

JEUS アプリケーション&デプロイメントガイド

JEUS v8.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

Restricted Rights Legend

All TmaxSoft Software (JEUS®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(JEUS®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

Trademarks

JEUS® is registered trademark of TmaxSoft Co., Ltd.

JEUS®は、TmaxSoft Co., Ltd.の登録商標です。

Java and Solaris are registered trademarks of Oracle Corporation and its subsidiaries and affiliates.

Java、Solarisは、Oracle Corporation及びその子会社、関連会社の登録商標です。

Microsoft, Windows, and Windows NT are registered trademarks or trademarks of Microsoft Corporation.

Microsoft、Windows、Windows NTは、Microsoft Corporationの登録商標または商標です。

HP-UX is a registered trademark of Hewlett Packard Enterprise Company.

HP-UXは、Hewlett Packard Enterprise Companyの登録商標です。

AIX is a registered trademark of International Business Machines Corporation.

AIXは、International Business Machines Corporationの登録商標です。

UNIX is a registered trademark of X/Open Company, Ltd.

UNIXは、X/Open Company, Ltd.の登録商標です。

Linux is a registered trademark of Linus Torvalds.

Linuxは、Linus Torvaldsの登録商標です。

Other products and company names are trademarks or registered trademarks of their respective owners.

その他、記載されている会社名、製品名などは、各社の商号、商標または登録商標です。

The names of companies, systems, and products mentioned in this manual may not necessarily be indicated with a trademark symbol (TM, ®).

本マニュアルに記載されている会社名、システム名、製品名などには必ずしも商標表示(TM、®)を付記しておりません。

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

Detailed Information related to the license can be found in the following directory : \${INSTALL_PATH}/lib/licenses

この製品の一部ファイルまたはモジュールは、APACHE2.0、CDDL1.0、EDL1.0、OPEN SYMPHONY SOFTWARE1.1、TRILEAD-SSH2、Bouncy Castle、BSD、MIT、SIL OPEN FONT1.1のライセンスに準拠します。

文書情報

文書名: JEUS アプリケーション&デプロイメントガイド

発行日: 2016年10月14日

ソフトウェアバージョン: JEUS v8.0

ガイドバージョン: v2.1.1

目次

このガイドについて	xiii
第1章 ドメイン環境でのアプリケーション管理	1
1.1. アプリケーション管理	1
1.1.1. 2フェーズ・デプロイメント	2
1.1.2. アプリケーションのID	5
1.1.3. アプリケーションの状態	5
1.1.4. アプリケーション管理のディレクトリー構造	8
1.1.5. アプリケーションのサービス対象	11
1.2. MSでのデプロイ	13
1.2.1. ランタイム・デプロイ	13
1.2.2. ブートタイム・デプロイ	14
1.2.3. アプリケーションの同期化	17
1.3. 自動再デプロイ	17
1.4. ディレクトリー・モードのデプロイ	18
1.5. アプリケーション・リポジトリー	19
1.5.1. アプリケーション・リポジトリーの追加/削除/照会	19
1.5.2. アプリケーション・リポジトリーにあるアプリケーションのデプロイ	22
1.6. パスを指定してデプロイ	22
1.7. ステージング・モード・デプロイ	22
1.8. アプリケーションのアンデプロイ	23
第2章 グレースフル・アンデプロイとグレースフル再デプロイ	25
2.1. グレースフル・アンデプロイ	25
2.2. グレースフル再デプロイ	27
2.2.1. グレースフル再デプロイを使用するための前提条件	27
2.2.2. グレースフル再デプロイの必須考慮事項	28
2.2.3. グレースフル再デプロイの使用方法	29
2.2.4. Webアプリケーションのグレースフル再デプロイ	31
2.2.5. EJBアプリケーションのグレースフル再デプロイ	32
2.2.6. EARアプリケーションのグレースフル再デプロイ	33
第3章 アプリケーション	35
3.1. モジュールとアプリケーション	35
3.1.1. モジュール	36
3.1.2. アプリケーション	37
3.2. デプロイメント記述子(DD)	38
3.3. アプリケーションでのライブラリーの使用	40
3.3.1. lib/applicationディレクトリー	40
3.3.2. 共有ライブラリー	42
3.3.3. ライブラリーのデプロイメント	46
第4章 アプリケーションの作成およびデプロイ	53

4.1.	アプリケーションの作成	53
4.2.	デプロイ	54
4.3.	アプリケーションの制御およびモニタリング	56
4.3.1.	アプリケーションをドメインにインストール	57
4.3.2.	アプリケーションをドメインから削除	57
4.3.3.	アプリケーションのデプロイ	58
4.3.4.	アプリケーションの再デプロイ	65
4.3.5.	アプリケーションのアンデプロイ	65
4.3.6.	アプリケーションの開始	66
4.3.7.	アプリケーションの停止	66
4.3.8.	サービス中のアプリケーションにサーバーを追加	66
4.3.9.	サービス中のアプリケーションからサービス中のサーバーを削除	67
4.3.10.	アプリケーション情報の確認	68
4.4.	ステージング・モード・デプロイ	76
4.5.	デプロイメント・プランを利用したデプロイ	78
4.5.1.	デプロイメント・プランの設定および動作方式	78
4.5.2.	デプロイメント・プランのインストール	85
4.5.3.	インストールしたデプロイメント・プランの確認	85
4.5.4.	デプロイメント・プランを適用したデプロイ	87
4.5.5.	アプリケーションに適用されたデプロイメント・プランの確認	87
4.5.6.	デプロイメント・プランのアンインストール	88
4.5.7.	デプロイメント・プランと再デプロイ	89
用語集		91
索引		93

目次

[図 1.1]	WEBモジュールがDISTRIBUTED状態で要求を行った場合に発生するエラー	4
[図 1.2]	ドメインでのアプリケーションのライフサイクル	6
[図 1.3]	サーバーでのアプリケーションのライフサイクル	7
[図 1.4]	ドメインのINSTALL_HOMEディレクトリー	8
[図 1.5]	ドメインに追加したアプリケーション・リポジトリ・ディレクトリー	8
[図 1.6]	サーバーにデプロイされたEARアプリケーションのイメージ・ディレクトリー	9
[図 1.7]	サーバーにデプロイされたEARアプリケーションのgenディレクトリー	10
[図 2.1]	EARアプリケーションのグレースフル・アンデプロイ	26
[図 3.1]	Java EEモジュールおよびアプリケーションの構成	35
[図 3.2]	.ear archiveの構成	37
[図 3.3]	ライブラリーの管理	48
[図 3.4]	依存ライブラリーの設定	50
[図 4.1]	WebAdminの[Deployed Application]画面	56
[図 4.2]	WebAdminでのアプリケーションの構成情報の確認	69
[図 4.3]	WebAdminでのアプリケーションの構成情報の確認	69
[図 4.4]	WebAdminでのEJBモジュール情報の確認	70
[図 4.5]	WebAdminでのEJBモジュールのBean情報の確認	70
[図 4.6]	WebAdminでのWEBモジュールのサーブレット情報の確認	71
[図 4.7]	WebAdminでのWEBモジュールのサーブレット要求スレッド情報の確認	72
[図 4.8]	WebAdmin - ステージング・モード・デプロイ	77

表目次

[表 3.1]	モジュール別のDD	38
[表 4.1]	アプリケーション関連のコマンド	59

例目次

[例 1.1]	EJBモジュールがDISTRIBUTED状態で要求を行った場合に発生するクライアントの例外エラー	3
[例 1.2]	指定された回数の再実行がすべて失敗した場合のログ・メッセージ	12
[例 1.3]	クラスター・メンバーのサーバーに指定されている場合、MSの起動時に発生するログ・メッセージ	12
[例 3.1]	<<共有ライブラリーの登録 : libraries.xml>>	43
[例 3.2]	<<共有ライブラリーの<files>タグ>>	44
[例 4.1]	コンソール・ツールでドメインにインストールしたアプリケーションのデプロイ	60
[例 4.2]	コンソール・ツールでアプリケーション・リポジトリにあるアプリケーションをデプロイ	63
[例 4.3]	コンソール・ツールでパスを指定してデプロイ	64
[例 4.4]	コンソール・ツールでアプリケーション情報の確認	72
[例 4.5]	コンソール・ツールを使用したステージング・モード・デプロイ	78

このガイドについて

対象読者

本書は、JEUS[®](以下、JEUS)製品群の1つのJEUSサーバーにJava EEアプリケーションまたはJava EEモジュールをデプロイおよび管理する管理者やアプリケーション開発者を対象としています。

前提知識

JEUSの基本的な使用方法と製品を理解するには、以下のガイドについてあらかじめ熟知することをお勧めします。

- JEUS 紹介ガイド
- JEUS インストール & スタートガイド
- JEUS サーバガイド

本書のすべてのサンプルと環境構成は、UNIXスタイルに準拠します。Microsoft Windows[™](以下、Windows)など他の環境で作業を行う場合は、次のような事項を考慮してください。たとえば、Windowsプラットフォームでは、ディレクトリー区切り子をUNIXスタイルのスラッシュ(/)からWindowsスタイルのバックスラッシュ(\)に変えて使用してください。また、環境変数もWindowsスタイル(%%)に変更して使用してください。本書で触れているJEUS_HOMEは、JEUSがインストールされているディレクトリーです。

制限事項

本書の内容は、Java標準に準拠して作成されていますが、本書で触れているJava EEやJava仕様については詳しく取り上げていません。関連内容についてはJava関連ドキュメントを参照してください。

本書の構成

本書は、計4章で構成されています。

- 「[第1章 ドメイン環境でのアプリケーション管理](#)」

ドメイン環境でのアプリケーション管理について説明します。

- 「[第2章 グレースフル・アンデプロイとグレースフル再デプロイ](#)」

グレースフル・アンデプロイとグレースフル再デプロイについて説明します。

- 「[第3章 アプリケーション](#)」

モジュールとアプリケーション、共有ライブラリーについて説明します。

- 「[第4章 アプリケーションの作成およびデプロイ](#)」

Java EEアプリケーション・ファイル(EAR)を作成し、これをJEUSにデプロイする方法について説明します。

表記上の規則

表記	意味
<<AaBbCc123>>	プログラム・ソースコードのファイル名
<Ctrl>+C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
ハイパーリンク	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
<div>参考</div>	参照/注意事項
<div>注</div>	注意事項
[図 1.1]	図の名前
[表 1.1]	表の名前
AaBbCc123	Javaコード、XMLドキュメント
[<i>command argument</i>]	オプション・パラメータ
< xyz >	「<」と「>」の間の内容は実際に使用される特定の名前または値で置き換えられる
	構文の中の相互に排他的な選択項目の選択肢を示す 例) A B: AとBのいずれかを選択
...	パラメータ、値、または他の情報が繰り返される
\${ }	環境変数

システム要件

	要求事項
プラットフォーム	Solaris 9, 10, 11
	HP-UX 11.x, 11i, 11iV2
	IBM AIX 5L, 6L, AIX 7L
	MS Windows 2008, 2012, Vista, 7, 8
ハードウェア	最小2GB以上、推奨20GBのハードディスク容量
	推奨1GB以上のメモリー容量
JDK	JDK 7, JDK 8

関連文書

ガイド	説明
JEUS 紹介ガイド	JEUSサーバーについて全般的に紹介し、JEUSのアーキテクチャーを含む各構成要素について記述しています
JEUS インストール&スタートガイド	JEUSについて紹介し、JEUSのインストールおよび開始方法について記述しています
JEUS サーバガイド	JEUSシステムおよびサーバーの概要とシステムの管理方法について記述しています
JEUS セキュリティガイド	JEUSでのセキュリティ・システムの設定と運用方法およびセキュリティ関連プログラミングについて記述しています
JEUS アプリケーションクライアントガイド	Java EEクライアントとJEUS間の相互運用について記述しています
JEUS Webエンジンガイド	JEUS Webエンジンの管理方法、Java EE WARアーカイブとサーブレット/JSPの管理およびデプロイ方法について記述しています
JEUS EJBガイド	JEUS EJBエンジンおよびEJBモジュールのデプロイについて記述しています
JEUS リファレンスガイド	JEUSを使用するために必要な詳細設定とJEUSの使用方法について記述しています
JEUS WebAdminガイド	JEUSのWeb管理ツールであるWebAdminを利用したJEUSの設定および制御、モニタリング、クラスタリング、リソースの設定および管理について記述しています

お問合せ先

Korea

TmaxSoft Co., Ltd.
45, Jeongjail-ro, Bundang-gu,
Seongnam-si, Gyeonggi-do, 13613
South Korea
Tel: +82-31-8018-1000
Fax: +82-31-8018-1115
Email: info@tmax.co.kr
Web (Korean): <http://www.tmaxsoft.com>
TechNet: <http://technet.tmaxsoft.com>

USA

TmaxSoft Inc.
101 North Wacker Drive, Suite 2014,
Chicago, IL 60606
U.S.A
Tel: +1-312-525-8330
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/us_en/home

Japan

TmaxSoft Japan Co., Ltd.
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073
Japan
Tel: +81-3-5765-2550
Fax: +81-3-5765-2567
Email: info@tmaxsoft.co.jp
Web (Japanese): <http://www.tmaxsoft.co.jp>

China

Beijing TmaxSoft System Software Co., Ltd.
Room103, No.2 Huizhong Building, Seven Street Shangdi,
Haidian District, Beijing, 100085
P.R.China
Tel: +86-10-6298-8827
Email: info@tmaxsoft.com.cn
Web (Chinese): http://www.tmaxsoft.com/cn_en/home_cn_en

Brazil

Tmax Brasil Sistemas e Serviços Ltda.
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel
Alphaville Barueri, Sao Paulo, 06472-001
Brazil
Tel: +55-11-4191-3100
Fax: +55(11) 4191-3705 (extension#112)
Email: info.bra@tmaxsoft.com
Web (Portuguese): http://www.tmaxsoft.com/br_en/home_br_en

Russia

Tmax Rus L.L.C.
Leninsky prospekt, 113/1 (Park Place Moscow),
Office 318e, Moscow, 117198
Russia
Tel: +7(495)970-01-35
Email: info.rus@tmaxsoft.com
Web (Russian): http://www.tmaxsoft.com/ru_ru/home_ru_ru

Singapore

Tmax Singapore Pte. Ltd.
430 Lorong 6, Toa Payoh #10-02,
OrangeTee Building, 319402
Singapore
Tel: +65-6259-7223
Fax: +65-6258-7112
Email: info.sg@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/sg_en/home_sg_en

United Kingdom

TmaxSoft UK Ltd.
215 Knyvett House, Watermans Business Park,
The Causeway, Staines TW18 3BAB
United Kingdom
Tel: +44-1784-895005
Email: info.uk@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/gb_en/home_gb_en

Canada

TmaxSoft Canada, Inc.
2425 Matheson Blvd East, 8th floor,
Unit 824 Mississauga, ON, L4W 5K4
Canada
Tel: +1-905-361-2888
Email: info.canada@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/ca_en/home_ca_en

Australia

TmaxSoft Proprietary Limited
L32, 101 Miller Street, North Sydney 2060
Australia
Tel: +91-9845-330-704
Email: info.aus@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/au_en/home_au_en

India

TmaxSoft Technologies Private Limited
Sobha Alexander Plaza, 3rd Floor,
16/2 Commissariat Road, Bangalore-560025
India
Tel: +91-9845-330-704
Email: info.india@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/in_en/home_in_en

Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office
Windowist Tower. Eski Buyukdere Cad. No:26,
Maslak 34467 Istanbul
Turkey
Tel: +90-544-553-6045
Email: cslee@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/tr_en/home_tr_en

第1章 ドメイン環境でのアプリケーション管理

本章では、ドメイン環境でのアプリケーション管理方法およびデプロイ手順について説明します。また、Managed Server(以下、MS)でのデプロイ手順について説明します。

1.1. アプリケーション管理

Domain Administration Server(以下、DAS)は、ドメインを管理するサーバーです。アプリケーションはドメイン単位で管理される必要があるため、DASではドメインに存在するすべてのアプリケーションを管理します。

アプリケーションに関連するデプロイおよび照会はDASを介してのみ実行可能です。DASの起動時にアプリケーションを管理するサービスが開始され、サーバーやクラスターにデプロイ可能となります。デプロイだけでなく、アプリケーションに関連するすべての命令がDASを通じて行われます。DASに障害が発生してMSがINDEPENDENT状態になった場合は、アプリケーションへのデプロイは実行できません。ただし、コンソール・ツール(jeusadmin)を使用し、該当のMSに接続してアプリケーションの情報を確認することは可能です。

注

MSに直接接続してアプリケーションの情報を照会できるのは、MSがINDEPENDENT状態の場合のみです。MSがINDEPENDENT状態であるということは、DASに接続できないか、DASでMSを管理できない状況であるため、MSに直接接続してアプリケーションの情報を確認します。

アプリケーションをサービスできるようにデプロイするためには、まずアプリケーションをドメインにインストールします。アプリケーションのインストールは、アプリケーション・ファイルをDASにインストールする作業であり、デプロイは、アプリケーションをサービスできるようにする作業です。

● インストール

アプリケーション・ファイルをDASにアップロードします。DASにインストールされたアプリケーションのみデプロイ可能なので、アプリケーションをDASにインストールします。

- WebAdminあるいはコンソール・ツールを使用して、アプリケーション・ファイルをドメインのAPPLICATION_INSTALL_HOMEに格納します。

APPLICATION_INSTALL_HOMEは、ドメインにインストールされたアプリケーションを管理するリポジトリです。このリポジトリを変更したり、手動でアクセスしたりできません。インストールされたアプリケーションの削除や新しいアプリケーションの追加は動的に適用されません。DASを再起動した場合は反映されますが、この方法は推奨しません。

- ユーザーが指定したディレクトリにアプリケーションを格納します。

ユーザーが指定したディレクトリーは、アプリケーションが格納される特定のディレクトリーを意味します。ユーザーはこのディレクトリーをアプリケーション・リポジトリーに追加し、ここにアプリケーション・ファイルを格納することができます。WebAdminあるいはコンソール・ツールの**add-application-repository**と**remove-application-repository**コマンドを使用して、ドメインにアプリケーション・リポジトリーを追加または削除することが可能です。

● デプロイ

アプリケーションがサービスできるように、アプリケーションをサーバーにデプロイし、サービスの準備作業を行います。

WebAdminあるいはコンソール・ツールを使用して、DASにアプリケーションのデプロイを命令すると、DASでは対象となるサーバーやクラスターにデプロイを行います。この際、DASとサーバーでは2フェーズ・デプロイメントを行います。2フェーズ・デプロイメントについての詳細は、「[1.1.1.2フェーズ・デプロイメント](#)」を参照してください。

1.1.1. 2フェーズ・デプロイメント

アプリケーション・ファイルを各サーバーにデプロイし、アプリケーション・サービスを実行するための事前作業を行います。この作業が終了すると、アプリケーションをサービスできる状態になります。

DASではデプロイを2フェーズに分けて処理します。すべてのデプロイ命令は、DASを介して対象となるサーバーに送られます。ユーザーが要求したデプロイは、対象として指定したすべてのサーバーに対するデプロイの成功が保証される必要があります。したがって、アプリケーションを配布と開始の2フェーズに分けて実行します。第1フェーズで失敗したデプロイは失敗したと見なされ、すべてのサーバーからアンデプロイされます。

- 第1フェーズの成功は、アプリケーションの配布と検証が成功したという意味なので、潜在的なデプロイの成功といえます。ただし、アプリケーションをサービスできる状態ではありません。
- アプリケーションをサービスできるのは、第2フェーズまで成功してからになります。第2フェーズでは、失敗したサーバーがあっても、再実行して該当のサーバーでアプリケーションをサービスできる状態にします。

上記の2フェーズは、デプロイではなく、配布と開始に分けてそれぞれ実行することもできます。

第1フェーズ(配布)

対象となるサーバーやクラスターにアプリケーションを配布し、検証します。JEUSではこれをアプリケーションの**配布**といいます。このフェーズでは、アプリケーション・ファイルを各サーバーにデプロイし、アプリケーションをサービスできるよう、事前準備と検証を行います。

配布がすべて成功すると、サービスを行うためのすべての作業は終了した状態になります。EJBモジュールの場合はサービス・ポートが開き、Webモジュールの場合はWebリスナーとWebモジュールのコンテキストが接続されます。ただし、アプリケーションはサービスできない**DISTRIBUTED**状態です。

アプリケーションがDISTRIBUTED状態のとき、アプリケーションにサービスを要求すると、EJBモジュールの場合は「Beanが存在しません」という例外エラーが発生し、Webモジュールの場合は「503 Service Unavailable」エラーが発生します。

[例 1.1] EJBモジュールがDISTRIBUTED状態で要求を行った場合に発生するクライアントの例外エラー

```
[2016.08.10 15:28:35][1] [t-1] [JNDI.Context-0073] exception occurred during JNDI
operation
<<__Exception__>>
javax.naming.NamingException: javax.ejb.EJBException: java.rmi.RemoteException:
EJB object is not read at
jeus.ejb.client.BusinessObjectFactory.getObjectInstance(BusinessObjectFactory.
java:89)
    at javax.naming.spi.NamingManager.getObjectInstance(NamingManager.java:304)
    at jeus.jndi.JNSContext.lookupInternal(JNSContext.java:594)
    at jeus.jndi.JNSContext.lookup(JNSContext.java:549)
    at jeus.jndi.JNSContext.lookup(JNSContext.java:538)
    at jeus.jndi.JEUSFailoverContext.lookup(JEUSFailoverContext.java:314)
    at javax.naming.InitialContext.lookup(InitialContext.java:392)
    at test.HelloTest.testHelloBean(HelloTest.java:29)
    .....
Caused by: java.rmi.RemoteException: EJB object is not ready at
jeus.ejb.container.RemoteInvocationManagerImpl.beforeInvoke(RemoteInvocationManagerImpl.
java:72
    at jeus.ejb.baseimpl.RemoteInvokerServer.preInvoke(RemoteInvokerServer.java:118)

    at jeus.ejb.baseimpl.RemoteInvokerServer.invoke(RemoteInvokerServer.java:97)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.
java:25)
    at java.lang.reflect.Method.invoke(Method.java:597)
    at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:305)
    at sun.rmi.transport.Transport$1.run(Transport.java:159)
    at java.security.AccessController.doPrivileged(Native Method)
    at sun.rmi.transport.Transport.serviceCall(Transport.java:155)
    at sun.rmi.transport.tcp.TCPTransport.handleMessages(TCPTransport.java:535)
    at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0(TCPTransport.
java:790)
    at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(TCPTransport.java:649)

    at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.
```

```
java:886)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)

  at java.lang.Thread.run(Thread.java:662)
```

[図 1.1] WEBモジュールがDISTRIBUTED状態で要求を行った場合に発生するエラー

HTTP Status : 503 - 503 Service Unavailable

Description	Worker (http1-2): An error occurred while processing the request.

