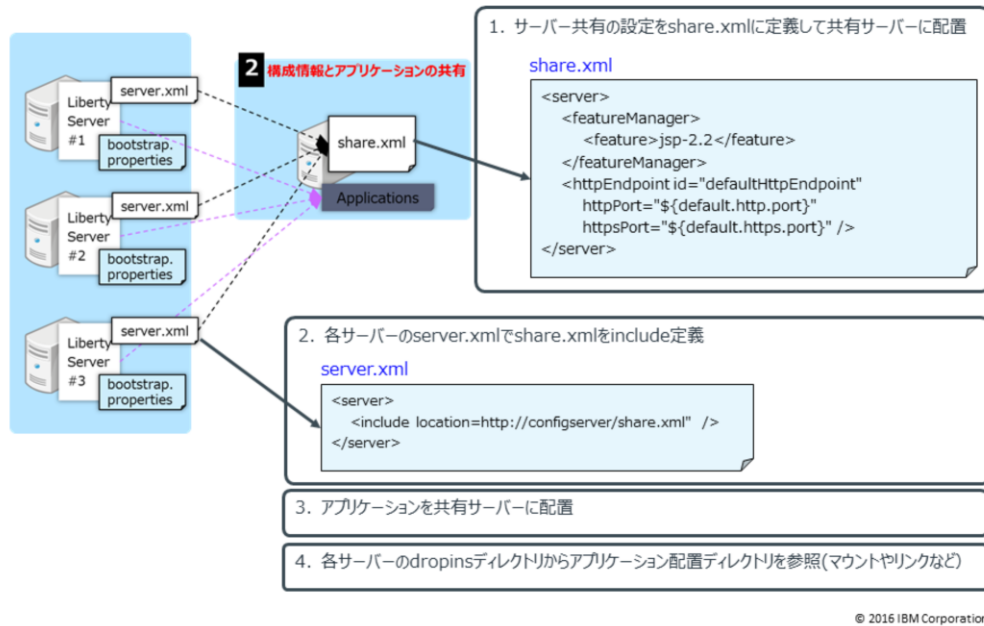
© 2016 IBM Corporation

こちらは、Libertyサーバーを稼働させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、サーバー固有の設定をする手順の例です。

1. シンプル・クラスターを構成する全てのノード上にLibertyプロファイルを導入します。
2. 各ノードのLibertyプロファイル上で、LibertyサーバーとしてLibertyServer#1～3を作成します。
3. シンプル・クラスター内のLibertyサーバーで共通的な設定は別の構成ファイルに定義するため、ここでは固有の設定があれば各Libertyサーバーの構成ファイルに定義します。ここでは、例えば同一ノード上に3つのLibertyサーバーを作成している場合、HTTPトランスポートおよびHTTPSトランスポート番号を分ける必要があるため、各Libertyサーバーのbootstrap.propertiesファイルにポート番号を定義している例を挙げています。

bootstrap.propertiesファイルは必須ではないため、作成しない限り存在しません。このファイルはサーバー・ディレクトリーに作成する必要があり、サーバー・ディレクトリーには、構成ルート・ファイル server.xml も含まれています。デフォルトでは、サーバー・ディレクトリーは usr/servers/server_name で、サーバー・ディレクトリーは変更できます。