ユーザーがデプロイ対象として指定したすべてのサーバーとクラスターへの配布が成功する必要があります。配布に失敗したサーバーが1つでもあると、ユーザーが実行したデプロイは失敗し、成功したサーバーではアンデプロイが実行されると同時に、このフェーズで行った作業をすべてロールバックします。デプロイ対象として指定したサーバーの一部が失敗またはシャットダウンした場合もデプロイは失敗となります。この場合、該当のサーバーをデプロイ対象から除外し、デプロイを再実行する必要があります。

デプロイ対象がクラスターの場合、クラスター・メンバーのうち一部のサーバーが失敗またはシャットダウンした状態であっても、配布が成功する場合があります。これらのサーバーを除き、他のすべてのサーバーで配布が成功したら、このアプリケーションの配布は成功となります。

第2フェーズ(開始)

対象となるサーバーやクラスターにアプリケーションが既に配布されている場合は、アプリケーションがサービスできるように開始させます。この段階では、アプリケーションをサービスできる状態にする簡単な操作を行います。JEUSではこれをアプリケーションの**開始**といいます。

開始は配布とは異なり、対象となるサーバーやクラスターのうちいずれかのサーバーで開始が成功したら、全体の成功と見なします。その理由は、配布の終了と同時にアプリケーションの検証が完了し、アプリケーションをサービスできるすべての準備が整った状態であるからです。

開始が失敗したということは、サーバーに一時的な不具合が発生している可能性があります。これは、サーバーが正常化すれば開始に成功し、アプリケーションをサービスできるという意味でもあります。したがって、開始に失敗したサーバーは、DASにて別のスレッドで5秒間隔で10回再試行します。

再試行も失敗した場合は、手動でサーバーを復旧する必要があります。サーバーが復旧してからアプリケーションの開始を再実行すると、アプリケーションの開始に失敗していたサーバーの開始が正常に実行されます。

1.1.2. アプリケーションのID

JEUS 8からは、ドメインでアプリケーションを管理するために、アプリケーションを識別できるIDをアプリケーションごとに発行して使用しています。アプリケーションのIDはドメインでアプリケーションを管理するために必要な名前です。また、アプリケーションを区別できる識別子でもあるため、ドメインで一意である必要があります。

アプリケーションのインストール時にIDを設定します。インストール時にアプリケーションのIDを設定しなかった場合は、アプリケーション・ファイル名でIDを作成して使用します。たとえば、examples.earのインストール時にIDを設定しなかった場合、アプリケーションのIDは「examples_ear」になります。IDはアルファベットまたは数字を使用することを推奨します。

アプリケーションのIDはデプロイやアンデプロイなど、アプリケーションの制御や照会時に必要です。ドメインに別のリポジトリを設定し、その場所にアプリケーション・ファイルを格納した場合は、アプリケーション・ファイル名をそのままIDとして使用します。

DASからアプリケーション・ファイルが送られると、ドメイン・ディレクトリーの下位に存在するAPPLICATION_INSTALL_HOMEにIDを使ってディレクトリーを作成し、その下位にアプリケーション・ファイルを格納します。

アプリケーションをインストールすると、以下のディレクトリーが作成されます。INSTALL_HOMEディレクトリーについての詳細は、[1.1.4節「ドメインのINSTALL_HOMEディレクトリー」](#)を参照してください。

```
INSTALL_HOME/<APPLICATION_ID>/<APPLICATION_FILE>
```

サーバーでは、アプリケーション・ファイルの受信や圧縮を解凍する際、DEPLOYEDディレクトリーの下位にAPPLICATION_IDを最上位ディレクトリーとして構成し、その下位にファイルを格納して処理します。

注

アプリケーション名はJava EE 6から標準DDに定義することができ、アプリケーション・サービスのために必ず必要な名前です。この値をアプリケーションIDと混同しないよう注意してください。

アプリケーション名はアプリケーションがサービスされているサーバーでのみ一意であればいいのですが、アプリケーションIDはドメインで一意である必要があります。アプリケーション名が設定されていない場合、拡張子を除いたアプリケーション・ファイル名がアプリケーション名になります。アプリケーション名の設定方法についての詳細は、『Java EE 7 Platform Specification』を参照してください。

1.1.3. アプリケーションの状態

ドメインではアプリケーションへの命令を管理するので、ドメインで示すアプリケーション状態は対象サーバー全体の状態となります。サーバーでは実際にアプリケーションのデプロイやサービスが行われるので、アプリケーションの状態を示します。

ドメインでのアプリケーションの状態

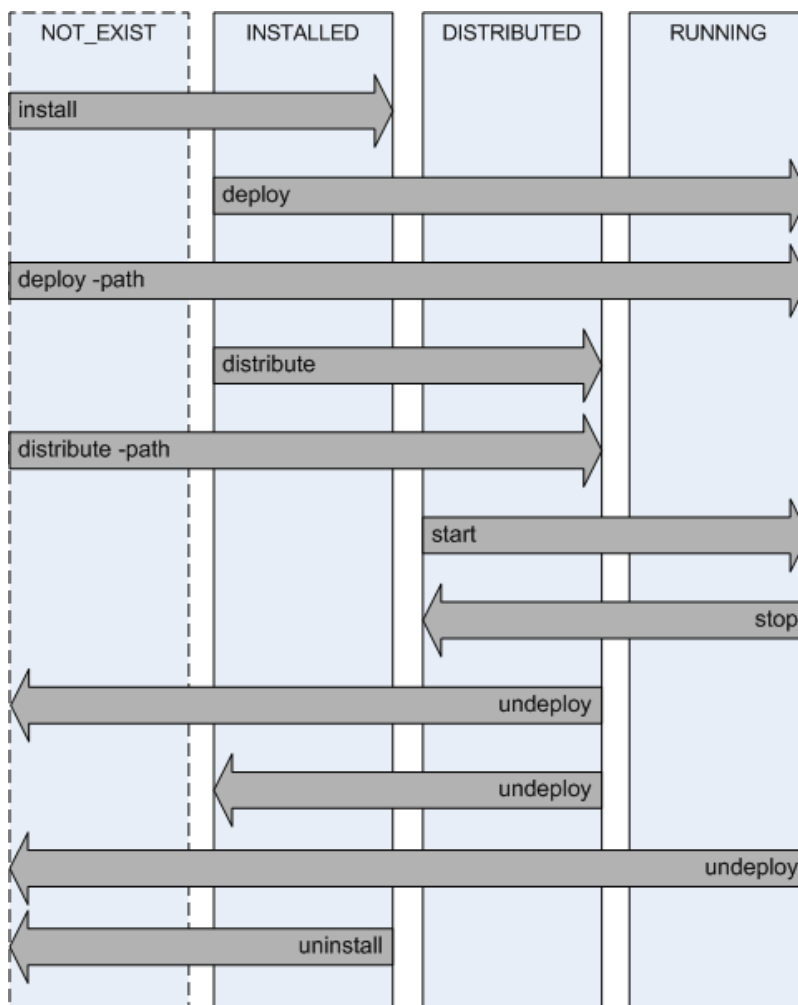
以下の図は、ドメインにおけるアプリケーションのライフサイクルと状態のプロセスを示しています。

INSTALLED → DISTRIBUTING → DISTRIBUTED → STARTING → RUNNING

以下は、各状態についての説明です。

状態	説明
INSTALLED	アプリケーション・ファイルがドメインにアップロードされた状態です。この状態でデプロイの対象を指定して、デプロイあるいは配布を実行できます
DISTRIBUTED	アプリケーションが正常に配布された状態です。対象サーバーがすべてSHUTDOWN状態であってもDISTRIBUTED状態と表示されます
RUNNING	アプリケーションがサービスされている状態です。アプリケーションにデプロイ対象として指定したサーバーのうち1つでもRUNNING状態でサービスされている場合、ドメインでの該当アプリケーションの状態はRUNNINGになります

【図 1.2】ドメインでのアプリケーションのライフサイクル



参考

ドメインでは上記の3つのほか、**DEPLOYED**という状態が存在します。

デプロイされた履歴があるアプリケーションの対象サーバーがすべてRUNNING状態ではない場合、ドメインではアプリケーションの状態をDEPLOYEDと表示します。対象サーバーが稼動していないということは、アプリケーションをサービスしているサーバーが存在しないという意味であるため、RUNNING状態ではなくDEPLOYED状態と表示します。

サーバーでのアプリケーションの状態

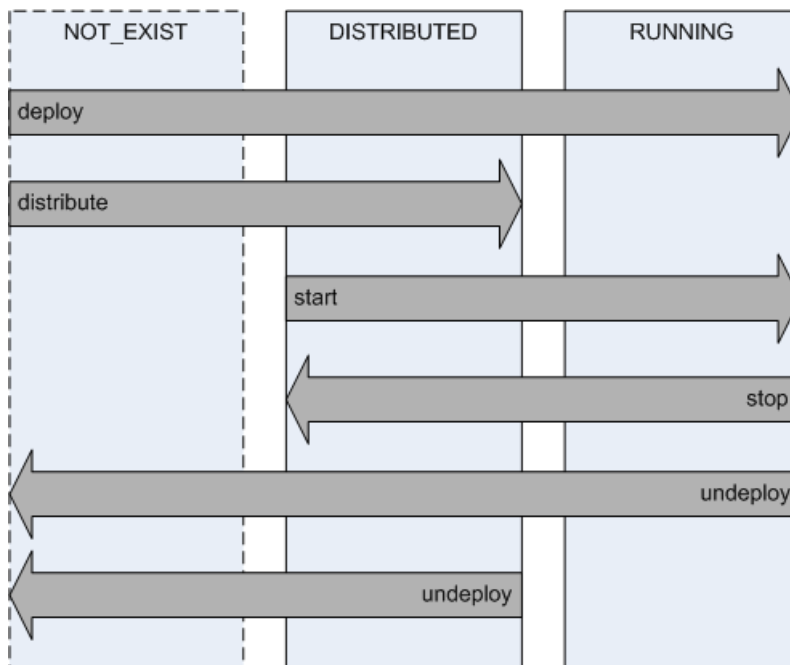
以下の図は、サーバーにおけるアプリケーションのライフサイクルと状態のプロセスを示しています。

DISTRIBUTING → DISTRIBUTED → STARTING → RUNNING

以下は、各状態についての説明です。

状態	説明
DISTRIBUTED	DASからの配布命令が成功した状態です。アプリケーションはサービス準備を終えているので、開始を実行してアプリケーションをサービス可能な状態にできます
RUNNING	該当のサーバーでアプリケーションがサービスされている状態です

[図 1.3] サーバーでのアプリケーションのライフサイクル



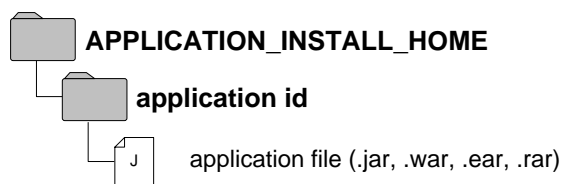
1.1.4. アプリケーション管理のディレクトリー構造

アプリケーションをデプロイする際に使用されるディレクトリーの構造は以下のとおりです。

ドメインのINSTALL_HOMEディレクトリー

アプリケーションがドメインにインストールされると、APPLICATION_INSTALL_HOMEに格納されます。

[図 1.4] ドメインのINSTALL_HOMEディレクトリー



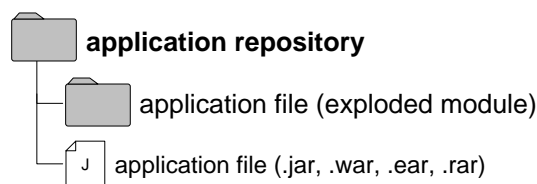
APPLICATION_INSTALL_HOME

DASが存在するマシンでのみ作成されるディレクトリーです。アプリケーションをドメインにインストールすると、当該ディレクトリーに格納されます。インストール時に設定したアプリケーションIDでディレクトリーを作成し、その下位にアプリケーション・ファイルを格納します。JEUSでは、このディレクトリーをAPPLICATION_INSTALL_HOMEと呼び、簡単に**INSTALL_HOME**ともいいます。APPLICATION_INSTALL_HOMEは、DOMAIN_HOME/.applicationsを意味します。

ドメインで管理するアプリケーション・リポジトリー

ドメインに追加したアプリケーション・リポジトリーは以下のように構成されます。

[図 1.5] ドメインに追加したアプリケーション・リポジトリー・ディレクトリー



アプリケーション・リポジトリーには、ディレクトリー形式のアプリケーションとアーカイブ形式のアプリケーションがすべて格納されます。アプリケーションのIDはユーザーが指定することはできず、常にファイル名を使用します。ユーザーはアプリケーションのリポジトリーを追加または削除することができます。その方法については、[「1.5.1. アプリケーション・リポジトリーの追加/削除/照会」](#)を参照してください。

デプロイ・イメージ・ディレクトリー

サーバーにアプリケーションをデプロイする際、DASから受信したアプリケーションを、以下のパスに格納します。

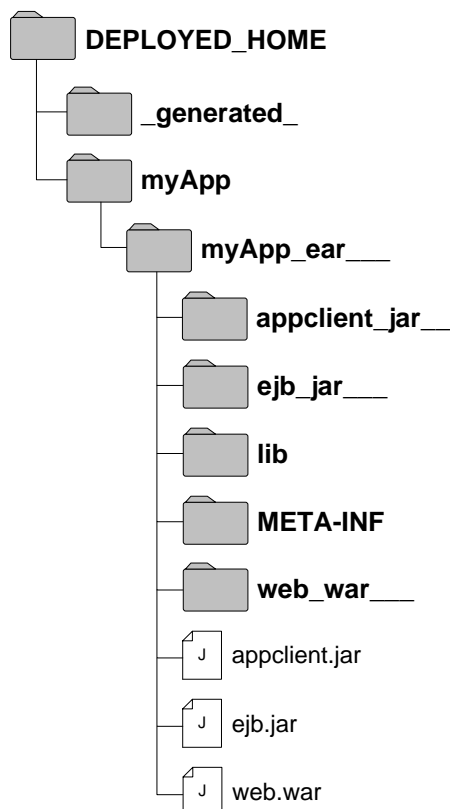
```
SERVER_HOME/.workspace/deployed
```

この場所を**DEPLOYED_HOME**といいます。DEPLOYED_HOMEにアプリケーションIDでディレクトリーを作成し、受信したアプリケーション・ファイルを下位に格納します。

また、デプロイを行いながらアーカイブされたアプリケーションは、ファイルの圧縮を解凍します。これをアプリケーションの**デプロイ・イメージ・ディレクトリー**といいます。アプリケーションの圧縮を解凍するディレクトリーは、アプリケーション・ファイル名を変更して使用します。myApp.earの場合、圧縮を解凍するイメージ・ディレクトリーは「myApp_ear____」となります。

EARアプリケーションの場合は、EARイメージ・ディレクトリーの下位にモジュールの圧縮を解凍します。モジュールの圧縮を解凍するディレクトリー名は、上記のルールと同様、ejb.jarの場合は「ejb_jar」となります。

【図 1.6】サーバーにデプロイされたEARアプリケーションのイメージ・ディレクトリー



アプリケーションのgenディレクトリー

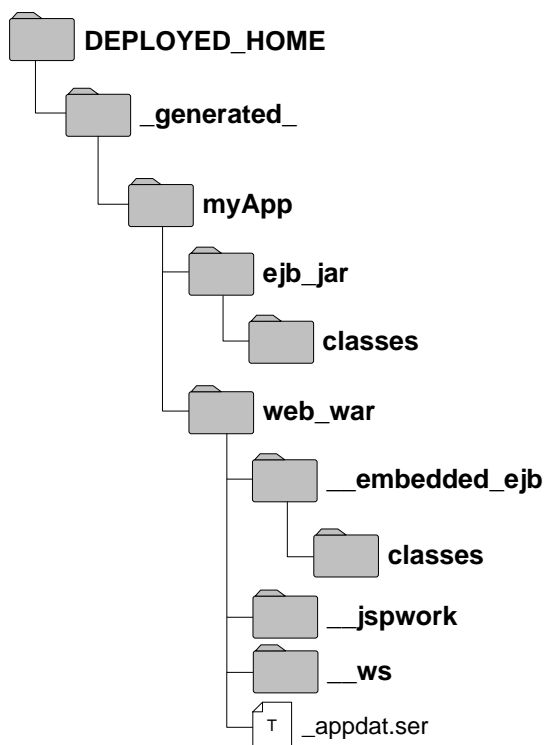
アプリケーションがサーバーにデプロイされることで、サービスに必要なファイルを作成する場合があります。

EJBアプリケーションとWebアプリケーションの場合は、ファイルの作成が必要です。このファイルをDEPLOYED_HOMEの「_generated_」というディレクトリーの下位に作成します。「_generated_」ディレクトリーの下位にアプリケーションIDでディレクトリーを作成し、アプリケーションごとにgeneratedディレクトリーを持つことになります。このディレクトリーをアプリケーションの**genディレクトリー**といいます。

各アプリケーションのgenディレクトリーには、以下のようなファイルが作成されます。

- EJBアプリケーションの場合、EJB 2.x形式のBeanを動的プロキシ方式ではなくスタブ方式で使用する際に、StubファイルとSkelファイルを作成します。
- Webアプリケーションの場合、JSPをJavaで作成したファイルと、組み込み可能EJB(Java EE 6でWebアプリケーション内に組み込まれたEJBモジュールをサポートします。これを組み込み可能EJBといいます)のためのStubファイルを作成します。
- 「_appdat.ser」は、アプリケーションのデプロイ時間を記録したシリアル化・ファイルです。これは、サーバーをシャットダウンしてから再起動する際、アプリケーションが変更されていなければデプロイ・イメージ・ファイルや作成したファイルを再作成しないようにして、起動速度を高めるためのファイルです。

【図 1.7】サーバーにデプロイされたEARアプリケーションのgenディレクトリー



1.1.5. アプリケーションのサービス対象

デプロイ時には、サービスされる対象サーバーを明確にする必要があります。デプロイ可能な対象は、サーバーとクラスターです。Webモジュールの場合、特定のホストのみサービスする場合は、追加で仮想ホストを設定してデプロイすることができます。

サーバー

サーバーはアプリケーションをサービスする最小単位であり、デプロイの対象となります。複数のサーバーを対象にしてデプロイを実行する際、指定したサーバーのうち1つ以上のサーバーでデプロイが失敗したか、ダウンした場合は、配布段階で失敗することになります。この場合、デプロイに失敗したサーバーに適切な措置を取ってから、再度デプロイを実行する必要があります。

また、失敗したサーバーをデプロイ対象から除外して再度デプロイを行うと、デプロイは成功します。サーバーがSTANDBY状態あるいはSUSPENDED状態の場合はサーバーがサービスできない状態であるため、このサーバーを対象にしてデプロイを実行してもアプリケーションは開始されません。

配布まで実行した後、サーバーがサービス可能なRUNNING状態になったとき、該当のアプリケーションを開始します。サーバー状態の詳細については、『*JEUS サーバガイド*』の「3.1.1. Managed Serverのライフサイクル」を参照してください。

サーバーにデプロイではなく配布のみ実行した場合、アプリケーションはサービスされません。サーバーが再起動した場合も、この状態は維持される必要があります。したがって、DASではアプリケーションの状態をINSTALL_HOMEの下位の「.deployInfo.ser」というファイルに記録し、サーバーの起動時にアプリケーションの状態を通知して、以前の作業が続行できるようにします

注

ユーザーが設定ファイルを直接修正した場合はこの状態が維持されないこともあります。したがって、設定ファイルを修正する際は、WebAdminやコンソール・ツールを使用することを推奨します。

クラスター

クラスターに属するサーバーに対しては、サービス対象がサーバーの場合はデプロイできず、サービス対象がクラスターの場合にのみデプロイできます。クラスターを対象にしてデプロイした場合は、クラスター・メンバーに属するサーバーのうち1つでも配布に失敗すると、クラスターのデプロイは失敗となります。

ただし、サーバー・リストを対象としてデプロイする場合とは違って、クラスターを対象としてデプロイする場合は、クラスター・メンバーのうち一部のサーバーが稼働状態でなくてもデプロイすることは可能です。クラスター・メンバーのうち、有効なサーバーすべてにデプロイが成功すれば、クラスターへのデプロイは成功です。

参考

クラスターとは、シングル・サーバーのリストを意味するのではなく、サービスを実行する1つのグループを意味します。クラスター・メンバーのうち稼動状態のサーバーへのデプロイに成功すると、稼動状態ではないサーバーへのデプロイも成功する可能性が非常に高くなります。クラスターのデプロイ時にAlive状態ではないメンバーは、正常化あるいは再起動される際、または新しいメンバーがクラスターに追加されて起動する際、既にクラスターにデプロイされているアプリケーションをデプロイする必要があります。クラスター・メンバーがブートタイム・デプロイに失敗するのであれば、DASは失敗したメンバーのアプリケーションが正常にデプロイされるように保証する必要があります。

デプロイに失敗またはRUNNING状態でないサーバーの場合も、フェイルバックあるいは再起動される際は、デプロイに成功する必要があります。サーバーがブートタイム・デプロイに失敗した場合、DASはデプロイの成功を保証する必要があります。

DASでは、ブートタイム・デプロイに失敗したクラスター・メンバーに対して再度デプロイを行うなど、デプロイを最大限保証できるようにしています。5分間隔で5回行いますが、すべて失敗した場合はJEUSで復旧できないエラーではないと判断し、DASではそれ以上デプロイを保証できないことを通知します。デプロイ時にRUNNING状態ではなかったサーバーだけでなく、クラスターに動的に追加されたサーバーにも、これらの方法でデプロイを保証しています。

DASでクラスター・メンバーに対して指定された回数の再実行がすべて失敗した場合、以下のようなメッセージを出力して、サーバーで直接エラーを解決するよう通知します。

[例 1.2] 指定された回数の再実行がすべて失敗した場合のログ・メッセージ

```
Deployment reattempts of the application[examples] failed 5 times.  
No more deployment attempts will be made. solve the problem for deploying  
application.
```

クラスター・メンバーであるサーバーがアプリケーションのサービス対象である場合、そのサーバーには該当アプリケーションをデプロイしません。該当のアプリケーションをアンデプロイするか、あるいはremove-application-targetコマンドを実行するよう、以下のような警告メッセージを出力します。

[例 1.3] クラスター・メンバーのサーバーに指定されている場合、MSの起動時に発生するログ・メッセージ

```
WARNING: The server[server1] is part of the cluster[cluster1], so this  
application[examples] can only be deployed to the cluster.  
Deploy again this application to the cluster target.
```

上記のような問題が発生するのは、xmlを手動で修正した場合です。これはJEUSで禁止している操作であり、それに伴うエラーについては責任を負いません。

サービス中の独立したサーバーをクラスター・メンバーとして動的に追加する場合は、サーバーでサービスされているアプリケーションをアンデプロイし、アプリケーションのサービス対象からもこのサーバーを削除します。また、クラスターを対象とするアプリケーションも動的にデプロイします。サーバーを対象としてデプロイされ、既にサービスされているアプリケーションではあるが、クラスターに追加して新しいサービスを行おうとしている場合なので、アプリケーションをアンデプロイして新しいアプリケーションをデプロイしても問題ありません。

クラスターに新しいメンバーを動的に追加することが可能であるため、アプリケーションの処理も自動的に行われています。ただし、シングル・サーバーとクラスターを別々に構成してサービス中に、シングル・サーバーをクラスターのメンバーとして動的に追加すると、サーバーで実行中のサービスに問題が生じることがあります。したがって、クラスターを拡張するときは、新しいサーバーを追加し、このサーバーをクラスターのメンバーに追加することを推奨します。

仮想ホスト

Webエンジンでは、仮想ホストを利用して、Webコンテキストを特定のホストでサービスできるようにしています。仮想ホストは単独では指定できず、サービス対象のサーバー、あるいはクラスターと一緒に指定する必要があります。対象となる仮想ホストを指定せずにデプロイする場合は、デフォルトの仮想ホストにデプロイされます。

仮想ホストについての詳細は、『*JEUS Webエンジンガイド*』の「第5章 仮想ホスト」を参照してください。また、デプロイについての詳細は、「[4.3.3. アプリケーションのデプロイ](#)」を参照してください。

1.2. MSでのデプロイ

本節では、MSでアプリケーションまたはモジュールをデプロイする方法について説明します。

1.2.1. ランタイム・デプロイ

ランタイム・デプロイとは、サーバーが起動している状態でアプリケーションまたはモジュールをデプロイすることです。JEUSのツールであるWebAdminとコンソール・ツールを利用して、DASに接続してランタイム・デプロイを行うことができます。

JEUS 7以降のバージョンからは、ランタイム・デプロイを実行すると**永久デプロイ**になります。永久デプロイは、デプロイや配布命令によってアプリケーションの配布に成功すると、その内容がdomain.xmlに記録されます。DASやMSが再起動しても、当該アプリケーションのデプロイ情報を設定ファイルから読み込み、ブートタイム・デプロイできるようにアプリケーション情報を保持することを意味します。

デプロイはDASを介してのみ可能であるため、domain.xmlにアプリケーション情報を記録する主体もDASのみです。MSでは、ブートタイムにDASから設定内容を受け取り、その設定に登録されているアプリケーション・ファイルをダウンロードしてXMLファイルの情報を基にブートタイム・デプロイを行うことができます。

ランタイム・デプロイの実行方法についての詳細は、「[4.3.3. アプリケーションのデプロイ](#)」を参照してください。

1.2.2. ブートタイム・デプロイ

ブートタイム・デプロイとは、domain.xmlに登録されているアプリケーションをサーバーの起動時にデプロイすることをいいます。JEUS 7以降のバージョンからは基本的に、デプロイしたアプリケーションの情報がdomain.xmlに記録されるため、以前正常にデプロイされた履歴のあるアプリケーションはブートタイム・デプロイの対象となります。

DEPENDENT状態でのブートタイム・デプロイ

MSは起動時にDASから設定ファイルのdomain.xmlを取得します。この設定ファイルに登録されているアプリケーションがある場合、アプリケーションの対象を確認して、自身のサーバーにデプロイする必要があるアプリケーションなのかどうかを判断してデプロイを行います。この際、アプリケーションごとにデプロイを行うのではなく、まず、すべてのアプリケーションの配布を行い、それらがすべて成功した場合にアプリケーションの開始を行います。

参考

サーバーがクラスターのメンバーではあるが、アプリケーションのサービス対象がサーバーと明示されている場合、アプリケーションはブートタイムにデプロイされません。クラスターに属するサーバーはデプロイの対象になることはできず、クラスター単位でのみデプロイできます。

これらは、JEUSのツールを使用してデプロイを実行した場合は保証されますが、手動でXMLファイルを修正した場合は保証できないので、XMLファイルを修正しないようにしてください。

ブートタイム時にサーバーで行う配布は、ランタイム時にサーバーで行う配布と大きく変わりません。まず、DASからアプリケーション・ファイルを取得します。以前取得したファイルがサーバーに存在する場合は、サーバーにキャッシュされているアプリケーション・ファイルとDASから取得するアプリケーション・ファイルのインストール時間を比較して、ファイルを取得するか否かを決定します。その後、アプリケーション・ファイルの圧縮を解凍してクラスをロードし、アプリケーションがサービスできる環境を構成する配布を行います。

サーバーの起動時に1つのアプリケーションでも配布に失敗するとサーバーはSTANDBY状態になり、その時点で起動が中止されます。このような状況で**start-server**コマンドを実行すると、配布に失敗したアプリケーションを再び配布します。ブートタイムと同様に、すべてのアプリケーションが配布に成功しなければアプリケーションの開始が行われません。この際も、配布に失敗したアプリケーションが存在すると、サーバーはSTANDBY状態になります。

アプリケーション全体の配布が成功していなくても、サーバーをRUNNING状態にしてサービスしたい場合は、以下のコマンドを実行します。

```
[DAS]domain1.adminServer>start-server -force server1
```

start-serverコマンドに-forceオプションを指定して実行すると、配布が成功したアプリケーションに対してのみ開始を行い、サーバーをサービス可能な状態にできます。この際、サーバーの状態はRUNNINGになります。

ブートタイム・デプロイを行う際、MSは配布が完了した時点で、DASから該当のアプリケーションが開始できる状態であるかどうかを確認します。

DASでの状態がDEPLOYEDまたはRUNNINGのアプリケーションに対してのみ開始を実行できます。DASからアプリケーションの状態を読み取れなかった場合、サーバーはアプリケーションの開始を行いません。この場合、サーバーは起動を中断し、STADNBY状態になります。

DASでのアプリケーション状態に関係なく、アプリケーションを開始してサービス可能にしたい場合は、配布に失敗したアプリケーションが存在する場合と同様、start-serverコマンドに-forceオプションを指定して実行し、サーバーをRUNNING状態にします。

参考

DASでのDISTRIBUTED状態のアプリケーションに対して、サーバーでは配布までのみ行うため、DASでは別途ファイルにこの情報を記録して、DASがダウンして再起動した場合でもアプリケーションの状態を保つようになっています。

サーバーがSUSPENDED状態でアプリケーションが停止、開始した場合も、サーバーが再びレジュームしてRUNNING状態になったとき、変更されたアプリケーションの状態が維持できるようにする必要があります。この場合は、DASからアプリケーションの状態を再び取得することはありません。SUSPENDED状態であっても、DASからアプリケーションの停止、開始命令を受けることはできるため、実際にアプリケーションがMSで命令を実行できない状態であっても、DASからの開始、停止命令を記憶しておき、サーバーが再びレジュームしたときにアプリケーションの状態を維持できるようにしています。

参考

サーバーがSUSPENDED状態のときにアプリケーションの停止命令を受けると、サーバーでは既にすべてのアプリケーションが停止した状態であるため、ApplicationAlreadyStoppedExceptionが発生します。DASでは、このような状況で発生した例外はアプリケーションの停止が失敗とは見なしません。

サーバーがSUSPENDED状態のときにアプリケーションの開始命令を受けると、サーバーではアプリケーションの開始を実行できません。この際、サーバーがSUSPENDED状態になる前にアプリケーションがDISTRIBUTED状態であった場合、サーバーがレジュームすると同時に、このアプリケーションを開始する必要があります。

INDEPENDENT状態でのブートタイム・デプロイ

INDEPENDENT状態でサーバーを起動した場合、DASの管理を受けません。また、DASと接続できない状況であるため、アプリケーション・ファイルも取得できません。したがって、ブートタイム・デプロイを行う際も、以前取得したアプリケーション・ファイルがないか正しくない場合は、デプロイを行うことはできません。

INDEPENDENT状態でサーバーを起動する際は、アプリケーション・ファイルに接続できる場合にのみデプロイを行うことができます。アプリケーション・ファイルを接続できる場合とは、以下のようなケースをいいます。

- MSがDASと同じマシンに存在し、DASにインストールされているアプリケーションのソース・ファイルに接続可能な場合
- アプリケーション・ファイルがNAS(network-attached storage)に存在しており、MSでも接続可能な場合
- 圧縮を解凍したデプロイ・イメージ・ディレクトリーが残っている場合

上記以外の場合は、デプロイするアプリケーション・ファイルが見つからないため、デプロイに失敗します。この場合も、サーバーはSTANDBY状態になり、起動が中止されます。

この場合は、コンソール・ツールを使用してMSに接続し、**local-start-server**コマンドを使用してサーバーをRUNNING状態にします。

参考

local-start-serverコマンドを実行してもデプロイが成功する可能性が非常に低いため、この場合は-forceオプションを指定してサーバーを開始してください。

INDEPENDENT状態でサーバーを起動する際は、DEPENDENT状態とは違って、配布のみ行うアプリケーション・リストをDASから取得することができません。したがって、配布に成功したアプリケーションすべてを開始させ、サービス可能な状態にします。

INDEPENDENT状態で起動したサーバーがDASと接続されDEPENDENT状態になったら、オプションに応じて配布に失敗したアプリケーションの配布を再実行します。すべてのアプリケーションの配布に成功したら、DASから開始するアプリケーション・リストを取得して開始させます。

また、配布のみ行う必要があるアプリケーションが開始されサービスされている場合、このアプリケーションは停止させます。ただし、サーバーの起動中にDASがフェイルバックされてDEPENDENT状態になった場合は、上記の操作は不要です。

1.2.3. アプリケーションの同期化

アプリケーション・ファイルの同期化は、DASの主な役割の1つです。以下のような場合に、DASとMSの間でアプリケーション・ファイルが同期化されます。

- ランタイム・デプロイを実行する場合

MSでは、新しくデプロイするアプリケーション・ファイルをDASから取得します。

- ブートタイム・デプロイを実行する場合

MSにデプロイするアプリケーション・ファイルが変更された場合、MSはDASからアプリケーション・ファイルを再度取得します。

- MSがINDEPENDENT状態からDEPENDENT状態に変わった場合

MSがINDEPENDENT状態からDEPENDENT状態に変わり、DASの管理を受けるようになった場合、オプションを使ってアプリケーションの同期化を行います。ドメインの「**enable-to-resynchronize-applications**」設定をtrueにした場合にのみ同期化を行います。デフォルト値はfalseであり、同期化を行いません。このオプションがtrueに設定された場合、MSでは、デプロイに失敗したアプリケーション・ファイルを再度取得します。また、デプロイに成功したアプリケーションのうち、DASで管理するアプリケーション・ファイルが変更された場合も、アプリケーション・ファイルを再度取得します。

DASでは、アプリケーション・ファイルだけでなく、アプリケーションのドメインでの状態も同期化します。

ブートタイム・デプロイの実行時、DASでアプリケーションがDISTRIBUTED状態で管理されている場合、MSでもDISTRIBUTED状態にします。このアプリケーションをサービスするためには、**start-application**コマンドを実行する必要があります。

MSがINDEPENDENT状態からDEPENDENT状態に変更された場合、MSではアプリケーション・ファイルだけでなく、状態も同期化します。DEPENDENT状態のMSでは、DASで管理しているアプリケーションの状態に従って、MSでサービスされているアプリケーションの状態も変更されます。

INDEPENDENT状態のMSでは、すべてのアプリケーションがサービスされるようにRUNNING状態にしましたが、DASと接続されDEPENDENT状態になった後はDASの状態に従います。

DASでアプリケーションがDISTRIBUTED状態の場合、MSのアプリケーションも中止してDISTRIBUTED状態にします。また、DASで既にアンデプロイしたアプリケーションがMSに存在する場合は、MSからもアンデプロイします。

1.3. 自動再デプロイ

JEUS 6までは、指定されたディレクトリまたはアプリケーションを定期的にチェックし、変更内容がある場合は自動的にデプロイを行うことが可能でした。この機能を使用するとアプリケーションを手動でデプロイする必要がないため、アプリケーションの開発時や頻繁なアップデートが必要なアプリケーションの場合に有効

です。JEUS 7以降のバージョンからは、自動デプロイ機能をサポートしません。ドメインでアプリケーションの変更を検知して再デプロイする機能のみサポートします。

参考

JEUS 7からドメイン構造に変更され、アプリケーションを管理する主体がDASとなったため、自動デプロイの提供が困難となりました。アプリケーションの対象をドメイン単位にすることが難しく、Explodedモジュールの場合は他のマシンに送信されないため、自動デプロイ機能は提供しません。

その代わり、初めてデプロイを行う際、アプリケーションのサービス対象と自動再デプロイをチェックする周期を設定して手動でデプロイすると、アプリケーション・ファイルの変更を検知して、アプリケーションのサービス対象に自動再デプロイを行うことができます。

自動再デプロイの設定をしたアプリケーションに対しては、設定した周期ごとにアプリケーション・ファイルの変更をチェックします。

- Archiveモジュールの場合は、アクティブ・ファイルの最終更新日時でアプリケーションの変更を検知して再デプロイします。
- Explodedモジュールの場合は、標準デプロイメント記述子(以下、DD)の最終更新日時でアプリケーションの変更を検知して再デプロイします。

DASが対象となるサーバーに再デプロイを命令します。サーバーでは、自動再デプロイと一般的な再デプロイの動作に違いはありません。DASでは、すべてのサーバーへの再デプロイが成功してから成功と見なします。ファイルの変更後、すぐに再デプロイされないこともあります。

アプリケーション・ファイルの変更の有無は設定周期ごとにチェックされます。周期のデフォルト値は10秒です。

1.4. ディレクトリー・モードのデプロイ

アプリケーションはJava EEの仕様で定義した各アプリケーション・タイプに合う形式で圧縮されている必要があります。ただし、開発への利便性のため、圧縮が解凍されたアプリケーションもデプロイできるようにサポートしています。JEUSではこのようなディレクトリー形式のアプリケーションを**Explodedモジュール**といいます。

DASと同じマシンに存在するサーバーやクラスターを対象としている場合、あるいは、DASと別のマシンに存在しているか、NAS(Network Attached Storage)のように他のマシンからもアクセス可能なパスに存在する場合にのみデプロイ可能です。このようなアプリケーションは、**install-application**コマンドを使用してDASのAPPLICATION_INSTALL_HOMEにインストールするのではなく、アプリケーションの上位ディレクトリーとしてWebAdminの「**Application Repository**」項目を指定するか、-pathオプションを指定してデプロイを行います。オプションとしてアプリケーションのIDを指定できますが、IDを指定しない場合はアプリケーション・ファイル名がIDとなります。

注

JEUSは、ディレクトリー・モードでデプロイしたアプリケーションの同期化や管理を行わないので、該当のディレクトリーはユーザーが管理する必要があります。

1.5. アプリケーション・リポジトリー

ドメイン環境では、該当のドメインでサービスされるすべてのアプリケーションをDASで管理するため、アプリケーションをデプロイする前にアプリケーション・ファイルをDASにインストールする必要があります。ただし、ドメインにアプリケーション・リポジトリーを追加した場合は、アプリケーション・ファイルをインストールしなくてもアプリケーションをデプロイすることができます。

1.5.1. アプリケーション・リポジトリーの追加/削除/照会

ドメインでは、DASにインストールされたアプリケーションを一箇所に集めて管理します。このディレクトリーには手動でアプリケーションを格納することはできません。複数のリポジトリーを設定することが可能であり、設定したディレクトリーはすべてアプリケーション・リポジトリーとして認識されます。アプリケーション・リポジトリーを追加および削除し、設定されている情報を照会するには、WebAdminやコンソール・ツールを使用します。

ドメインにアプリケーション・リポジトリーとして設定したディレクトリーにはJava EEアプリケーションを格納することができます。ArchiveモジュールだけでなくExplodedモジュールも可能です。

ドメインに登録されているアプリケーション・リポジトリーには、install-application/uninstall-applicationコマンドを使ってアプリケーション・ファイルを追加および削除することはできません。アプリケーション・ファイルを手動で追加および削除すると、ドメインでこれを自動認識できます。このように格納したアプリケーションは、ドメインでINSTALLED状態と認識されます。削除されたアプリケーション・ファイルは、アプリケーションの情報から除去され、以降はドメインの管理から外されます。

アプリケーションのIDはアプリケーション・ファイルの名前であるため、デプロイを実行する際にファイル名を-idオプションで指定します。

アプリケーション・ファイルをアップデートする場合は、このリポジトリーに存在するファイルを手動でアップデートします。アプリケーションをデプロイする際に自動再デプロイの周期を設定した場合は、DASでファイルの変更の有無を自動で検知して自動的に再デプロイを行います。

アプリケーション・リポジトリーを追加する際、リポジトリーに存在するファイル名がドメインで既に使用されているアプリケーションIDの場合、該当のアプリケーションはドメインで認識されず、以下のようなログ・メッセージが発生します。

```
[2016.08.08 12:37:40.625][1] [adminServer-68] [Deploy-0005] WARNING: The application[myApp] already exists on /JEUS_HOME/.applications/myApp.ear. Change the file name of [/home/user1/apps/myApp].
```

重複するアプリケーションIDを使用した場合、アプリケーション情報を要求するかDASを開始した場合、同じログが発生します。

また、追加するアプリケーション・リポジトリに有効なアプリケーションが存在しない場合、以下のようなログ・メッセージが発生します。

```
[2016.08.08 12:37:40.625][1] [adminServer-68] [Deploy-0062] WARNING: The repository(/home/user1/apps) does not contain any valid applications, but new applications can be added.
```

アプリケーション・リポジトリを削除する際、リポジトリに存在するファイルがドメインに配布されているか、デプロイされている場合、アプリケーション・リポジトリは削除されますが、アプリケーションがドメインの管理対象から削除されず、以下のようなログ・メッセージが発生します。

```
[2016.08.08 12:44:04.328][1] [adminServer-93] [Deploy-0124] WARNING: The application[myApp] is RUNNING in repository[/home/user1/apps]. To remove this application from the domain, undeploy it.
```

注

JEUSは、アプリケーション・リポジトリを削除する際、サービスされているアプリケーションまで削除するかどうかを判断するのは困難です。したがって、サービスされているアプリケーションを削除したい場合は、まずアンデプロイを行う必要があります。

WebAdminの使用

以下は、WebAdminを使用してアプリケーションのリポジトリを追加する手順についての説明です。

1. WebAdminの左側のメニューで**[Domain]**を選択すると、**Domain**設定画面に移動します。リポジトリを追加するため、左側のメニュー下段の**[LOCK & EDIT]**ボタンを選択します。
2. **Domain**設定画面で「**Application Repositories**」項目に追加するリポジトリ・ディレクトリを入力します。**[確認]**ボタンをクリックすると、変更された設定が保存され、画面の上段に保存結果についてのメッセージが表示されます。

「**Application Repositories**」項目の**[入力]**ボタンをクリックすると、**Navigation**ポップアップ・ウィンドウが表示され、アプリケーション・リポジトリのパスを選択できます。
3. **[Activate Changes]**ボタンをクリックして、追加したアプリケーション・リポジトリをドメインに反映します。

4. **Activate Changes**ポップアップ・ウィンドウで変更する事項に関する説明を作成できます。「**Description**」項目で変更についての説明を作成し、**[確認]**ボタンをクリックします。
5. サーバーのアクティブ化が完了すると、設定したアプリケーション・リポジトリ・ディレクトリーがドメインに追加され、反映結果が表示されます。
6. WebAdminの左側のメニューで**[Applications]**を選択して**[Deployed Applications]**画面に移動します。ドメインに追加したアプリケーション・リポジトリにあるアプリケーションがINSTALLED状態でドメインに登録されたことを確認できます。

コンソール・ツールの使用

ドメインにアプリケーション・リポジトリを追加するには**add-application-repository**コマンドを使用し、削除するには**remove-application-repository**コマンドを使用します。インストール時に設定するIDでアプリケーション管理ディレクトリーの下位にディレクトリーを作成し、そこにアプリケーションのアーカイブ・ファイルを格納します。