シンプル・クラスター構成方法 (2) 構成とアプリケーションの共有



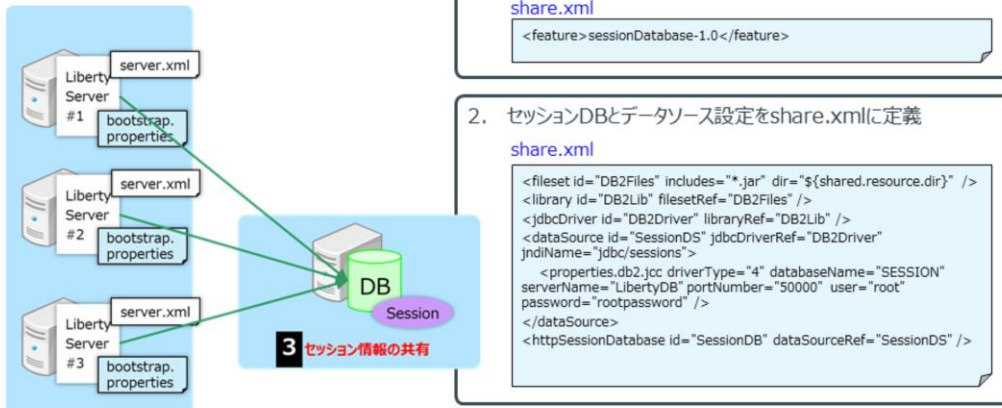
66

© 2016 IBM Corporation

こちらは、構成変更時の管理対象を最小化するために、Libertyサーバー共通となる構成情報とアプリケーションを共有させるように構成する手順の例です。

1. シンプル・クラスター内のLibertyサーバーで共通となる設定をshare.xmlなどserver.xmlとは別の構成ファイルに定義し、ファイルサーバーやWebサーバーなど共有サーバーに配置します。
2. 各Libertyサーバーのserver.xml内にshare.xmlをincludeするように定義します。
3. アプリケーションも共有サーバーに配置します。
4. 各Libertyサーバーのdropinsディレクトリからアプリケーションを配置したディレクトリを参照させるようにOSのマウント等の機能で構成します。

シンプル・クラスター構成方法 (3) セッション情報の共有



67

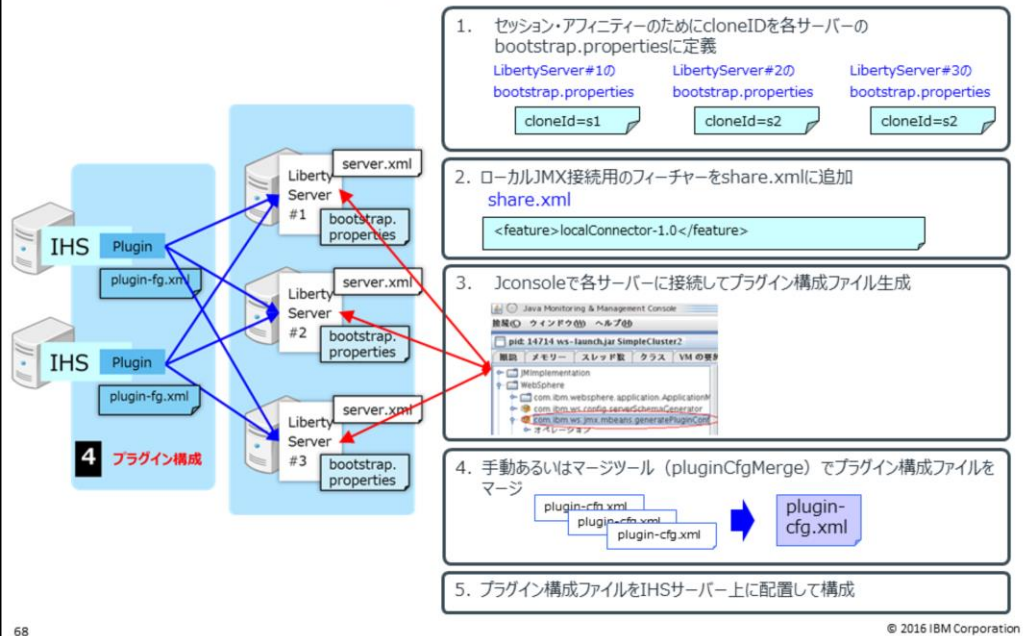
© 2016 IBM Corporation

こちらは、セッション情報を取り扱うアプリケーションの場合は、Libertyサーバー同士でセッション情報を共有するように構成する手順の例です。

1. セッション情報をDBに保管するセッション・パーシスタンスを有効にするために、share.xmlファイルにフィーチャー「sessionDatabase-1.0」を追加します。
2. share.xmlファイルに、セッションDBおよびセッションDBに接続するためのデータソースを定義します。

Libertyプロファイルでは、セッション・パーシスタンスのセッション保管先として、DBの他にもWebSphere eXtreme Scale やIBM WebSphere DataPower Appliance XC10 V2 caching appliance が選択できます。