以下は、コンソール・ツールを用いてアプリケーション・リポジトリを追加、削除、照会する例です。

```
[DAS]domain1.adminServer>add-application-repository /apps/test
Successfully performed the ADD operation for An application repository.
Check the results using "add-application-repository or
list-application-repositories"

[DAS]domain1.adminServer>list-application-repositories
Application Repositories
=====
+-----+
| Path to Application Repository |
+-----+
| /apps/test |
+-----+
=====

[DAS]domain1.adminServer>remove-application-repository /apps/test
Successfully performed the REMOVE operation for An application repository.
Check the results using "remove-application-repository or
list-application-repositories"

[DAS]domain1.adminServer>list-application-repositories
Application Repositories
=====
+-----+
| Path to Application Repository |
+-----+
```

(No data available)

参考

コンソール・ツールを用いてアプリケーション・リポジトリを追加、削除、照会する方法についての詳細は、『JEUS リファレンスガイド』の「4.2.6.1. add-application-repository」と『JEUS リファレンスガイド』の「4.2.6.11. remove-application-repository」、および『JEUS リファレンスガイド』の「4.2.6.9. list-application-repositories」を参照してください。

1.5.2. アプリケーション・リポジトリにあるアプリケーションのデプロイ

アプリケーション・リポジトリを追加した後、リポジトリに存在するアプリケーションをWebAdminとコンソール・ツールを使用してデプロイできます。詳細については、[4.3.3.2節「コンソール・ツールの使用」](#)を参照してください。

1.6. パスを指定してデプロイ

ドメイン環境で推奨するアプリケーション・ファイルの管理方法は2種類に分けられます。

アプリケーション・ファイルをドメインで管理する場合は、アプリケーションをINSTALL_HOMEにインストールします。また、ユーザーが直接アプリケーション・ファイルを管理する場合は、アプリケーション・リポジトリを設定します。アプリケーション・ファイルを直接管理する場合でも、実際の運用環境ではドメインで使用するアプリケーション・ファイルを業務別にまとめて、複数のディレクトリに格納することをお勧めします。

ただし、開発時には上記のようにまとめず、それぞれ異なるパスに格納する場合もあるので、利便性を高めるために-pathオプションを指定してデプロイできるようにしています。

パスを指定してアプリケーションをデプロイするときは、-idオプションを設定できません。ExplodedモジュールとArchiveモジュールのいずれもデプロイ可能であり、アプリケーション・ファイルのパスを-pathオプションで指定してデプロイしたアプリケーションは、ドメインのAPPLICATION_INSTALL_HOMEにコピーせず、DASに登録してデプロイを行います。この場合、アプリケーションのIDはアプリケーション・ファイル名をそのまま使用します。

WebAdminとコンソール・ツールを用いて、-pathオプションを指定してデプロイする方法についての詳細は、[4.3.3.3節「コンソール・ツールの使用」](#)を参照してください。

1.7. ステージング・モード・デプロイ

Explodedモジュールの場合、DASと同じマシンに存在するMSIにのみデプロイできます。Explodedモジュールは主に開発段階で使用しており、複数のマシンを使用しないため、同じマシンでのみExplodedモジュールをデプロイおよびサービスできます。ただし、テスト段階では、アプリケーションをExplodedモジュール形式に

維持しつつ、複数のマシンでドメインを構成してテストできるため、Explodedモジュールを他のマシンにデプロイする方法が必要です。

アプリケーションのデプロイ時に**-staging**オプションを指定してデプロイすると、Explodedモジュールを圧縮して、他のマシンに存在するMSにも転送できます。ステー징環境にデプロイされたアプリケーションは、MSではArchiveモジュールと同様に取り扱われます。圧縮を解凍してデプロイし、MSが再起動する際は、DASとアプリケーション・ファイルを同期化します。

アプリケーションの内容が変更された場合は、redeployコマンドを使って変更されたファイルをMSに適用し、再デプロイできます。redeployコマンドを実行する際は、-stagingオプションを指定する必要がなく、既存のArchiveモジュールを再デプロイする場合と同様にredeployコマンドを実行します。ただし、この際は-pathオプションを指定してはなりません。MSでは、redeployコマンドを受信すると、優先的にアプリケーション・ファイルの同期化を行い、既存のアプリケーションをアンデプロイした後、新しいアプリケーションをデプロイします。

Webadminとコンソール・ツールを用いて、-stagingオプションを指定してデプロイする方法についての詳細は、[「4.4. ステージング・モード・デプロイ」](#)を参照してください。

1.8. アプリケーションのアンデプロイ

JEUSの管理ツールを利用して明示的にアプリケーションをアンデプロイする場合と、サーバーのシャットダウン時にアプリケーションがアンデプロイされることは、動作方式が異なります。

以下は、2つの動作の違いです。

	管理ツールによるアンデプロイ	サーバーのシャットダウン時のアンデプロイ
ファイルの削除可否	サーバー内部で作成したファイルを削除します。たとえば、JSP、EJBコンパイル結果が削除され、EJB 3.0スケジュール情報がDBから削除されます	サーバー内部で作成したファイルを削除しません
適用されるタイムアウト	Graceful Undeploy Timeout	Graceful Shutdown Timeout (Undeploy Timeoutは適用されません)

第2章 グレースフル・アンデプロイとグレースフル再デプロイ

本章では、アプリケーションをアンデプロイして再デプロイする際、サービス中の要求をすべて処理してから、アンデプロイや再デプロイを行うグレースフル・アンデプロイ(Graceful Undeploy)とグレースフル再デプロイ(Graceful Redeploy)について説明します。

2.1. グレースフル・アンデプロイ

グレースフル・アンデプロイとは、アンデプロイの命令を受けた場合、サービス中のユーザー要求をすべて処理してから、アンデプロイを行うことをいいます。

EAR、EJB、Webアプリケーションは、アンデプロイ命令が受信された場合、ユーザーの要求を処理してからアンデプロイを実行するグレースフル・アンデプロイを行います。ただし、EJBの場合はSessionBeanのみグレースフル・アンデプロイが可能です。

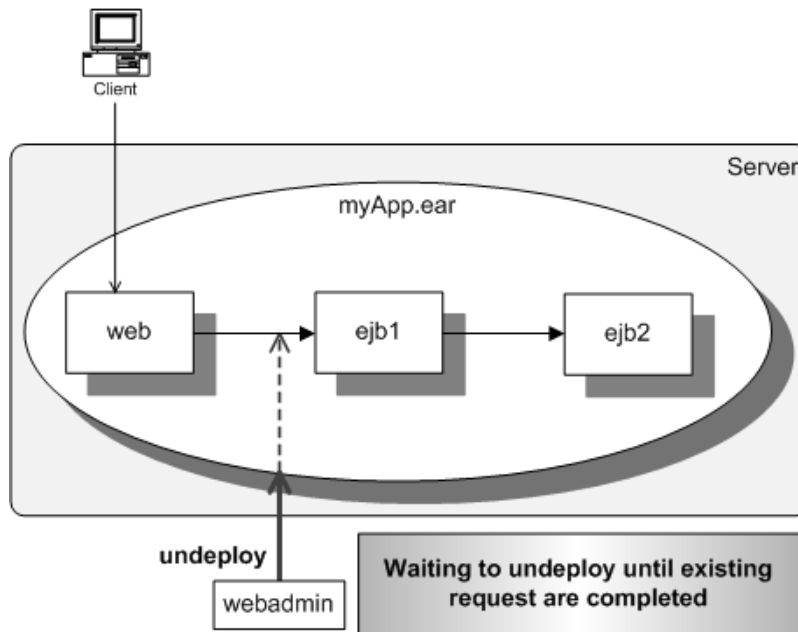
アプリケーションのアンデプロイ時にユーザーからの要求がまだ処理されていない場合、要求処理がすべて完了まで待機してからアンデプロイを行います。アンデプロイを実行する際、`-gracefultimeout`をオプションとして指定することが可能であり、ここに設定した値に応じて、要求処理が完了する前にアンデプロイが実行されることもあります。まだ要求処理が完了していない状態で、グレースフル・タイムアウトによってアンデプロイが行われる場合は、処理中の要求スレッドに割り込みシグナルを送ります。

アンデプロイ命令を受信すると、アプリケーションはサービスできないDISTRIBUTED状態に変わり、処理中の要求が完了するまで、またはユーザーが指定したグレースフル・タイムアウトになるまで待機します。グレースフル・アンデプロイを行うには、ユーザーがアンデプロイを実行する際、適切な`-gracefultimeout`をオプションとして指定する必要があります。`gracefultimeout`オプションを指定しない場合は、5分に設定されます。この場合、5分の間のみの処理中の要求の完了を保証します。

アプリケーションがDISTRIBUTED状態の場合、外部から受信した要求は処理できません。ただし、EARアプリケーションの場合はアンデプロイの開始後にもアプリケーション内の要求は実行されます。スタンドアロンEJBの場合も、Beanが複数であれば、EJBアプリケーションのアンデプロイが開始されたとしても、Bean間の要求は処理されます。

以下の図では、要求が処理される状況を示しています。

【図 2.1】 EARアプリケーションのグレースフル・アンデプロイ



上の図では、webでejb1を呼び出したときはアプリケーションがRUNNING状態であるため、要求が処理されることに問題はありません。ただし、ejb1がejb2を呼び出した時点ではアプリケーションがDISTRIBUTED状態に変わっているため、要求が受信されても拒否されるべき状況であると考えがちです。しかし、これは外部アプリケーションからの要求ではなく、同じEARアプリケーション内での要求であるため、アプリケーションがDISTRIBUTED状態になっていても、要求が処理されることを保証しなければなりません。

上の図のように、webでejb1を呼び出し、webの呼び出しを受けたejb1でejb2を呼び出すコール・スタックは1つの要求です。つまり、EARアプリケーションの観点では、ejb1からejb2への要求は新しい要求ではなく、Webから始まった、既に進行中の要求です。外部アプリケーションがこのアプリケーションのejbを要求した場合であれば、アプリケーションの状態に応じて要求が拒否されます。しかし、EARは1つのアプリケーションであるため、EARがDISTRIBUTED状態になったからといってejb2の要求が失敗してはなりません。ejb2の要求は引き続きサービスされる必要があり、この要求が完了した後にアンデプロイされるように保証します。要求が長時間かかることもあるため、アンデプロイ時に指定したタイムアウトが、要求が処理される時間より短い場合は、指定された時間の間待機してからアンデプロイが行われます。この際、要求を処理中のスレッドには割り込みシグナルを送り、処理中の作業を中止します。

参考

割り込みシグナルを送ったからといって、すべての処理が中止されるわけではありません。スレッドで処理中の作業によっては、中止されることもあれば無視されることもあります。割り込みによって作業が中止されても、JEUSでは該当のスレッドに対して別途処理は行いません。アプリケーションの要求をサービスするスレッドで割り込みがあったため、アプリケーションがこの状況を考慮して後処理する必要があります。スレッドの割り込みの詳細については、『JEUS サーバガイド』の「3.3.2. スレッドの制御」を参照してください。

1つのアプリケーション内での要求に対するグレースフル・アンデプロイは、EARだけでなく、スタンドアロンEJBアプリケーションにも適用されます。EJBアプリケーションにBeanが複数存在する場合、このBean間の要求も同じアプリケーション内の要求であるため、アンデプロイ命令が受信されても、処理中の要求はすべて完了する必要があります。

undeployコマンドは基本的に5分のタイムアウトが適用されます。5分の間にアプリケーションの要求を保証するのが基本動作です。-gracefultimeoutオプション(-toオプションと同じ)を指定して、アプリケーションの要求を保証する時間を変更することが可能です。また、値を大きく指定してグレースフル・アンデプロイを実行することもできます。

JEUS 6の基本動作のように、処理中の要求を待機せずアンデプロイを行うためには、-force(-f)オプションを使用します。この場合、処理中のアプリケーションの要求は保証されません。グレースフル・アンデプロイの方法については、「[4.3.5. アプリケーションのアンデプロイ](#)」を参照してください。

2.2. グレースフル再デプロイ

グレースフル再デプロイとは、現在サービス中のアプリケーションを停止することなく、新しく再デプロイしたアプリケーションでサービスを提供するメカニズムです。この機能はアプリケーションのタイプ別に動作が異なります。

本節では、グレースフル再デプロイを使用するために必要な知識と、各アプリケーションのタイプ別の動作について説明します。

2.2.1. グレースフル再デプロイを使用するための前提条件

以下は、グレースフル再デプロイを使用するための前提条件です。

- WARファイルやJARファイルのようにパッケージングされたアプリケーションである必要があります。
- ディレクトリー形式のアプリケーションの場合、旧アプリケーションと新アプリケーションのディレクトリーが異なる必要があります。ディレクトリー形式のアプリケーションはソース・ディレクトリーをそのままサービス・イメージとして使用するので、旧アプリケーションと新アプリケーションが同じディレクトリーを指すと、グレースフル再デプロイの要件を満たされません。ただし、ディレクトリー形式のアプリケーションであっても、ステージング・モードでデプロイした場合は、旧アプリケーションと新アプリケーションが同じディレクトリーを指すことができます。ステージング・モードでは、ソース・ディレクトリーとは別に、サービス・イメージを旧アプリケーションと新アプリケーションそれぞれに作成するためです。
- アプリケーション・リポジトリにあるアプリケーションに対しては、新アプリケーションを明示する再デプロイはできません。アプリケーション・リポジトリにあるアプリケーションは固定的にドメインにインストールされているという属性を持つので、再デプロイによる新アプリケーションへの代替を許可しません。そのため、アプリケーション・リポジトリにあるアプリケーションは、新アプリケーションの明示を前提とするグレースフル再デプロイも実行できません。

- アプリケーションをパッケージングする際、META-INF/MANIFEST.MFファイルに以下のフィールドを追加します。

また、ディレクトリー形式のアプリケーションにもMETA-INF/MANIFEST.MFファイルを追加し、このフィールドを追加する必要があります。

```
Jeus-Use-Graceful-Redeploy: true
```

参考

MANIFEST.MFファイルのルール上、ファイルの最終行にはnew line(\n)が存在する必要があります。

-
- 再デプロイの対象となるアプリケーションも上記のフィールドが必要です。そうでない場合は一般的な再デプロイが実行されます。
 - ディレクトリー形式のアプリケーションの場合、旧アプリケーションと新アプリケーションの標準DDにアプリケーション名とモジュール名を同じく設定する必要があります。
 - パッケージングされたファイルの最終更新日時が異なる必要があります。
 - EJBの場合、セッションBeanに対してのみグレースフル再デプロイがサポートされます。
 - SHAREDモードのアプリケーションの場合、グレースフル再デプロイをサポートしません。この場合、EARでパッケージングして使用することを推奨します。

2.2.2. グレースフル再デプロイの必須考慮事項

グレースフル再デプロイの実行時に以下の事項を考慮する必要があります。

- サービス開始前にグレースフル再デプロイ機能の使用可否を決定して、前提条件で説明したようにアプリケーションを準備します。
- 既存アプリケーションをすぐにアンデプロイするのではなく、一定時間の間は2つのアプリケーションがJVMに同時に存在します。したがって、JVMのpermgen領域のサイズを事前に算定しておく必要があります。permgen領域が十分に確保されていない場合は、OutOfMemoryErrorが発生することがあります。
- JNIライブラリーを使用する場合、JVMで同時に同じライブラリーをSystem.loadLibrary()でロードできないため、グレースフル再デプロイを使用できません。

2.2.3. グレースフル再デプロイの使用法

グレースフル再デプロイはredeployコマンドを使って実行できます。使用法は、サービス中に実際に発生し得る状況を例にして説明します。

まず、以下のようにパッケージングされたWARファイルがあると想定します。

- old application : myservice.war (08/09/2016 1:00 pm)
- new application : myservice.war (08/25/2016 11:00 am)
- install id = myservice_war, context path = /myservice

サービス提供者は、2016年8月25日午前に、5月9日にパッケージングしてデプロイしたmyservice.warに問題があることを発見しました。そのため、これを修正して午前11時に新しくパッケージングしました。しかし、既にmyservice.warを使用している顧客が存在しており、JEUSを停止できないため、グレースフル再デプロイを使用することに決定しました。

このような状況で、サービス提供者はコンソール・ツールを使用して以下のようなコマンドを実行します。

```
redeploy myservice_war -path /path_to_new_application/myservice.war
```

redeployコマンドに特別なオプションを指定するのではなく、JEUSでは、META-INF/MANIFEST.MFファイルのフィールドとパッケージング時間をベースにしてグレースフル再デプロイの実行可能性を判断します。2016年8月9日午後1時にパッケージングしたアプリケーションのMETA-INF/MANIFEST.MFファイルに前提条件で説明したフィールドがないか、falseに指定されている場合、ノーマル再デプロイを実行します。

参考

クライアントの要求を処理するのに多くの時間を要する場合は、以前のアプリケーションを指定された時間の間に処理できるよう、アンデプロイ・タイムアウトの値を適切に設定して使用してください。

以前のバージョンのアプリケーションを強制的にアンデプロイする方法

グレースフル再デプロイが正常に終了した時点を基準にして、新しいサービス・ユーザー（顧客）は新しいmyservice.warを使用することになります。既存のサービス・ユーザーは引き続き以前のバージョンのmyservice.warを使用します。既存のサービス・ユーザーと新しいサービス・ユーザーを区別する基本的な基準はコネクション（ソケット接続）です。Webアプリケーションの場合はHTTPセッションもその基準に含まれます。

グレースフル再デプロイの目的は、既存のサービス・ユーザーがサービスの停止を経験しないようにすることです。したがって、以前のバージョンのmyservice.warに接続したコネクションや、それによって作成されたHTTPセッションがすべて削除されるまで、以前のバージョンのmyservice.warはアンデプロイされません。

HTTPセッションのタイムアウトが長過ぎて、一部のHTTPセッションによってアンデプロイされずに残っている場合は、サービス管理者が以下のように`undeploy`コマンドを実行して、強制的にアンデプロイできます。

```
undeploy -old myservice_war
```

以前のバージョンのアプリケーションを一定時間が過ぎた後に自動的にアンデプロイする方法

一定時間が過ぎると、以前のバージョンの`myservice.war`が自動的にアンデプロイされるようにできます。`redeploy`コマンドを実行する際に`Timeout`オプションを指定します。タイムアウト値の単位は秒です。

```
redeploy -path /path_to_new_application/myservice.war -to 5
```

以前のバージョンのアプリケーションと新しいアプリケーションが共存する場合、以前のバージョンのアプリケーションにロールバックする方法

以前のバージョンの`myservice.war`がアンデプロイされていない状態で、サービス提供者が新しい`myservice.war`の問題を発見しました。この際、新しい`myservice.war`をアンデプロイすると、以前のバージョンの`myservice.war`にロールバックされ、正常にサービスされます。以下のように`undeploy`コマンドを実行します。

```
undeploy -new myservice_war
```

注意

新しいアプリケーションに多少深刻な問題があり、即時アンデプロイが必要な状況であると想定して新しいアプリケーションを直ちにアンデプロイします。それによって、新しいアプリケーションを使用していたユーザーはサービス停止を経験することになります。このような状況を避けたい場合は、以前のバージョンの`myservice.war`がアンデプロイされた後、その`myservice.war`を再デプロイします。

新しいアプリケーションを配布のみ行う方法

新しい`myservice.war`が正常に動作することを確認してからユーザーに提供するには、`redeploy`コマンドを実行する際に配布のみを行うように指定します。

```
redeploy myservice_war -path /path_to_new_application/myservice.war -distonly
```

新しいアプリケーションのサービス動作の可否を事前に確認する方法は、アプリケーションのタイプに応じて異なります。新しいWebアプリケーションの動作についての詳細は、[2.2.4節「新しいWebアプリケーションの動作可否を事前に確認する方法」](#)を参照してください。

動作を確認してから、新しいアプリケーションをサービスするには、**start**コマンドを実行します。

```
start -new myservice_war
```

新しいアプリケーションに問題があってアンデプロイする場合は、**-new**オプションを指定して**undeploy**コマンドを実行します。

```
undeploy -new myservice_war
```

以前のバージョンのアプリケーションと新しいアプリケーションの両方をアンデプロイする方法

旧アプリケーションと新アプリケーションが共存する場合は、**-old**オプションまたは**-new**オプションを使用して、アンデプロイするアプリケーションを明示する必要があります。両方のアプリケーションをいずれもアンデプロイする場合は、以下のように**-all**オプションを指定して**undeploy**コマンドを実行します。

```
undeploy -force myservice_war
```

強制的に以前のアプリケーションをアンデプロイする方法

グレースフル再デプロイの代わりにノーマル再デプロイを行うには、**redeploy**コマンドを実行します。

```
redeploy myservice_war -path /path_to_new_application/myservice.war -force
```

2.2.4. Webアプリケーションのグレースフル再デプロイ

既存のWebアプリケーションで提供するサービスを停止せず、新しいWebコンテキストを再デプロイする機能です。

一般的な再デプロイは既存のWebコンテキストをアンデプロイするため、その過程でWebブラウザでエラーページが表示されることがあります。しかし、グレースフル再デプロイは、既存のWebコンテキストのサービスを利用しているユーザーが存在すると、アンデプロイを行わず、そのまま維持した状態で新しいWebコンテキストをデプロイします。一時的に既存のWebコンテキストと新しいWebコンテキストが同時に存在することになります。

既存のWebコンテキストのサービスを利用するユーザーを判別する基準は、現在進行中の要求やHTTPセッションの存在有無です。現在進行中の要求の場合、DBの遅延問題や長期間実行する非同期処理(Async Processing)でなければ、長時間かかる処理はないと考えられます。しかし、HTTPセッションの場合は最小限のアイドル・タイムアウトだけ残っているので、既存のWebコンテキストはその間はアンデプロイされません。再デプロイの実行者はこれらを踏まえて、既存のWebコンテキストに対するアンデプロイ・タイムアウトを指定でき、命令を通じて強制的にアンデプロイすることもできます。

新しいWebコンテキストのサービスが正常に開始すると、新しい要求は既存のWebコンテキストに渡されなくなります。既存のWebコンテキストのサービスを利用するユーザーはグレースフル再デプロイの終了前に接続したクライアントです。

参考

既存のWebコンテキストのサービスを利用するユーザーがHTTPセッションを使用する場合、そのセッションが期限切れになるまで新しいWebコンテキストを利用することができません。これを避けたい場合は、一般的な再デプロイを実行するか、グレースフル再デプロイを実行する際に、既存のWebコンテキストのアンデプロイ・タイムアウトなどを設定して、既存のWebコンテキストを強制的に削除してください。使用方法については、「[2.2.3. グレースフル再デプロイの使用方法](#)」を参照してください。

新しいWebアプリケーションの動作可否を事前に確認する方法

JEUSでは、新しいWebアプリケーションを配布のみ行い、正常に動作するか否かを確認する方法を提供します。新しいWebアプリケーションを配布のみ行う方法は、[2.2.3節「新しいアプリケーションを配布のみ行う方法」](#)を参照してください。

例で示したように、myservice.warのコンテキスト・パスが「/myservice」の場合、Webブラウザを使用して新しいWebアプリケーションを配布したサーバーのデフォルト・ポートにHTTP要求を送信します。新しいmyservice.warを配布のみ行ったサーバーのホスト名が「host1」で、デフォルト・ポートが「9736」と想定します。

```
http://host1:9736/myservice/
```

2.2.5. EJBアプリケーションのグレースフル再デプロイ

EJBアプリケーションのグレースフル再デプロイは、既存のEJB要求に対する処理を保証しつつ、新しいEJBを再デプロイする機能です。グレースフル再デプロイと同様、セッションBeanに対してのみ使用できます。したがって、JAR形式のアプリケーションをグレースフル再デプロイする場合、該当のJARはセッションBeanでのみ構成されている必要があります。

グレースフル再デプロイの時点を基準にして、新しいクライアントは新しいEJBをルックアップして使用し、既存のEJBを使用していたクライアントはユーザーが設定したタイムアウトの間は要求処理が保証されます。また、既存のEJBに対する要求処理がないと判断された場合は、タイムアウト内であっても既存のEJBをアンデプロイします。

要求処理に対する判断基準はEJBの種類によって異なります。ステートフル・セッションBeanの場合は、現在進行中の要求を基準にしており、ステートフル・セッションBeanの場合はセッションが存在するか否かで判断します。つまり、ステートフル・セッションBeanの場合はすべてのセッションが破壊されると、それ以上の追加要求がないと判断し、アンデプロイが行われます。

現在JEUSでは、性能向上のために、内部的にクライアントにホーム・スタブとEJBオブジェクト・スタブをキャッシュして使用しています。したがって、既存EJBの情報をキャッシュしているクライアントは、グレースフル再デプロイが完了した時点以降のキャッシュを削除し、新しいEJBの情報を再び取得する必要があります。正常動作のため、EJB 3.xは、別の設定なしで自動的に変更されますが、EJB 2.xの場合は<use-dynamic-proxy-for-ejb2>がtrueに設定されている必要があります。(<use-dynamic-proxy-for-ejb2> のデフォルト値はtrueです)

2.2.6. EARアプリケーションのグレースフル再デプロイ

EARアプリケーションの場合、EARに属しているWEB、EJBモジュールのグレースフル再デプロイを保証します。スタンドアロン・モジュールのグレースフル再デプロイと同様、基本的にWEBモジュールの場合はコンテキストのセッションが期限切れになるまで以前のサービスを保証します。また、以前のステートレス・セッションBeanの要求処理とステートフル・セッションBeanのセッションの期限切れまでの要求処理を保証します。

ただし、EARアプリケーションは1つのアプリケーションであるため、EARに属する1つのEJBモジュールが要求処理を完了しても、他のモジュールの要求を処理中であれば、EJBモジュールをアンデプロイできません。グレースフル再デプロイと同様、要求処理が完了していない以前のモジュールからEJBモジュールに要求することができるため、EARアプリケーションのすべての要求が正常に処理されることを保証するためには、1つのEJBモジュールまたはWEBモジュールのみアンデプロイすることは不可能です。EARアプリケーションをアンデプロイするには、EARアプリケーションに属するすべてのWEBモジュールまたはEJBモジュールの要求処理が完了する必要があります。

参考

旧アプリケーションの名前と新アプリケーションの名前が一致する必要があり、アプリケーションに属する各モジュールの名前も一致する必要があります。アプリケーション名の設定方法については、Java EE 7 Platform Specificationを参照してください。

第3章 アプリケーション

本章では、モジュールとアプリケーションの構成と、それぞれの構成要素について説明します。また、作成されたアプリケーションをデプロイするためにJEUSで提供する機能について説明します。

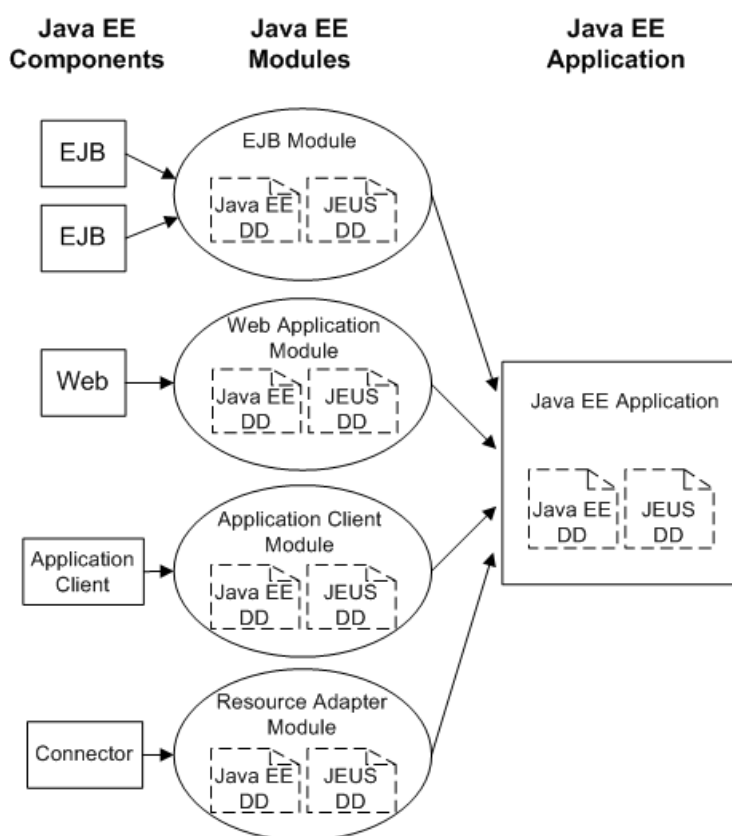
3.1. モジュールとアプリケーション

Java EEアプリケーションは、1つ以上のモジュールで構成されています。Java EEモジュールは1つ以上の同じタイプであるJava EEコンポーネント(EJB、Webアプリケーション、アプリケーション・クライアント、コネクタ)とDeployment Descriptor(以下、DD)から構成されます。

DDには、Java EEの標準DD(application.xmlなど)とJEUS DD(jeus-application-dd.xmlなど)があります。Java EE 5以降からは、標準DDに存在するほとんどの設定をアノテーションとしてクラスに明示できます。

以下は、Java EEモジュールおよびアプリケーションの構成です。

【図 3.1】 Java EEモジュールおよびアプリケーションの構成



3.1.1. モジュール

Java EEモジュールには以下の4つのタイプがあります。

- **EJBモジュール(.jar file)**

EJB(Enterprise JavaBeans)モジュールは、トランザクションおよびセキュリティー・サービスを利用するビジネス・ロジックを実装するための標準サーバー側のコンポーネント・モデルです。EJBモジュールは、EJBを実装してグループ化するための概念で、JEUSではJEUS EJBエンジンにデプロイできる最小単位です。

したがって、1つのEJBがデプロイする場合も、必ずEJBモジュールにパッケージ化(.jar file)する必要があります。EJBモジュールの詳細については、『JEUS EJBガイド』を参照してください。

- **Webモジュール(.war file)**

Webモジュールは、クライアントからの要求に対するWebベースのサービス(たとえば、ショッピングカートに商品を追加することや、ショッピングカートに入れた商品を購入すること、Webベースのオークションサイトをブラウジングすることなど)を実行するための静的コンテンツ(static content)と動的コンテンツ(dynamic content)の集まりです。

Webアプリケーション・モジュールの詳細については、『JEUS Webエンジンガイド』を参照してください。

- **アプリケーション・クライアント・モジュール(.jar file)**

アプリケーション・クライアント・モジュールは別のJVMで実行されるクライアント・プログラムです。アプリケーション・クライアント・モジュールはmain()メソッドを呼び出して実行し、仮想マシンが終了すると終了されます。

他のJava EEアプリケーション・コンポーネントと同様、アプリケーション・クライアント・モジュールはシステム・サービスを提供するクライアント・コンテナで動作します。クライアント・コンテナは他のJava EEコンテナに比べて、使用するシステム・リソースが非常に少ないです。

アプリケーション・クライアント・モジュールの詳細については、『JEUS アプリケーションクライアントガイド』を参照してください。

- **リソース・アダプター・モジュール(.rar file)**

リソース・アダプター・モジュールはJava EEコネクタ・アーキテクチャの中核コンポーネントとして、特定のEIS(Enterprise Information System)用に開発され、EISと相互作用するためのAPIを提供します。また、Java EEアプリケーション・サーバーと連携するためのシステムAPIも提供します。JEUS管理者は、必要に応じてリソース・アダプターを設定し、デプロイのみを行います。

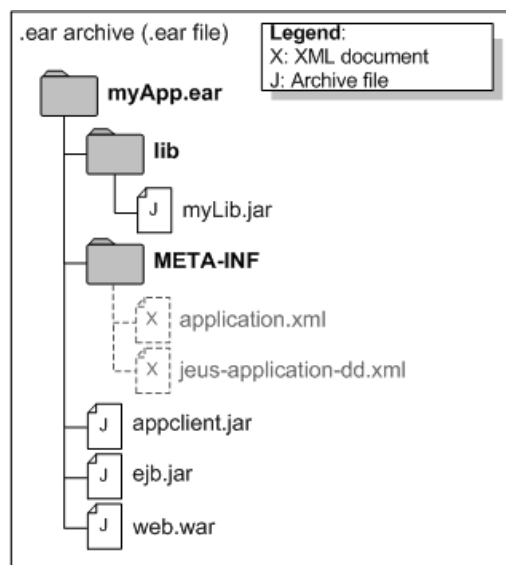
リソース・アダプター・モジュールの詳細については、『JEUS JCAガイド』を参照してください。

3.1.2. アプリケーション

[図 3.1]のJava EEアプリケーションは、1つ以上のJava EEモジュールと2つのDDから構成されています。DDはJava EE DD(application.xml)とJEUS DD(jeus-application-dd.xml)から構成されています。Java EEモジュールは1つ以上の同じタイプのコンポーネントとDDで構成されているため、1つのコンポーネントだけでもアプリケーションを作成することができます。

Java EEアプリケーションは拡張子が.earの一般的なJARアーカイブ形式のファイルです。以下は、EARファイルの構成の簡単な例です。EARファイルのルート・ディレクトリーには、アプリケーションに含まれている各モジュールのアーカイブ・ファイルとlibディレクトリー、META-INFディレクトリーが存在します。

[図 3.2] .ear archiveの構成



各ディレクトリーについての説明です。

lib

libディレクトリーはEARアプリケーションの共通ライブラリーのディレクトリーです。ディレクトリーの下位には、ライブラリー.jarファイルが格納されます。このように格納されたライブラリーは、EARアプリケーションの下位モジュールで自動的にクラス・パスに追加されます。このディレクトリーは、設定によって、他のディレクトリーに代替することもあります。

Java EE 5からは、**ライブラリー・ディレクトリー**をサポートしており、EARアプリケーションの標準DDのapplication.xmlに<library-directory>タグで設定して使用できます。application.xmlファイルがない場合やapplication.xmlに<library-directory>を設定していない場合は、基本的にlibディレクトリーが使用されます。

参考

JEUS 5から提供しているAPP-INFディレクトリーは共通ライブラリーのディレクトリーであり、Webアプリケーション・モジュールのWEB-INFディレクトリーと類似した機能を提供します。WEB-INFと同様、APP-INFも下位にclassesとlibディレクトリーを持っており、各クラス・ファイルと.jarファイルが格納されます。

APP-INFの下に存在するclassesディレクトリーのクラスとlibディレクトリーの.jarファイルは、以下のアプリケーションでライブラリーとして使用できます。ただし、ライブラリー・ディレクトリーを使用するのが標準であるため、JEUS 6以降からは、APP-INFディレクトリーの代わりにライブラリー・ディレクトリーを使用することを推奨します。

META-INF

META-INFディレクトリーには、Java EE DDのapplication.xmlとJEUS DDのjeus-application-dd.xmlの2つのデプロイメント記述子が存在します。このファイルはなくても構いません。このディレクトリーには、DDのほか、MANIFEST.MFというファイルが存在しています。アーカイブされたEARファイルの情報やクラスパスを明示できます。

3.2. デプロイメント記述子(DD)

Java EEアプリケーションはEAR(Enterprise ARchive .earファイル)形式でデプロイされます。

このファイルには、EJBモジュール(EJB .jarファイル)、Webアプリケーション・モジュール(.warファイル)、リソース・アダプター・モジュール(.rarファイル)、クライアント・モジュール(クライアント.jarファイル)、そしてその他必要なJavaクラスが含まれています。また、1つのモジュールで構成されているスタンドアロン・モジュールもJava EEアプリケーションの一種です。これらのアプリケーション・サービスを開始するために、JEUSにモジュール・ファイルをアップロードして制御するすべての動作を **デプロイメント**といいます。

アプリケーション・サーバーにデプロイするためのモジュールまたはアプリケーションを作成するためには、DDが必要です。DDにはJava EEの標準DDとJEUS DDの2つがあります。

参考

標準DDで使用可能な設定はアノテーションとして実装可能になったため、EJBモジュールとEARファイルの場合、Java EE 5に準拠したJEUS 6から標準DDとJEUS DDを省略可能です。Webアプリケーション・モジュールの場合、Java EE 6に準拠したJEUS 7から標準DDとJEUS DDを省略可能です。

各モジュールとアプリケーションに必要なDDは以下のとおりです。

[表 3.1] モジュール別のDD

	Java EEの標準DD	JEUS DD
アプリケーション	application.xml	jeus-application-dd.xml
EJBモジュール	ejb-jar.xml	jeus-ejb-dd.xml

	Java EEの標準DD	JEUS DD
Webアプリケーション・モジュール	web.xml	jeus-web-dd.xml
アプリケーション・クライアント・モジュール	application-client.xml	jeus-client-dd.xml
リソース・アダプター・モジュール	ra.xml	jeus-connector-dd.xml
Webサービス	webservices.xml	jeus-webservices-dd.xml

モジュールごとにJava EEの標準DDのほか、JEUS用のDDを持っているのと同様、アプリケーション記述子においても、それに対応するjeus-application-dd.xmlがEARのMETA-INFに格納されています。このファイルは、アプリケーションの詳細設定を行うために使用します。このファイルがEARまたはスタンドアロン・モジュールに含まれていない場合は、デフォルト設定でデプロイされます。

参考

Java EEの標準DDについての詳細は、Java EE 7の仕様を参照してください。また、各モジュールに対応するJEUS DDについては、該当するJEUSのガイドを参照してください。

jeus-application-dd.xmlには、アプリケーションがサービスされるための詳細情報を設定できます。サーバーのデプロイでは、アプリケーションをデプロイする際、該当のファイルに設定されている情報を読み込んでアプリケーションをサービスするための環境を構成します。

以下は、<application>タグの設定項目についての説明です。

● セキュリティー設定

アプリケーションに適用されるセキュリティ関連の設定は以下のとおりです。項目の詳細については、『JEUS セキュリティガイド』を参照してください。

Element	説明
<role-permission>	アプリケーションで使用するprincipal-roleマッピングを指定します。この設定はアプリケーションの下位のすべてのモジュールに適用されます
<java-security-permission>	J2SEセキュリティ・マネージャーを使用する場合、アプリケーションに割り当てるパーミッションを指定します

● ライブラリーの設定

アプリケーションで共有ライブラリーを使用する場合、以下の設定でライブラリーを指定できます。項目の詳細については、「[3.3.2. 共有ライブラリー](#)」を参照してください。

Element	説明
<library-ref>	アプリケーションで使用する共有ライブラリーを指定します

- Java EEでの名前空間に関する設定

アプリケーションで使用する名前空間に関する情報はapplication.xmlまたはアノテーションに明示できます。それに基づくマッピング情報はjeus-application-dd.xmlに記録できます。この情報はアノテーションですべて代替可能です。ネーミング環境の詳細については、Java EE 7の仕様を参照してください。この設定の詳細については、『JEUS XMLリファレンスガイド』の「第23章 jeus-application-dd.xmlの設定」を参照してください。

- クラスローディングの設定

EARアプリケーションに含まれているライブラリーをロードする順序を設定します。基本的にJavaのクラス・ローダーによって親クラス・ローダーからクラス・ロードを試みます。ただし、アプリケーション設定によって親クラス・ローダーより前に、アプリケーションが持っているライブラリーをロードする必要がある場合があります。このような場合は、設定を通じてクラス・ロードの順序を変更することができます。

Element	説明
<classloading>	EARアプリケーションの内部ライブラリーをロードする方式を設定します。 trueに設定すると、EAR内に含まれているライブラリーを先にロードします。ライブラリーが存在しない場合は、親クラス・ローダーのクラス・ローダーを利用してクラス・ロードを行います

3.3. アプリケーションでのライブラリーの使用

本節では、複数のアプリケーションで利用できるアプリケーション・ライブラリー(lib/application)と共有ライブラリーを追加および使用する機能について説明します。また、JEUS 8から新しく追加されたライブラリー・デプロイメント機能について説明します。

3.3.1. lib/applicationディレクトリー

アプリケーション・ライブラリー(lib/application)はアプリケーション同士が共有するライブラリーであり、JEUSシステムに追加されるシステム・ライブラリー(JEUS_HOME/lib/system)とは区別されます。

システム・ライブラリーはシステムで使用するライブラリーであり、lib/applicationに存在するライブラリーは、アプリケーションまたはユーザーが設定したクラス(サーバー・ライフサイクルの呼び出し機能、ユーザー・ロガー、ロガーのフィルター、ロガーのフォーマットなど)で使用するライブラリーです。

lib/applicationにライブラリーを格納すると、ライブラリー・ファイルをアプリケーション同士で共有することができます。したがって、ユーザーがアプリケーション・ライブラリーを毎度一緒にパッケージングする必要はありません。主にドメインやサーバーに存在するすべてのアプリケーションが共通で使用するライブラリーが格納されます。ライブラリー形式のJARファイルが格納されることもあります。

注

lib/applicationディレクトリーに同じライブラリーを複数のバージョンでインストールした場合、アプリケーションが必要とするライブラリーが選択されない場合がありますので、アプリケーションが使用するライブラリーのバージョンを明示できる共有ライブラリーを使用してください。共有ライブラリーの詳細については、「[3.3.2. 共有ライブラリー](#)」を参照してください。

lib/applicationディレクトリーはDOMAIN_HOMEとSERVER_HOMEに配置できます。

DOMAIN_HOME/lib/applicationには、ドメイン全体で共有できるライブラリーを格納します。ドメインにインストールされたアプリケーション・ライブラリーを他のマシンにあるサーバーに渡したい場合、他のマシンで初めてドメインを構成する際、pack-domainコマンドを使用すると、DOMAIN_HOME/lib/applicationも一緒にコピーされます。以降のアプリケーション・ライブラリーの追加または変更はユーザーが手動で同期化する必要があります。

特定のサーバーでのみ使用するアプリケーション・ライブラリーをインストールしたい場合は、使用するライブラリーをSERVER_HOME/lib/applicationに格納します。これはコマンドで提供していないので手動で格納してください。

参考

SERVER_HOME/lib/applicationディレクトリーはJEUSのインストール時には作成されませんが、必要に応じて直接作成してライブラリーを格納することができます。

クラス・ローディング方式

アプリケーション・ライブラリー(lib/application)のクラスはJEUSルート・クラスローダーによってローディングされます。したがって、サーバーにデプロイされるすべてのアプリケーションで使用できます。

システム・ライブラリー(JEUS_HOME/lib/system)と同じクラスローダーによってローディングされますが、システム・ライブラリーはユーザーがアクセスしてはならないディレクトリーに存在する一方、アプリケーション・ライブラリーはユーザーが参照するためのライブラリーが格納されるため、両ディレクトリーの用途は異なります。

参考

サーバーのクラス・ローディング方式の詳細については、『*JEUS サーバガイド*』の「1.4. クラス・ローダーの構造」を参照してください。

アプリケーション・ライブラリーはSERVER_HOMEとDOMAIN_HOMEに存在します。

クラスローダーでクラスパスに追加する順序は、SERVER_HOME/lib/applicationが先でDOMAIN_HOME/lib/applicationが後になります。したがって、SERVER_HOMEとDOMAIN_HOMEに同じライブラリーが存在する場合は、SERVER_HOMEにインストールされているライブラリーがローディングされるため、DOMAIN_HOMEに存在するライブラリーは無視されることがあります。

クラスローダーにライブラリーをクラスパスとして追加する際、同じライブラリー・ファイル名が使われている場合は、サーバーの起動時に以下のような警告メッセージを出力します。

```
Warning [same classpath-name] :  
[JEUS_HOME/domains/domain1/servers/adminServer/lib/application/applib.jar] is  
registered earlier than [JEUS_HOME/domains/domain1/lib/application/applib.jar] in  
JEUSClassLoader.
```