シンプル・クラスター構成方法 (4) プラグイン構成



こちらは、各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成する手順の例です。

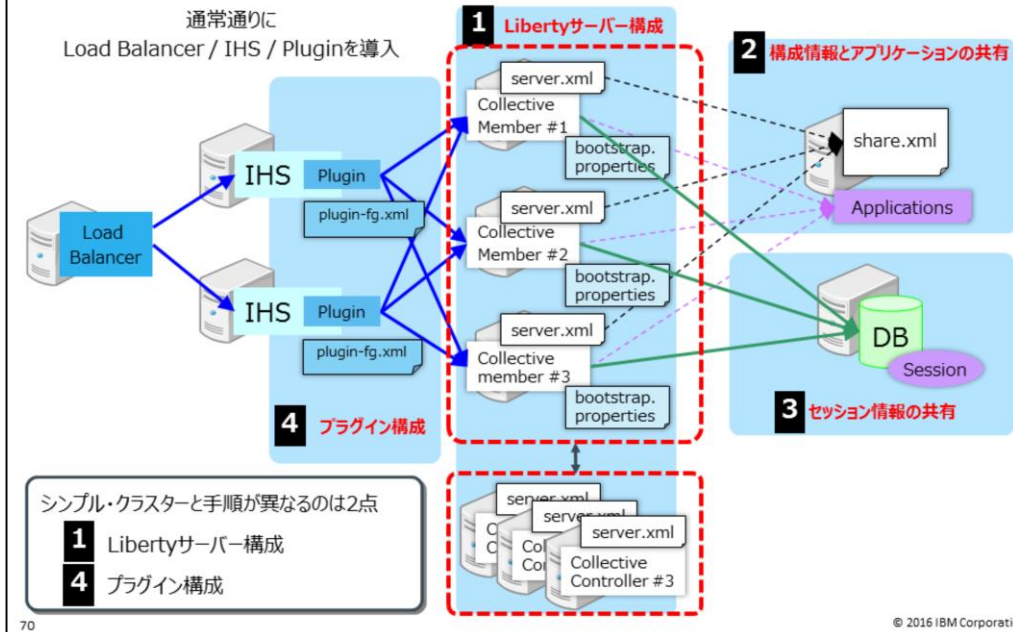
1. セッション・アフィニティーの機能を持たせるために、各Libertyサーバーのbootstrap.propertiesファイルにcloneIDを定義します。
2. プラグイン構成ファイルを生成するために各LibertyサーバーにJMX接続を行う必要があるため、share.xmlにローカルJMX接続用のフィーチャー「localConnector-1.0」を追加します。
3. Jconsoleで各Libertyサーバーに接続し、プラグイン構成ファイルを生成するオペレーションを実行します。
4. 各Libertyサーバーで生成されたプラグイン構成ファイルを手動あるいはフルプロファイルで提供されているマージツールpluginCfgMergeを使用して1つにマージします。
5. マージされたプラグイン構成ファイルをIHSプラグインが読み込めるようにIHSサーバー上に配置します。

<参考>

<http://www-01.ibm.com/support/docview.wss?uid=swg21674883>

3. 高可用性構成

「高可用性構成」 構成の流れ



LibertyプロファイルでLibertyクラスターを構成する場合の構成方法についてご説明します。

まず、フルプロファイルと同様、通常通りにLoad Balancer/IHS/PluginといったLibertyサーバー前段のコンポーネントを導入します。

以降の大まかな作業の流れは以下の通りです。

(1) Libertyサーバーを稼働させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、Liberty Collective を構成してサーバー固有の設定をします。

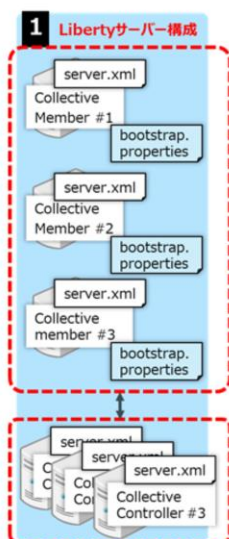
(2) 構成変更時の管理対象を最小化するために、Libertyサーバー共通となる構成情報とアプリケーションを共有させるように構成します。

(3) セッション情報を取り扱うアプリケーションの場合は、Libertyサーバー同士でセッション情報を共有するように構成します。

(4) 各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成します。

これらのステップのうち、シンプル・クラスターの構成手順と異なるのは(1)と(4)のみですので、それらに関する詳細な手順を次ページ以降でご説明します。

高可用性構成方法 (1) Libertyサーバー構成



1. 各ノードにLibertyプロファイルを導入
2. LibertyサーバーとしてCollectiveController #1~3を作成
3. CollectiveController #1のCollective作成コマンドを実行して起動
4. CollectiveController #2のReplicateコマンドを実行して起動
5. CollectiveController #2のaddReplicaコマンドを実行
6. CollectiveController #3のReplicateコマンドを実行して起動
7. CollectiveController #3のaddReplicaコマンドを実行
8. LibertyサーバーとしてCollectiveMember #1~3を作成
9. CollectiveMember #1のJoinコマンドを実行して起動
10. クラスター・メンバーのフィーチャーをshare.xmlに追加
 share.xml `<feature>clusterMember-1.0</feature>`
11. 所属するクラスターのグループ名をshare.xmlに追加
 share.xml `<clusterMember name="LibertyCluster"/>`