参考

上記のログ・メッセージは、JEUSルート・クラスローダーにlib/applicationをクラスパスとして追加する際に発生します。これはサーバーの起動時にクラスローダーを構成する段階で行われる作業であり、サーバーの起動プロセスの最初の段階に該当します。サーバー・ロガーが設定される前なので、このメッセージはランチャー・ログから確認する必要があります。

アプリケーション・ライブラリーは、アプリケーションだけでなく、サーバーに特定のクラスを設定して使用できる場合に使用可能です。例として、サーバー・ライフサイクルの呼び出し機能とユーザー・ロガー、ログ・メッセージのフィルタークラス、ログ・メッセージのフォーマット・クラスなどがあります。これらのクラスでは、アプリケーション・ライブラリーのライブラリーを参照しないため、該当のクラスがlib/applicationに格納される必要があります。

3.3.2. 共有ライブラリー

共有ライブラリーはアプリケーション間で共有されるライブラリーであり、JEUSシステム・ライブラリー(JEUS_HOME/lib/system)とアプリケーション・ライブラリー(DOMAIN_HOME/lib/application、SERVER_HOME/lib/application)とは区別されます。

共有ライブラリーは、JEUSのシステム全体に影響を与えず、アプリケーションごとにライブラリーの使用有無を設定することができます。また、JEUSを再起動せずに動的に追加することや、同じライブラリーを複数のバージョンでインストールして選択的に使用することができます。

以下は、共有ライブラリーの特性です。

- アプリケーション同士がライブラリー・ファイルを共有できるため、ユーザーが毎度一緒にパッケージングする必要がありません。
- サーバーが運用中の場合もライブラリーを動的に追加および削除することができます。

- 新しいバージョンでライブラリーを追加し、アプリケーションを再デプロイすると、アップグレードされたライブラリーを使用することができます。
- ライブラリーの複数バージョンのインプリメンテーションを登録でき、使用するインプリメンテーションはデプロイ時に指定できます。

各ライブラリーは2つのバージョン(仕様と実装)を持つことができます。これにより、ユーザーは同じライブラリーの複数のバージョンをインストールすることができ、デプロイ時にアプリケーションが必要なバージョン(highest、minimum、exact)を動的に選択できます。

複数のバージョンのライブラリーを使用するため、常に最初からバージョンを明示することを推奨します。ただし、単純なユースケースであれば、バージョンを明示しなくても構いません。バージョンを明示しない場合は、基本的にバージョン値を0と見なします。

ライブラリーのインストールと設定

1つのライブラリーは複数のJARファイルから構成されますが、一般的に共有ライブラリー・ディレクトリーの `JEUS_HOME/lib/shared` の下位に格納されます。以下のようにJARファイルを `JEUS_HOME/lib/shared/libraries.xml` 設定ファイルにライブラリーとして登録します。

以下の例で、「myLibrary」は2.0バージョンの仕様を実装する2.1バージョンのインプリメンテーションで定義して登録されています。このライブラリーは、複数のJARファイル(`commons-logging.jar`、`commons-util.jar`、`myLib-2.1`サブ・ディレクトリーに存在するすべてのJARファイル)で構成されています。

[例 3.1] <<共有ライブラリーの登録 : libraries.xml>>

```
<library>
  <library-name>myLibrary</library-name>
  <specification-version>2.0</specification-version>
  <implementation-version>2.1</implementation-version>
  <files dir=".">
    <include name="commons-logging.jar" />
    <include name="commons-util.jar" />
  </files>
  <files dir="myLib-2.1" />
</library>
```

以下は、設定タグについての説明です。

- `<library-name>`, `<specification-version>`, `<implementation-version>`
 - アプリケーションがライブラリーを参照する際に使用します。
 - `<*-version>`フィールドは、同じ名前のライブラリーを複数のバージョンで管理する場合に使用します。

- <files>

実際のライブラリーのクラス・パスを設定します。<files>タグは以下の例のように様々な方法でクラス・パスを設定することができます。

[例 3.2] <<共有ライブラリーの<files>タグ>>

```
<files dir=".">
  <include name="a.jar" />
  <include name="b.jar" />
</files>

<files dir="testa" />

<files dir="/home/works/lib/testc" />

<files dir="/home/works/lib/testd" mode="classes" />
```

- dirの値はJARファイルを格納したディレクトリーまたはclassesディレクトリーを指定でき、相対パスと絶対パスのいずれも使用できます。相対パスを使用する場合、ベース・ディレクトリーとして共有ライブラリー・ディレクトリーであるJEUS_HOME/lib/sharedの相対パスとして解釈されます。
- <include>サブ・タグを用いて、含めるJARファイルを指定できます。<include>タグを設定していない場合は、該当するディレクトリーのすべてのJARファイルが指定されます。この場合、当該ディレクトリーはデプロイ時にJARファイルを動的に検索するので、以降設定を変更することなくJARファイルをディレクトリーに追加することができます。JARファイル・ディレクトリーではなく、クラス・ディレクトリーを指定した場合は、mode値を「classes」と指定します。
- 新しいアプリケーションのデプロイ時にこの設定が変更された場合は、再び読み込んで処理するため、JEUSが動作中にもライブラリーを動的に追加できます。すなわち、JEUSを再起動せずに、新しいライブラリーまたはアップグレードされたバージョンのライブラリーを追加することができます。

注

上記の設定は該当するXMLを直接変更する必要があります。JEUS 6ではWebAdminを使用して変更することが可能でしたが、JEUS 7以降のバージョンからはサポートしません。

アプリケーションでのライブラリーの参照

Java EEアプリケーションまたはスタンドアロン・モジュールは、jeus-application-dd.xml、jeus-web-dd.xml、jeus-ejb-dd.xmlのエントリーを通じて登録された共有ライブラリーを使用することができます。

以下は、共有ライブラリーの「myLibrary」を参照する例です。

```
<library-ref>
  <library-name>myLibrary</library-name>
</library-ref>
```

上記の例で、デプロイ時にアプリケーションは「myLibrary」という名前のライブラリーを検索し、該当するクラス・パスをアプリケーションのクラス・パスに追加します。複数バージョンの「myLibrary」が存在する場合は、最上位バージョンを選択します。参照されるライブラリーのバージョンが設定されていない場合は、[3.3.2節「バージョンの付け方」](#)に基づいて常に最上位バージョンを選択します。

以下は、最も古いバージョンが必要な場合の例です。

```
<library-ref>
  <library-name>myLibrary</library-name>
  <specification-version>2.0</specification-version>
  <implementation-version>2.0</implementation-version>
</library-ref>
```

以下は、正確なバージョンを必要とする場合の例です。

```
<library-ref>
  <library-name>myLibrary</library-name>
  <specification-version exact-match="true">2.0</specification-version>
  <implementation-version exact-match="true">2.1</implementation-version>
</library-ref>
```

以下は、決められた仕様バージョンのみを設定したい場合の例です。このようにすると、該当する仕様バージョンの最上位実装バージョンを選択することになります。

```
<library-ref>
  <library-name>myLibrary</library-name>
  <specification-version exact-match="true">2.0</specification-version>
</library-ref>
```

アプリケーションのデプロイ時に参照ライブラリーが見つからない場合は、デフォルトでWARNINGログが出力されますが、デプロイは続行されます。このような動作を望まない場合は、以下のように「failonerror」属性を設定してデプロイを失敗させることができます。

```
<library-ref failonerror="true">
  <library-name>myLibrary</library-name>
</library-ref>
```

クラス・ローディング方式

共有ライブラリーのクラスは、ライブラリーの参照を定義している場所に応じて、アプリケーション・クラスローダーまたはモジュール・クラスローダーによってローディングされます。

たとえば、「lib1」がjeus-application-dd.xmlで参照として定義されている場合は、EARレベルのアプリケーション・クラスローダーによってローディングされます。一方、jeus-web-dd.xmlで参照として定義されている場合は、Webレベルのクラスローダーによってローディングされます。

各アプリケーションのクラスローダーによってローディングされたクラスの場合、該当のアプリケーションにのみ制限されます。これはライブラリーの場合も同様です。したがって、クラス・インスタンスはアプリケーション間で共有されません。

参考

サーバーのクラス・ローディング方式の詳細については、『JEUS サーバガイド』の「1.4. クラス・ローダーの構造」を参照してください。

バージョンの付け方

バージョンは、「6.2.3-b12」のように<fraction_part>(6.2.3)と<string(non-fraction)_part>(-b12)から構成されます。

```
Version ::= <fraction_part> | <string_part> | <fraction_part> <string_part>
fraction_part ::= <integer> | <integer> "." <fraction_part>
string_part ::= <non-numeric> <character>*
```

バージョンを付ける規則は、以下のとおりです。

- <fraction part>をメジャー、マイナーの順に数値的に比較します。
- <fraction part>が同じの場合は<string part>を比較します。この場合、文字列の比較方式に従います。

以下は、規則に基づいたバージョンの付け方の例です。

```
6.0 < 6.2.3 < 6.2.3-b12 < 6.2.3-beta < 6.2.4
```

3.3.3. ライブラリーのデプロイメント

JEUS 8からは、DASを利用してライブラリーを管理できるよう、ライブラリー・デプロイメント機能を提供します。同機能は、ライブラリーをコンソール・ツールまたはWebAdminを用いてドメインにデプロイした後、設定を通じてアプリケーションが該当するライブラリーを参照できるようにします。これにより、複数のアプリケーションが異なるバージョンの同じライブラリーを参照するときに発生し得るクラスの競合を最小限に抑えることができます。

以下は、ライブラリー・デプロイメント機能の長所です。

- 使用するライブラリーをアプリケーションごとにパッケージングする必要がありません。複数のアプリケーションが同じライブラリーを使用する際、アプリケーションごとにパッケージングした場合に発生する利便性の問題や重複クラス・ロードによるリソース使用の問題を解決することができます。

- アプリケーションをデプロイする際、使用するライブラリーを指定することができます。同じライブラリーが複数デプロイされている場合は、バージョンを指定して使用するライブラリーを特定することができます。

主な機能

アプリケーションでライブラリーを使用するためには、インストールおよびデプロイが必要です。以下は、ライブラリーをデプロイするために提供する主な機能です。

1. ライブラリーのインストール

ライブラリーをデプロイするためには、まず、該当するライブラリーをDASが存在するマシンのドメイン・ディレクトリーにインストールします。インストールはコンソール・ツールまたはWebAdminを用いて行います。インストールに必要な情報は以下のとおりです。

区分	説明
ライブラリーの識別子(ID)	デプロイ、削除および参照する際に使用するライブラリーの名前です。ドメイン内で一意である必要があり、指定しないとインストールできません
ライブラリーのバージョン	インストールするライブラリーのバージョンを指定できます。指定しない場合は、1.0と見なします
ライブラリーのパス	ライブラリーがインストールされるパスを指定します。指定しないとインストールできません

インストールが完了すると、ライブラリー・ファイルはDOMAIN_HOME/.libraries/LIBRARY_ID/VERSION下に格納されます。

参考

インストールしたライブラリーの情報は、domain.xmlに記録されません。DASがライブラリーのインストール状態を把握するときは、DOMAIN_HOME/.libraries下のディレクトリー構造を探索するので、任意でディレクトリーを変更しないようにしてください。

2. ライブラリーのデプロイ

インストールしたライブラリーは、コンソール・ツールまたはWebAdminを用いてデプロイすることができます。デプロイ時に該当のライブラリーを使用するサーバーやクラスターを指定するか、あるいはすべてのサーバーを対象とすることもできます。デプロイが正常に行われると、DASはその内容をdomain.xmlに記録するので、以降の再起動時にもライブラリーのインストール状態を保つことができます。

3. ライブラリーのアンデプロイ

ライブラリーが不要になったら、コンソール・ツールまたはWebAdminを用いて該当するライブラリーをアンデプロイすることができます。アンデプロイが完了すると、DASはアンデプロイされたライブラリーの情報をdomain.xmlから削除しますが、ライブラリー・ファイルは削除せずに保持します。

4. ライブラリーのアンインストール

コンソール・ツールまたはWebAdminを用いてライブラリー・ファイルを削除することができます。

3.3.3.1. ライブラリーのインストールとアンインストール

本節では、WebAdminとコンソール・ツールを用いてライブラリーをインストールおよびアンインストールする方法について説明します。

WebAdminの使用

WebAdminのメイン画面で[Applications]メニューを選択すると、アプリケーションとライブラリーを管理できる画面に移動します。この画面にて、ライブラリーのインストールとアンインストールを行います。

[図 3.3] ライブラリーの管理



コンソール・ツールの使用

以下は、コンソール・ツール(jeusadmin)を使用してライブラリーをインストールおよびアンインストールする方法です。

- インストールされているライブラリーの情報を確認

library-infoコマンドを使用して、インストールおよびデプロイされているライブラリーの一覧を確認することができます。

```
[DAS]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+
(No data available)
=====
[DAS]domain1.adminServer>
```

- ライブラリーのインストール

install-libraryコマンドを使用してライブラリーをインストールできます。ライブラリーの識別子(ID)とパスを指定しないとインストールできないので必ず入力してください。

```
[DAS]domain1.adminServer>install-library log4j -path
/usr/lib/apache-log4j-1.2.17/log4j-1.2.17.jar -version 1.2.17
Successfully installed the library [log4j] version [1.2.17].
[DAS]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target | Target Clusters | Applicatio |
|           |        |      | Servers|                  | ns         |
+-----+-----+-----+-----+-----+-----+-----+
| log4j     | 1.2.17 | INSTALLED|        |                  |           |
+-----+-----+-----+-----+-----+-----+-----+
=====
[DAS]domain1.adminServer>
```

● ライブラリーのデプロイ

deploy-libraryコマンドを使用してライブラリーをデプロイします。デプロイ時にライブラリーを使用する対象(サーバー、クラスター、あるいはすべてのサーバー)を指定します。

```
[DAS]domain1.adminServer>deploy-library log4j -servers adminServer
deploy the library [log4j] succeeded.
[DAS]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+-----+
| log4j     | 1.2.17 | RUNNING| adminServer   |                  |             |
+-----+-----+-----+-----+-----+-----+-----+
=====
[DAS]domain1.adminServer>
```

● ライブラリーのアンデプロイ

デプロイしたライブラリーが不要になった場合、undeploy-libraryコマンドを使用して該当するライブラリーをアンデプロイすることができます。不要になったライブラリーをアンデプロイしなくてもサーバーの機能には問題ありませんが、サーバーの起動時にライブラリーを同期化するなど、不要な操作が発生する可能性があります。

```
[DAS]domain1.adminServer>undeploy-library log4j
undeploy the library [log4j] succeeded.
[DAS]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target | Target Clusters | Applicatio |
|           |        |      | Servers|                  | ns         |
+-----+-----+-----+-----+-----+-----+-----+
=====
```

```
| log4j | 1.2.17 | INSTALLED | | |
+-----+-----+-----+-----+-----+-----+
=====
[DAS]domain1.adminServer>
```

● ライブラリーのアンインストール

uninstall-libraryコマンドを使用して、ドメインからライブラリーを完全に削除することができます。デプロイ済みのライブラリーは削除できないので、アンデプロイを行ってから削除してください。

```
[DAS]domain1.adminServer>uninstall-library log4j
uninstall the library [log4j] succeeded. : Successfully deleted [log4j].
[DAS]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+
(No data available)
=====
[DAS]domain1.adminServer>
```

3.3.3.2. アプリケーションのライブラリー参照


アプリケーションのデプロイ時に参照するライブラリーを指定することができます。コンソール・ツールまたはWebAdminを用いてデプロイを行うとき、参照するライブラリーのIDとバージョンを指定することができます。バージョンを指定しない場合は、最新バージョンのライブラリーが使用されます。

JEUSはデプロイ時に指定した情報を使用して、アプリケーションが該当するライブラリーを参照できるようにします。アプリケーションが参照するライブラリーの情報は、他のデプロイ情報と一緒にdomain.xmlに記録されます。


WebAdminの使用

WebAdminを使用してデプロイする際、デプロイ関連の設定を入力するポップアップウィンドウで使用するライブラリーを指定することができます。

[図 3.4] 依存ライブラリーの設定


Dependent Libraries

アプリケーションが使用するライブラリを定義します。各ライブラリはアプリケーションより先にデプロイされている必要があります。この値は、デプロイ作業の終了と同時にDASのxmlに書き込まれます。ユーザが任意で変更してはなりません。


Dependent Library

アプリケーションが使用するライブラリを定義します。デプロイされたライブラリから選択します。

ID	Version
log4j	<input type="checkbox"/> 1.2.17

コンソール・ツールの使用

deploy-applicationコマンドを使用するときに参照するライブラリーのIDとバージョンを指定します。

```
[DAS]domain1.adminServer>deploy-application sample -lib log4j -version 1.2.17  
-servers adminServer  
deploy the application for the application [sample] succeeded.  
[DAS]domain1.adminServer>library-info  
Library information  
=====
```

Library ID	Version	State	Target Servers	Target Clusters	Applications
log4j	1.2.17	RUNNING	adminServer		sample

```
=====
```

[DAS]domain1.adminServer>

第4章 アプリケーションの作成およびデプロイ

本章では、Java EEアプリケーション・ファイル(EAR)を作成し、これをJEUSにデプロイする方法について説明します。また、JEUSが提供するアプリケーションをデプロイできるツールを使用してJEUSサーバーにアプリケーションをデプロイする方法と、その他の操作について説明します。

4.1. アプリケーションの作成

本節では、作成された各モジュールを含むアプリケーションの作成方法の中で、jarユーティリティを使用してJava EEアプリケーションを直接作成する方法について説明します。

EARファイルを作成する前に、まずEARファイルに含まれるモジュールを作成する必要があります。EARファイルに含まれるモジュールには、EJBモジュールのJARファイル、Webアプリケーション・モジュールのWARファイル、リソース・アダプタ・モジュールのRARファイルなどがあります。モジュール・ファイルの作成についての詳細は、該当するガイドを参照してください。

以下は、アプリケーションの作成環境についての説明です。

- myApp.earというEARファイルを作成します。このファイルには、ejb.jarというEJBモジュールと、web.warというWebアプリケーション・モジュール、appclient.jarのアプリケーション・クライアント・モジュールが含まれています。
- アプリケーションmyApp.earをドメインのDASである「adminServer」にインストールしてから、サーバー「server1」にデプロイします。

以下の手順でアプリケーションを作成します。

1. EARファイルに含まれる、JAR、WAR、RARファイルを作成します。
2. JAR、WAR、RARファイルと同じディレクトリーでMETA-INFディレクトリーを作成します。
3. application.xmlファイルを作成(EARファイルのモジュールを含む)し、META-INFディレクトリーにコピーします。

以下はapplication.xmlの例です。

```
<?xml version="1.0" encoding="UTF-8"?>
<application version="7"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/application_7.xsd">
<description>Application description</description>
<display-name>Sample application</display-name>
<module>
    <ejb>ejb.jar</ejb>
</module>
<module>
    <web>
        <web-uri>web.war</web-uri>
        <context-root>hello</context-root>
    </web>
</module>
<module>
    <java>appclient.jar</java>
</module>
</application>

```

4. 以下のように、jarユーティリティを使用してコマンドを実行すると、myApp.earファイルが作成されます。

```
> jar cf myApp.ear ejb.jar web.war appclient.jar META-INF
```

注

META-INFディレクトリは必ずアルファベットの大文字で作成してください。小文字で作成すると問題が発生します。

4.2. デプロイ

コンソール・ツールとWebAdminで提供するデプロイ関連の機能により、以下のような操作を実行することができます。

ツール	説明
配布	アプリケーション・ファイルを対象サーバーやクラスターにコピーし、アプリケーションをサービスするための事前準備を行います。対象として指定したサーバーやクラスターのうち1つでも配布に失敗したら全体の失敗と見なし、配布に成功したサーバーからすべてアンデプロイします
デプロイ	アプリケーション・ファイルを対象サーバーにコピーし、アプリケーションをサービスするための事前準備を行います。この作業が完了したら、アプリケーションを実行します。対象として指定したサーバーやクラスターのうち

ツール	説明
	<p>1つでも配布に失敗したら全体の失敗と見なし、配布に成功したサーバーからすべてアンデプロイします。</p> <p>開始に失敗したサーバーがあっても、1つでもサービス可能な状態であればアプリケーションの開始は成功となります</p>
開始	対象サーバーに配布されたアプリケーションを動作させます。開始に失敗したサーバーがあっても、1つでもサービス可能な状態であればアプリケーションの開始は成功となります
停止	対象サーバーにデプロイされて実行中のアプリケーションを一時停止します。この際、アプリケーションはサーバーから削除しません。また、アプリケーションの名前で開始または再デプロイすることもできます
アンデプロイ	デプロイされて実行中のアプリケーションを停止し、デプロイされている対象サーバーやクラスターからアプリケーションを削除する作業です
再デプロイ	<p>デプロイされて実行中のアプリケーションの設定が変更された場合、変更内容を現在のアプリケーションに反映して再デプロイします。</p> <p>すべてのアプリケーションを再デプロイする場合、1つでも失敗したデプロイがあると、すべてのアプリケーションが停止します</p>

以下は、JEUSでのみ提供するデプロイです。コンソール・ツールまたはWebAdminを使用して実行します。

操作	説明
アプリケーションのサービス対象の追加	<p>デプロイまたは配布されたアプリケーションのサービス対象に特定のサーバーまたはクラスターを追加します。既にサービスされているアプリケーションのサービス対象を拡張する際に行います。</p> <p>複数のサーバーまたはクラスターをサービス対象に設定できます。サービス対象と指定したサーバーやクラスターでこの作業が失敗した場合は作業全体の失敗と見なし、既にサービス中のサーバーを除き、新しく追加するサーバーからはすべてアンデプロイされます</p>
アプリケーションのサービス対象の削除	<p>デプロイまたは配布されたアプリケーションのサービス対象から特定のサーバーまたはクラスターを削除します。既にサービスされているアプリケーションのサービス対象を減らす際に行います。</p> <p>複数のサーバーまたはクラスターをサービス対象に設定できます</p>

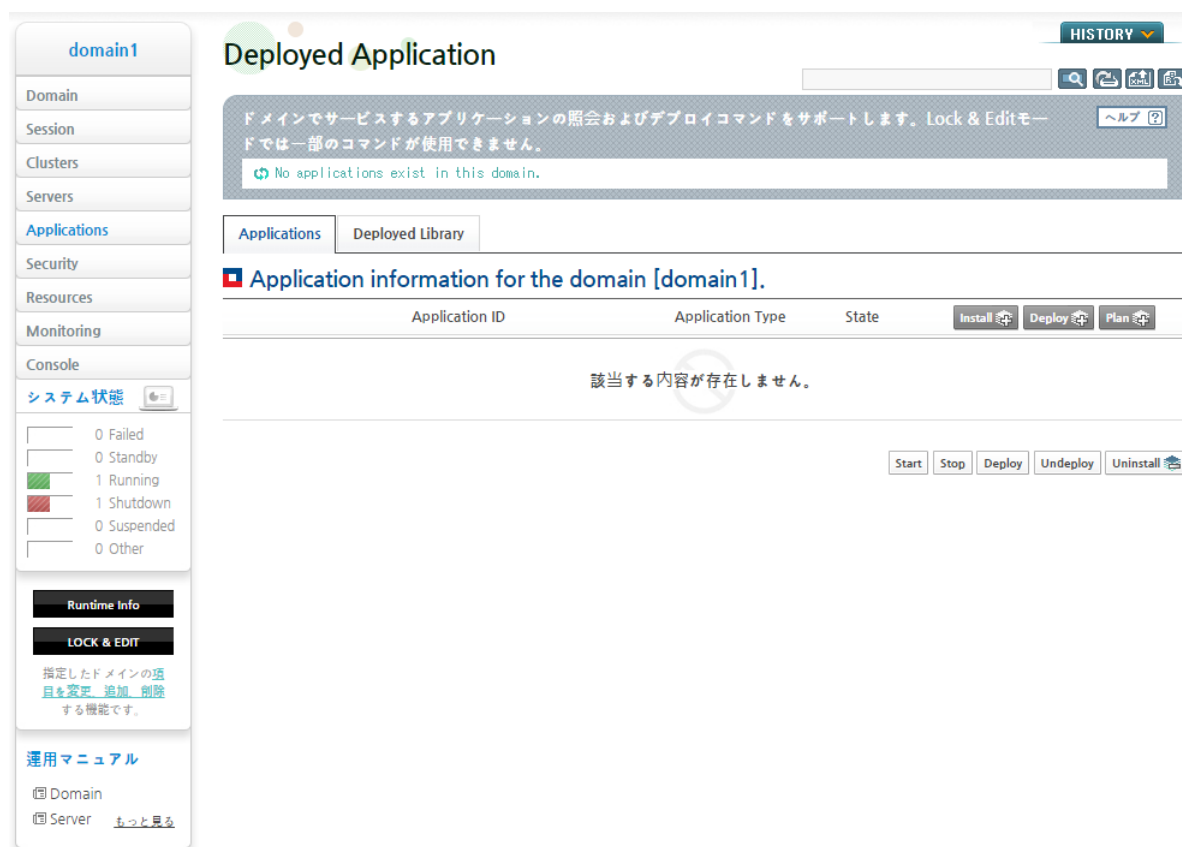
4.3. アプリケーションの制御およびモニタリング

本節では、WebAdminとコンソール・ツールを使用したアプリケーションの制御とモニタリング方法について説明します。

WebAdminは、JEUSを全般的に管理するために提供されるWebベースの管理ツールです。

WebAdminの左側のメニューで**[Applications]**を選択すると、ドメインに存在するアプリケーションを管理できる**[Deployed Application]**画面に移動します。WebAdminでアプリケーションをデプロイしたり、新しいアプリケーションをドメインにインストールするとき、あるいはドメインに存在するアプリケーションの情報を照会するときなど、すべてこの画面から開始します。

[図 4.1] WebAdminの[Deployed Application]画面



次に、WebAdminを使用してmyApp.earアプリケーションをドメインにインストール、デプロイ、および照会する方法について説明します。

注

アプリケーションの制御(デプロイ、配布、アンデプロイ、開始、停止)は、設定変更ロックがドメインに適用されていない状態でのみ可能です。**[LOCK & EDIT]**ボタンでロックを適用した状態の場合、**[Activate Changes]**ボタンを選択して変更中のすべての設定を反映するか、**[Undo All Changes]**ボタンを選択して変更した設定をキャンセルしてから、アプリケーションの制御を行います。

4.3.1. アプリケーションをドメインにインストール

以下は、アプリケーションをドメインにインストールする手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。同画面にてドメインに存在するアプリケーションの情報を確認することができます。

アプリケーション一覧の上段にある**[install]**ボタンをクリックすると、ドメインに新しいアプリケーションをインストールできる画面に移動します。

2. **[install]**ボタンをクリックすると表示される画面で、アプリケーションのIDとアプリケーション・ファイルのパスなど、インストール関連の情報を入力して**[確認]**ボタンをクリックします。

以下は、設定項目についての説明です。

項目	説明
Id	インストールするアプリケーションに付与するIDを指定します。この値を指定していない場合は、アプリケーション・ファイル名を変更してIDとして使用します。アプリケーションIDは入力しなくても作成されますが、管理の便宜上入力することを推奨します
Path	インストール時のアプリケーション・ファイルのパスを入力します。 [ファイル選択] ボタンでアプリケーション・ファイルを選択できます
Force	同じIDを持つアプリケーションが既にドメインに存在する場合は、既存のアプリケーション・ファイルを上書きします。デプロイ済みのアプリケーション内容が変更され、ファイルをアップデートする必要がある場合に使用できます
Upgrade	JEUS 6以下のバージョンのJEUS DDを使用する場合、JEUS 8に適切なDDに変更するか否かを設定します

3. インストールを終えると、画面の上段に結果が表示されます。インストールが正常に実行されると、アプリケーション一覧で該当のアプリケーションを確認することができます。この際、アプリケーションはINSTALLED状態になります。

4.3.2. アプリケーションをドメインから削除

ドメインでアプリケーションが不要になった場合は、ドメインから削除できます。アプリケーションがINSTALLED状態か、あるいはDEPLOYED状態の場合、アプリケーションをアンインストールすることができます。

以下は、アプリケーションをドメインから削除する手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。ドメインで削除するアプリケーションを選択し、アプリケーション一覧の下段にある**[uninstall]**ボタンをクリックします。

2. **[uninstall]**ボタンをクリックすると表示される画面で、**[確認]**ボタンをクリックするとアプリケーションのアンインストールが行われます。また、**[キャンセル]**ボタンをクリックするとアンインストールはキャンセルされ、アプリケーションはINSTALLED状態で残ります。
3. アンインストールを終えると、画面の上段に結果が表示されます。アンインストールが正常に実行されると、アプリケーション一覧から該当のアプリケーション情報が削除されます。

4.3.3. アプリケーションのデプロイ

本節では、WebAdminとコンソール・ツールを使用してアプリケーションをデプロイする方法について説明します。

デプロイを実行して、インストールされたアプリケーションをサービスできるようにします。以下は、アプリケーションをデプロイする3つの方法です。

- ドメインにインストールしたアプリケーションのデプロイ
- アプリケーション・リポジトリにあるアプリケーションのデプロイ
- パスを指定してデプロイ

コンソール・ツールを使用したランタイム・デプロイについて、以下のように想定して説明します。

- アプリケーションは、「[4.1. アプリケーションの作成](#)」で作成したmyApp.earを使用します。
- JEUSのドメイン名は「domain1」であり、アプリケーションをデプロイするサーバーは「server1」です。

参考

deploy-applicationコマンドを実行するときに使用できるオプションについては、『*JEUS リファレンスガイド*』の「4.2.6.4. deploy-application」を参照してください。

JEUSでは、コンソール・ツールを使ってアプリケーションをランタイム・デプロイする際、アプリケーションの位置に応じて以下の3つの方法を提供しています。各方法は、デプロイ、開始、停止、再デプロイ、アンデプロイの順でコマンドを実行し、各コマンドを実行した後は、**applist**を実行して各段階でのアプリケーションの状態を確認します。

JEUSを直接管理できるコンソール・ツールを使ってDASに接続し、サーバーやクラスターの制御および監視を行います。また、アプリケーションをデプロイすることも可能です。

以下は、コンソール・ツールで使用するアプリケーションのデプロイに関連するコマンドです。

[表 4.1] アプリケーション関連のコマンド

コマンド	説明
distribute-application	アプリケーションをデプロイできるように、デプロイ対象にアプリケーションをコピーし、アプリケーション・サービスのための事前準備を行います
deploy-application	アプリケーションをデプロイします。 デプロイとは、デプロイ対象にアプリケーションをコピーし(配布)、アプリケーション・サービスを開始することです。 デプロイが正常に完了するとRUNNING状態になり、配布後サービス段階で障害が発生するとDISTRIBUTED状態になります
start-application	DISTRIBUTED状態のアプリケーションを実行します。実行中の状態はSTARTINGであり、実行が完了するとRUNNING状態になります
stop-application	RUNNING状態のアプリケーションを停止します。実行中の状態はSTOPPINGであり、実行が完了するとDISTRIBUTED状態になります
redploy-application	デプロイ済みのアプリケーションの内容が変更された場合、変更内容を反映して再デプロイします。デプロイと同じプロセスで行われ、各段階での状態も同じです
undeploy-application	アプリケーション・サービスを終了し、デプロイ対象からアプリケーションを削除します
application-info	ドメインに存在するアプリケーションの情報を出力します
add-application-target	デプロイ済みのアプリケーションにサービス対象を追加します。サービス対象はサーバーやクラスターです
remove-application-target	デプロイ済みのアプリケーションからサービス対象を削除します。サービス対象はサーバーやクラスターです

参考

コンソール・ツールの説明と各コマンドについては、『JEUS リファレンスガイド』の「4.2. jeusadmin」を参照してください。

4.3.3.1. ドメインにインストールしたアプリケーションのデプロイ

WebAdminの使用

以下は、WebAdminでドメインにインストールしたアプリケーションをデプロイする手順について説明します。

1. WebAdminの左側のメニューで[Applications]を選択すると、[Deployed Application]画面に移動します。アプリケーション一覧からデプロイするアプリケーションをチェックボックスから選択し、アプリケーション一覧の下段にある[deploy]ボタンをクリックします。

2. 「**Server**」項目から、デプロイ対象のサーバーを選択して[**確認**]ボタンをクリックすると、アプリケーションのデプロイが行われます。デプロイではなく配布までのみ行う場合は、**詳細設定領域**の「**Only Distribute**」項目にチェックして[**確認**]ボタンをクリックします。この場合、アプリケーションをサービスするためにはアプリケーションを開始する必要があります。
3. デプロイを終えると、画面の上段に結果が表示されます。デプロイが正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはRUNNING状態になります。配布のみを行った場合、アプリケーションはDISTRIBUTED状態になります。

コンソール・ツールの使用

install-applicationコマンドを使用してアプリケーションをドメインにインストールし、INSTALL_HOME(DOMAIN_HOME/.applications)に格納します。この際、アプリケーションのIDをオプションとして指定できます。IDを指定していない場合は、myApp_earがアプリケーションのIDになります。アプリケーションがドメインにインストールされた後、**deploy-application**コマンドを使用して対象サーバーにアプリケーションをデプロイします。

[例 4.1] コンソール・ツールでドメインにインストールしたアプリケーションのデプロイ

```
-----
using deploy command for application with install application
-----

[DAS]domain1.adminServer>install-application -id myApp /usr/apphome/myApp.ear
Successfully installed the application [myApp].

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati|Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    |           | INSTAL|         |         | ${INSTALL_HOME}/myApp/m|
|          |           | LED   |         |         | yApp.ear             |
+-----+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>deploy myApp -servers server1
deploy the application for the application [myApp] succeeded.

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati|Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
```

```

+-----+-----+-----+-----+-----+-----+
| myApp    | EAR      | RUNNING | server1  |          | ${INSTALL_HOME}/myAp |
|          |          |          |          |          | p/myApp.ear           |
+-----+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>stop-application myApp
stop the application for the application [myApp] succeeded.

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server  | Cluster | Application Path |
| on ID    | Type      |      | Targets | Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR      | DISTRIB| server1  |          | ${INSTALL_HOME}/myApp/m |
|          |          | UTED   |          |          | yApp.ear           |
+-----+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>start-application myApp
start the application for the application [myApp] succeeded.

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server  | Cluster | Application Path |
| on ID    | Type      |      | Targets | Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR      | RUNNING | server1  |          | ${INSTALL_HOME}/myAp |
|          |          |          |          |          | p/myApp.ear           |
+-----+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>redploy-application myApp
redploy application on das for the application [myApp] succeeded.

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server  | Cluster | Application Path |
| on ID    | Type      |      | Targets | Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR      | RUNNING | server1  |          | ${INSTALL_HOME}/myAp |
|          |          |          |          |          | p/myApp.ear           |
+-----+-----+-----+-----+-----+-----+
=====

```

```

| | | | |p/myApp.ear |
+-----+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>undeploy myApp
Undeploying [myApp] (This may take time due to graceful undeployment) .....
undeploy the application for the application [myApp] succeeded.
successfully undeployed (elapsed = 415ms)

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati|Application| State | Server | Cluster | Application Path |
| on ID | Type | | Targets | Targets | |
+-----+-----+-----+-----+-----+-----+
| myApp | EAR | INSTAL| | | ${INSTALL_HOME}/myApp/m|
| | | LED | | | yApp.ear |
+-----+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>uninstall-application myApp
uninstall the application for the application [myApp] succeeded. : Successfully
deleted [myApp].

[DAS]domain1.adminServer>application-info
No applications exist in this domain.
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Application| Application | State | Server | Cluster | Application |
| ID | Type | | Targets | Targets | Path |
+-----+-----+-----+-----+-----+-----+
(No data available)
=====

```

4.3.3.2. アプリケーション・リポジトリにあるアプリケーションのデプロイ

アプリケーション・リポジトリを追加すると、リポジトリに存在するアプリケーションをデプロイします。アプリケーション・リポジトリの機能とリポジトリを追加、削除する方法の詳細については、「[1.5.1. アプリケーション・リポジトリの追加/削除/照会](#)」を参照してください。

WebAdminの使用

WebAdminを使用してアプリケーション・リポジトリにあるアプリケーションをデプロイする方法は、ドメインにインストールされているアプリケーションをデプロイする方法と同じです。アプリケーションのデプロイが完

了すると、画面の上段に結果が表示されます。デプロイが正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはRUNNING状態になります。

コンソール・ツールの使用

以下は、コンソール・ツールを使用してアプリケーション・リポジトリにあるアプリケーションをデプロイする例です。

[例 4.2] コンソール・ツールでアプリケーション・リポジトリにあるアプリケーションをデプロイ

```
[DAS]domain1.adminServer>add-application-repository /home/user1/apps
Successfully performed the ADD operation for An application repository.
Check the results using "add-application-repository or
list-application-repositories"
```

```
[DAS]domain1.suok>list-application-repositories
Application Repositories
=====
+-----+-----+-----+-----+-----+-----+
|                                     Path to Application Repository                                     |
+-----+-----+-----+-----+-----+-----+
| /home/user1/apps                                                            |
+-----+-----+-----+-----+-----+-----+
=====
```

```
[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati|Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                  |
+-----+-----+-----+-----+-----+-----+
| exploded |           | INSTAL|         |         | /home/user1/apps/explod|
|           |           | LED   |         |         | ed                     |
+-----+-----+-----+-----+-----+-----+
=====
```

```
[DAS]domain1.adminServer>deploy exploded -servers server1
deploy the application for the application [exploded.war] succeeded.
```

```
[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati|Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                  |
+-----+-----+-----+-----+-----+-----+
=====
```

exploded	WAR	RUNNING	server1		/home/user1/apps/exp
					loded
+-----+-----+-----+-----+-----+-----+					
=====					

4.3.3.3. パスを指定してデプロイ

以下は、上位ディレクトリーをアプリケーション・リポジトリーに追加できない場合に、pathオプションを設定してデプロイする方法について説明します。DASが存在するマシンにあるアプリケーションの絶対パスを設定してデプロイする機能については、「[1.6. パスを指定してデプロイ](#)」を参照してください。

WebAdminの使用

以下は、WebAdminでパスを指定してデプロイする方法についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧の上段の**[deploy]**ボタンをクリックすると、ドメインに新しいアプリケーションをデプロイできる画面に移動します。
2. 「**Path**」項目にデプロイするアプリケーションの絶対パスを入力します。**[入力]**ボタンをクリックすると、DASに存在するアプリケーションのパスを選択できます。「**Server**」項目でデプロイ対象のサーバーである「server1」を選択し、**[確認]**ボタンをクリックすると、アプリケーションのデプロイが行われます。必要に応じて、**詳細設定**でデプロイのオプションを設定できます。
3. デプロイを終えると、画面の上段に結果が表示されます。デプロイが正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはRUNNING状態になります。

コンソール・ツールの使用

以下は、コンソール・ツールを用いてパスを指定してデプロイする方法です。

[例 4.3] コンソール・ツールでパスを指定してデプロイ

```
[DAS]domain1.adminServer>deploy -path /home/user1/apps/myApp.ear -servers server1
deploy the application for the application [/home/user1/apps/myApp.ear] succeeded.
```

```
[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Applicati | Application | State | Target | Target | Application Path |
| on ID    | Type       |      | Servers | Clusters |                  |
+-----+-----+-----+-----+-----+-----+
| myApp.ear | EAR        | RUNNING | server1 |          | /home/user1/apps/ |
```

					myApp.ear	
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+
=====	=====	=====	=====	=====	=====	=====

4.3.4. アプリケーションの再デプロイ

アプリケーションが変更された場合は、デプロイを実行してアプリケーションがサービスされるようにできます。詳細については、「[2.2. グレースフル再デプロイ](#)」を参照してください。

以下は、アプリケーションを再デプロイする手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると**[Deployed Application]**画面に移動します。アプリケーション一覧からデプロイするアプリケーションを選択し、一覧の右側にある**[redeploy]**ボタンをクリックします。
2. 「**Timeout**」項目で、グレースフル再デプロイのタイムアウトと、その他必要な再デプロイのオプションを設定して**[確認]**ボタンをクリックすると、アプリケーションの再デプロイが行われます。
3. 再デプロイを終えると、画面の上段に結果が表示されます。

4.3.5. アプリケーションのアンデプロイ

ドメインでサービスされているアプリケーション・サービスを停止するには、アンデプロイを実行します。

以下は、アプリケーションをアンデプロイする手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧からアンデプロイするアプリケーションをチェックボックスから選択し、アプリケーション一覧の下段にある**[undeploy]**ボタンをクリックします。
2. 「**Timeout**」項目を設定して、アプリケーションが処理中のサービスを保証するようにします。オプションを設定していない場合、アプリケーションは処理中の要求がすべて完了するまで5分(300秒)間アンデプロイを行いません。グレースフル・アンデプロイのためにタイムアウトを設定して**[確認]**ボタンをクリックすると、アンデプロイが行われます。アプリケーションのグレースフル・アンデプロイについての説明は、「[2.1. グレースフル・アンデプロイ](#)」を参照してください。
3. アンデプロイを終えると、画面の上段に結果が表示されます。アンデプロイが正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはINSTALLED状態になります。

4.3.6. アプリケーションの開始

アプリケーションのデプロイ時に「Only Distribute」オプションを指定して配布のみ実行した場合や、サービス中のアプリケーションを停止してDISTRIBUTED状態となったアプリケーションを再度サービスする場合は、開始を行います。

以下は、一時停止されたアプリケーションのサービスを開始する手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧から、サービスを開始するDISTRIBUTED状態のアプリケーションをチェックボックスから選択し、アプリケーション一覧の下段にある**[start]**ボタンをクリックします。
2. 開始を終えると、画面の上段に結果が表示されます。開始が正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはRUNNING状態になります。

4.3.7. アプリケーションの停止

サービス中のアプリケーションにstopコマンドを実行して、アプリケーションのサービスを一時停止することができます。

以下は、アプリケーションがサービスを停止する手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧からサービスを停止するアプリケーションをチェックボックスから選択し、アプリケーション一覧の下段にある**[stop]**ボタンをクリックします。
2. **[stop]**ボタンをクリックすると表示される画面で**[確認]**ボタンをクリックすると、アプリケーション・サービスが停止し、それ以降の要求は処理しません。**[キャンセル]**ボタンをクリックするとアプリケーション・サービスの停止がキャンセルされます。
3. 停止を終えると、画面の上段に結果が表示されます。停止が正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはDISTRIBUTED状態になります。

4.3.8. サービス中のアプリケーションにサーバーを追加

アプリケーション・サービスを拡張するために、アプリケーションのサービス対象にサーバーやクラスターを追加する機能を提供します。既にドメインに存在するアプリケーションを特定のサーバーにのみデプロイできないため、サービス対象を追加するコマンド(add-target)を提供しています。add-targetコマンドは、アプリケーションがRUNNING状態あるいはDISTRIBUTED状態の場合に実行可能です。

参考

アプリケーションのサービス対象にサーバーを追加する場合は、サーバーがもう1つ必要です。サーバーを追加する方法は、『JEUS サーバガイド』の「2.2. サーバーの追加」を参照してください。

以下は、サービス中のアプリケーションのサービス対象にサーバーを追加する手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧からアプリケーションをサービスするサーバーを追加するためにアプリケーションを選択し、一覧の右側にある**[add-target]**ボタンをクリックします。
2. 「**Server**」項目から、アプリケーションのサービス対象に追加するサーバーを選択して**[確認]**ボタンをクリックすると、該当のサーバーにアプリケーションのデプロイが行われます。
3. デプロイ対象の追加を終えると、画面の上段に結果が表示されます。
4. 正常に実行されたら、アプリケーション一覧から該当のアプリケーションを選択して、「**Server**」と「**Running Servers**」項目に選択したサーバーが追加されていることを確認できます。