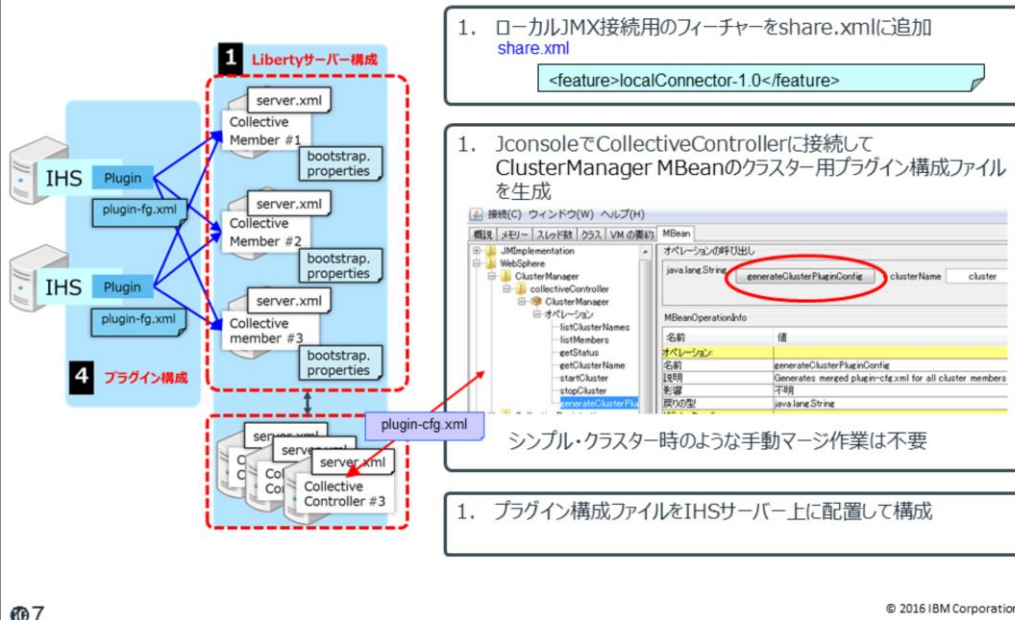
71

© 2016 IBM Corporation

こちらは、Libertyサーバーを稼働させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、Liberty Collective を構成してサーバー固有の設定をする手順の例です。

1. Liberty Collective を構成する全てのノード上にLibertyプロファイルを導入します。
2. Collective Controller となるLibertyサーバーとしてCollectiveController #1~3を作成します。
- 3~7. Libertyプロファイルで提供されているCollective Controller を構成するためのCollective、Replicate、addReplicaコマンドを使用してCollective Controller とReplica Set を構成します。
8. Collective Member となるLibertyサーバーとしてCollectiveMember #1~3を作成します。
9. Libertyプロファイルで提供されているCollective Member を構成するためのJoinコマンドを使用してCollective Member を構成します。
10. 共通設定を共有するためのshare.xmlファイルに「clusterMember-1.0」フィーチャーを追加します。
11. 更にshare.xmlにCollective Member が所属するクラスターのグループ名を定義します。

高可用性構成方法 (4)プラグイン構成



こちらは、各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成する手順の例です。

1. プラグイン構成ファイルを生成するために各LibertyサーバーにJMX接続を行う必要があるため、`share.xml`にローカルJMX接続用のフィーチャー「`localConnector-1.0`」を追加します。

2. JconsoleでいずれかのCollective Controller に接続し、ClusterManager Mbeanのプラグイン構成ファイルを生成するオペレーションを実行します。ここでは、既にクラスターのメンバー分の定義情報を含んだマージ版のプラグイン構成ファイルが生成されますので、手動でのマージ作業は不要です。

3. 生成されたプラグイン構成ファイルをIHSプラグインが読み込めるようにIHSサーバー上に配置します。

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出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もありません。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。



*WebSphere
Application Server
Liberty*