4.3.9. サービス中のアプリケーションからサービス中のサーバーを削除

アプリケーション・サービスを縮小するために、アプリケーションのサービス対象からサーバーやクラスターを削除する機能を提供します。既にドメインに存在するアプリケーションを特定のサーバーでのみアンデプロイできないため、サービス対象から削除するコマンド(remove-target)を提供しています。remove-targetコマンドは、アプリケーションがRUNNING状態あるいはDISTRIBUTED状態の場合に実行可能です。

以下は、サービス中のアプリケーションからサービス中のサーバーを削除する手順についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧からサービスを停止するアプリケーションを選択し、一覧の右側にある**[remove-target]**ボタンをクリックします。
2. 「**Server**」項目から、アプリケーション・サービスを行わないサーバーを選択して**[確認]**ボタンをクリックすると、該当のサーバーでのみアプリケーションのアンデプロイが行われます。
3. アプリケーションのサービス対象の削除を終えると、画面の上段に結果が表示されます。
4. 正常に実行されたら、アプリケーション一覧から該当のアプリケーションを選択して、「**Server**」と「**Running Servers**」項目から選択したサーバーが削除されたことを確認できます。

4.3.10. アプリケーション情報の確認

本節では、WebAdminとコンソール・ツールを使用してアプリケーション情報を確認する方法について説明します。

4.3.10.1. WebAdminの使用

WebAdminを使用して、以下のアプリケーション情報を確認することができます。

- アプリケーションがどのようにデプロイされているのか？
- アプリケーションがどのように構成されているのか？

アプリケーションのデプロイ情報の確認

アプリケーションがサーバーやクラスターにどのような属性でデプロイされたのかを確認できます。

以下は、WebAdminでアプリケーションのデプロイ情報を確認する方法です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧でアプリケーションIDをクリックすると、アプリケーション情報を確認できます。
2. 確認できる情報は、アプリケーションのID、DAS上のアプリケーションのパス、アプリケーションのタイプ、JEUSが提供するデプロイ時に設定するオプション情報です。デプロイのオプションについての詳細は、WebAdminの項目の説明や、『JEUS リファレンスガイド』の「4.2.6.4. deploy-application」を参照してください。

アプリケーション構成モジュール情報の確認



アプリケーションがどのように構成されているのか、実際のアプリケーションのモジュール情報を確認できます。

EARアプリケーションの場合は、EARアプリケーションを構成しているモジュール一覧を確認できます。アプリケーションを構成するモジュールのいずれかを選択すると、そのモジュールに関する構成情報の詳細を確認することができます。この情報はEJBモジュールとWebモジュールについてのみ確認可能です。EJBモジュールを選択した場合はEJB Bean一覧を、Webモジュールを選択した場合はサーブレット一覧を確認することができます。

WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deppoyed Application]**画面に移動します。アプリケーション一覧からアプリケーションIDをクリックすると、アプリケーションのパス、形式、デプロイ時間、対象サーバーなど、基本的な情報を確認することができます。

[図 4.2] WebAdminでのアプリケーションの構成情報の確認

Deployed Application

ID	myApp
Path	D:\TmaxSoft\jeus8\domains\domain1\applications\myApp\myApp.ear
Type	EAR
Application Time	Tue Oct 04 13:37:26 KST 2016
 Targets	
Server	adminServer
Running Servers	adminServer [RUNNING]
 Options	
Security Domain Name	SYSTEM_DOMAIN

アプリケーション・デプロイ情報画面でアプリケーション情報の「**Running Servers**」から特定のサーバーをクリックすると、アプリケーションの構成情報を確認することができます。

[図 4.3] WebAdminでのアプリケーションの構成情報の確認

Application [myApp]

Module Name	Unique Module Name	Module Type
appclient	myApp#appclient	CAR
web	myApp#web	WAR
ejb	myApp#ejb	EJB

EARアプリケーションの場合は、上のようなアプリケーションを構成するモジュール一覧に加え、アプリケーションを構成しているEJBモジュールとWebモジュールの情報も確認できます。

参考

モジュールの情報はEJBかWEBの場合にのみ確認できます。クライアント・モジュールやリソース・モジュールの場合は、モジュール情報を確認する機能を提供しません。

● EJBモジュール情報の確認

以下は、EARアプリケーションのmyAppを構成しているEJBモジュール「ejb」に関する情報を確認する方法です。

1. アプリケーションmyAppのモジュール一覧から「**ejb**」を選択すると、EJBモジュールを構成しているBean一覧を確認できます。

[図 4.4] WebAdminでのEJBモジュール情報の確認

■ General information about the EJB module [ejb].

Module Name	Unique Module Name
ejb	myApp#ejb

■ Beans

Bean Name	Type	Local Export Name	Remote Export Name
CalcBean	StatelessSessionBean		
HelloBean	StatelessSessionBean		

2. モジュール「ejb」のBean一覧から「**HelloBean**」を選択すると、HelloBeanの詳細情報を確認できます。

[図 4.5] WebAdminでのEJBモジュールのBean情報の確認

■ Bean name: HelloBean

Name	(Count)	WaterMark(High:Low:Cur)	Bound(Upper:Lower)	Time(Max:Min:Total)
create	times(0)			
comitted	transaction(0)			
total-remote-thread		thread(100:100:100)		
timed-rb	transaction(0)			
remove	times(0)			
active-bean		bean(0:0:0)		
request	request(0)			
total-bean		bean(0:0:0)		
rolledback	transaction(0)			
active-thread		thread(0:0:0)		
MethodReadyCount		bean(0:0:0)		

- Webモジュール情報の確認

以下は、EARアプリケーションのmyAppを構成しているWEBモジュール「web」に関する情報を確認する方法です。

1. アプリケーションmyAppのモジュール一覧で「web」を選択すると、Webモジュールを構成しているサーブレット一覧を確認できます。

[図 4.6] WebAdminでのWEBモジュールのサーブレット情報の確認

■ General information about the web module [web].

Module Name	Unique Module Name	Context Path	Target Session Cluster	Command
web	myApp#web	/hello		Thread Info

■ Servlets

Name	Class	State	Count	Attribute	RegType	URLPatterns
dvt.myApp.HelloServlet	dvt.myApp.HelloServlet	NOT_LOADED	-1	SYNC	ANNOTATION	N/A

■ Filters

Name	Class	Attribute	RegType	URLPatterns	Servlets
------	-------	-----------	---------	-------------	----------

該当する内容が存在しません。

■ Listeners

Name	Type	RegType
------	------	---------

該当する内容が存在しません。

■ Jeus web deployment descriptor

2. 「**Command**」列の[**thread-info**]ボタンをクリックすると、WEBモジュールの要求スレッドに関する情報を確認できます。

[図 4.7] WebAdminでのWEBモジュールのサーブレット要求スレッド情報の確認

■ Thread information for the server [adminServer]

該当する内容が存在しません。

■ Asynchronous processing background threads for the web context[myApp#web]

tid	name	state	elapsed	uri
-----	------	-------	---------	-----

該当する内容が存在しません。

■ elapsed: Elapsed time (ms)

■ Asynchronous processing background thread statistics for the web context[myApp#web]

	total	active	idle	blocked	reconn
The number of threads.	0	0	0	0	0

■ total = active + idle, reconn: reconnecting

■ Asynchronous processing dispatch threads for the web context[myApp#web]

tid	name	state	elapsed	uri
-----	------	-------	---------	-----

該当する内容が存在しません。

■ elapsed: Elapsed time (ms)

■ Asynchronous processing dispatch thread statistics for the web context[myApp#web]

4.3.10.2. コンソール・ツールの使用

コンソール・ツールの**application-info**コマンドを使ってmyApp.earアプリケーションの情報を照会することができます。

以下は、コンソール・ツールを使ってアプリケーション情報を確認する例です。各オプションについての詳細は、『*JEUS リファレンスガイド*』の「4.2.6.3. application-info」を参照してください。

[例 4.4] コンソール・ツールでアプリケーション情報の確認

```
[DAS]domain1.adminServer> application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type       |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+
| myApp   | EAR       | RUNNING | server1   |           | ${INSTALL_HOME}/myAp |
|         |           |         |           |           | p/myApp.ear           |
+-----+-----+-----+-----+-----+-----+
=====

```

```

[DAS]domain1.adminServer>application-info -id myApp -server server1
Application information for the server [server1] in the domain [domain1].
=====

```

```

+-----+-----+-----+-----+-----+-----+
| Application| Application Name| Application | State| Server | Cluster |
| ID         |                 | Type       |      | Targets| Targets  |
+-----+-----+-----+-----+-----+-----+
| myApp      | myApp          | ear        | RUNNI| server1 |          |
|            |                 |            | NG   |         |          |
+-----+-----+-----+-----+-----+-----+
=====

```

```

[DAS]domain1.adminServer>application-info -id myApp -server server1 -detail
Application name : myApp
Application [myApp]
=====

```

```

+-----+-----+-----+-----+
| Module Name | Unique Module Name | Module Type |
+-----+-----+-----+-----+
| ejb         | myApp#ejb          | EJB         |
| appclient   | myApp#appclient    | CAR         |
| web         | myApp#web          | WAR         |
+-----+-----+-----+-----+

```

To view detailed information about EJBs or web modules in an EAR, use the "-module" or "-type" option.

```

[DAS]domain1.adminServer>application-info -id myApp -server server1 -detail -module ejb

```

```

Application name : myApp
General information about the EJB module [ejb].
=====

```

```

+-----+-----+-----+
| Module Name | Unique Module Name |
+-----+-----+-----+
| ejb         | myApp#ejb          |
+-----+-----+-----+
=====

```

Beans

```

=====
+-----+-----+-----+-----+
| Bean Name |          Type          | Local Export Name | Remote Export Name |
+-----+-----+-----+-----+
| HelloBean | StatelessSessionBean   |                   |                   |
+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>application-info -id myApp -server server1 -detail -type
war
Application name : myApp
There are no EJBs in this module.
General information about the web module [web].
=====
+-----+-----+-----+
| Module Name | Unique Module Name | Context Path |
+-----+-----+-----+
| web         | myApp#web          | /hello       |
+-----+-----+-----+
=====

Servlets
=====
+-----+-----+-----+-----+-----+-----+-----+
| Name | Class | State | Count | Attribute | RegType | URLPatterns |
+-----+-----+-----+-----+-----+-----+-----+
| HelloSer | dvt.deployment.se | READY | 0 | SYNC | WEB_XML | /HelloServ |
| vlet | rvlet.HelloServlet | | | | | let |
+-----+-----+-----+-----+-----+-----+-----+
=====

Filters
=====
+-----+-----+-----+-----+-----+-----+-----+
| Name | Class | Attribute | RegType | URLPatterns | Servlets |
+-----+-----+-----+-----+-----+-----+-----+
(No data available)
=====

Listeners
=====
+-----+-----+-----+
| Name | Type | RegType |
+-----+-----+-----+
(No data available)
=====

```



```

EJBs
=====
+-----+-----+-----+-----+
| Bean Name | Type | Local Export Name | Remote Export Name |
+-----+-----+-----+-----+
(No data available)
=====

[DAS]domain1.suok>application-info -id myApp -server server1 -detail -module ejb
-bbean HelloBean
bean HelloBean
Application name : myApp
Module name : ejb
Bean name: HelloBean
=====
+-----+-----+-----+-----+
| Name | (Count) | WaterMark(High:Low | Bound(Upper:L | Time(Max:Min:T |
| | | :Cur) | ower) | otal) |
+-----+-----+-----+-----+
| create | times(0) | | | |
+-----+-----+-----+-----+
| remove | times(0) | | | |
+-----+-----+-----+-----+
| timed-rb | transactio | | | |
| | n(0) | | | |
+-----+-----+-----+-----+
| request | request(0) | | | |
+-----+-----+-----+-----+
| active-bean | | bean(0:0:0) | | |
+-----+-----+-----+-----+
| rolledback | transactio | | | |
| | n(0) | | | |
+-----+-----+-----+-----+
| total-bean | | bean(0:0:0) | | |
+-----+-----+-----+-----+
| comitted | transactio | | | |
| | n(0) | | | |
+-----+-----+-----+-----+
| MethodReadyCou | | bean(0:0:0) | | |
| nt | | | | |
+-----+-----+-----+-----+
| active-thread | | thread(0:0:0) | | |
+-----+-----+-----+-----+
| total-remote-t | | thread(100:100:100) | | |
| hread | | | | |
+-----+-----+-----+-----+
=====

```

application-infoコマンドは、ドメインに存在するすべてのアプリケーションの情報を出力します。基本的に出力される情報は以下のとおりです。

項目	説明
Application ID	アプリケーションのIDです。この値はドメインで一意である必要があります
Applicatio Type	アプリケーションの5つのタイプです。以下のいずれかのタイプにできます <ul style="list-style-type: none">– EAR : アプリケーション– EJB : EJBモジュール– WAR : Webアプリケーション・モジュール– RAR : リソース・アダプター・モジュール– CAR : アプリケーション・クライアント・モジュール
state	ドメインでのアプリケーションの状態です。以下のいずれかの状態となります <ul style="list-style-type: none">– INSTALLED– DISTRIBUTED– RUNNING– DEPLOYED 各状態についての詳細は、「 1.1.3. アプリケーションの状態 」を参照してください
Server Target	アプリケーションがデプロイされているサーバーです
Cluster Target	アプリケーションがデプロイされているクラスターです
Application Path	DASに存在するアプリケーション・ファイルのパスです

4.4. ステージング・モード・デプロイ

explodedモジュール形式のアプリケーションは、ファイルを圧縮して、他のマシンのMSIにデプロイできます。これをステージング・モード・デプロイといいます。アプリケーション・ファイルがアプリケーション・リポジトリに存在するか、DASが存在するマシンの絶対パスを設定してデプロイすることができます。ステージング・モード・デプロイ機能の詳細については、「[1.7. ステージング・モード・デプロイ](#)」を参照してください。

WebAdminの使用

以下は、WebAdminを使用してパスを指定し、デプロイする方法についての説明です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。アプリケーション一覧からINSTALLED状態のexplodedアプリケーションを選択し、アプリケーション一覧の下段にある**[deploy]**ボタンをクリックして「**Path**」項目を入力します。
2. 「**Server**」項目から、デプロイ対象のサーバーを選択します。
3. デプロイ情報の入力画面の**詳細設定**で「**Staging**」項目をチェックして**[確認]**ボタンをクリックすると、アプリケーションのデプロイが行われます。

[図 4.8] WebAdmin - ステージング・モード・デプロイ

詳細設定		すべてを開く
Upgrade	<input type="checkbox"/>	
Only Distribute	<input type="checkbox"/>	
Concurrent	<input type="checkbox"/>	
Options		
Classloading	<input type="text"/>	
Use Fast Deploy	<input type="checkbox"/>	
Keep Generated	<input type="checkbox"/>	
Shared	<input type="checkbox"/>	
Staging	<input type="checkbox"/>	
Security Domain Name	<input type="text"/>	
Auto Redeploy Interval	<input type="text" value="ms"/>	
Plan	<input type="text"/>	
Context Path	<input type="text"/>	
Node JAVA Context	<input type="checkbox"/>	

4. デプロイを終えると、画面の上段に結果が表示されます。デプロイが正常に実行されると、アプリケーション一覧で表示される該当のアプリケーションはRUNNING状態になります。
5. server1のログから、アプリケーションがステージング・モードでデプロイされたことを確認できます。

```
...
[2016.08.09 23:17:48][2] [server1-40] [WEB-1032] Distributed the web context
[exploded] information
- Virtual host      : DEFAULT_HOST
- Context path     : /exploded
- Document base    :
/home/user1/jeus/domains/domain1/servers/server1/.workspace/deployed/exploded/exploded_war_
...
```

コンソール・ツールの使用

以下は、コンソール・ツールを使用したステージング・モード・デプロイの方法です。stagingオプションを指定してdeployコマンドを実行します。

[例 4.5] コンソール・ツールを使用したステージング・モード・デプロイ

```
[DAS]domain1.adminServer>deploy exploded -servers server1 -staging
deploy the application for the application [exploded] succeeded.

[DAS]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type       |       | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| exploded | WAR        | RUNNING | server1 |          | /home/user1/apps/exp|
|          |            |         |         |          | loded                |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1 |          | ${INSTALL_HOME}/myAp|
|          |            |         |         |          | p/myApp.ear          |
+-----+-----+-----+-----+-----+-----+
=====
```

4.5. デプロイメント・プランを利用したデプロイ

デプロイメント・プランは、デプロイ時にアプリケーションのDDの内容を変更できるアプリケーション外部の設定ファイルです。JEUSはXML形式のデプロイメント・プランを定義し、EJB、Webアプリケーション、EARの標準DDとJEUS DDにデプロイメント・プランを適用できるようにサポートします。アプリケーションのデプロイ時にデプロイメント・プランの使用について明示すると、デプロイ時にDDとデプロイメント・プランの設定をマージしてアプリケーションの設定を確定した後、デプロイを実行します。

本節では、デプロイメント・プランの設定方法と動作方式について説明します。また、WebAdminとコンソール・ツールでデプロイメント・プランを利用したデプロイ方法について説明します。

4.5.1. デプロイメント・プランの設定および動作方式

デプロイメント・プランの作成例を通じて、設定方法と動作方式について説明します。

デプロイメント・プランの設定

以下は、デプロイメント・プランの作成例です。

```

<?xml version="1.0" encoding="UTF-8"?>
<jeus-deployment-plan xmlns="http://www.tmaxsoft.com/xml/ns/jeus"
    xmlns:jeus="http://www.tmaxsoft.com/xml/ns/jeus"
    xmlns:javaee="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:j2ee="http://java.sun.com/xml/ns/j2ee">

  <descriptors>
    <!-- For standalone EJB -->
    <descriptor>
      <uri>META-INF/ejb-jar.xml</uri>
      <configurations>
        <configuration>
          <action>REPLACE</action>
          <xpath>/j2ee:ejb-name[.='ByeBean']</xpath>
          <value>
            <![CDATA[<ejb-name>HiBean</ejb-name>]]>
          </value>
        </configuration>
        <configuration>
          <action>DELETE</action>
          <xpath>/child::j2ee:ejb-jar/child::j2ee:enterprise-beans/
child::j2ee:session/child::j2ee:local-home[.='HelloHomeLocal']</xpath>
        </configuration>
        <configuration>
          <action>APPEND_CHILD</action>
          <xpath>/child::j2ee:ejb-jar/descendant::j2ee:session[2]</xpath>

          <value>
            <![CDATA[<transaction-type>Bean</transaction-type>]]>
          </value>
        </configuration>
        <configuration>
          <action>INSERT_BEFORE</action>
          <xpath>/j2ee:ejb-jar/j2ee:enterprise-beans/j2ee:session/
j2ee:ejb-name[.='HelloBean']/../j2ee:transaction-type</xpath>
          <value>
            <![CDATA[<session-type>Stateless</session-type>]]>
          </value>
        </configuration>
      </configurations>
    </descriptor>
    <descriptor>
      <uri>META-INF/jeus-ejb-dd.xml</uri>
      <configurations>
        <configuration>
          <action>REPLACE</action>
          <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:export-name[.='ByeBean']</xpath>

```

```

        <value>
            <![CDATA[<export-name>HiBean</export-name>]]>
        </value>
    </configuration>
    <configuration>
        <action>DELETE</action>
        <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:local-export-name[.='HelloBeanLocal']</xpath>
    </configuration>
    <configuration>
        <action>APPEND_CHILD</action>
        <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean</xpath>

        <value>
            <![CDATA[<jeus-rmi>false</jeus-rmi>]]>
        </value>
    </configuration>
    <configuration>
        <action>INSERT_BEFORE</action>
        <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:ejb-name[.='HiBean']/../jeus:jeus-rmi</xpath>
        <value>
            <![CDATA[<thread-max>100</thread-max>]]>
        </value>
    </configuration>
</configurations>
</descriptor>
<!-- For standalone web application -->
<descriptor>
    <uri>WEB-INF/web.xml</uri>
    <configurations>
        <configuration>
            <action>REPLACE</action>
            <xpath>/child::javaee:web-app/child::javaee:servlet-mapping/
child::javaee:servlet-name[.='HiServlet']</xpath>
            <value>
                <![CDATA[<servlet-name>HelloServlet</servlet-name>]]>
            </value>
        </configuration>
        <configuration>
            <action>DELETE</action>
            <xpath>//javaee:load-on-startup</xpath>
        </configuration>
        <configuration>
            <action>APPEND_CHILD</action>
            <xpath>/javaee:web-app/descendant::javaee:login-config</xpath>

```

```

        <value>
            <![CDATA[<auth-method>BASIC</auth-method>]]>
        </value>
    </configuration>
    <configuration>
        <action>INSERT_BEFORE</action>
        <xpath>/javaee:web-app/javaee:env-entry/
javaee:env-entry-value[.='value1']</xpath>
        <value>

<![CDATA[<env-entry-type>java.lang.String</env-entry-type>]]>
        </value>
    </configuration>
</configurations>
</descriptor>
<descriptor>
    <uri>WEB-INF/jeus-web-dd.xml</uri>
    <configurations>
        <configuration>
            <action>REPLACE</action>
            <xpath>//jeus:enable-jsp</xpath>
            <value>
                <![CDATA[<enable-jsp>true</enable-jsp>]]>
            </value>
        </configuration>
        <configuration>
            <action>DELETE</action>

<xpath>/jeus:jeus-web-dd/child::jeus:max-instance-pool-size</xpath>
        </configuration>
        <configuration>
            <action>INSERT_BEFORE</action>
            <xpath>/jeus:jeus-web-dd/descendant::jeus:enable-jsp</xpath>
            <value>
                <![CDATA[<context-path>/hello</context-path>]]>
            </value>
        </configuration>
        <configuration>
            <action>APPEND_CHILD</action>
            <xpath>//jeus:jeus-web-dd</xpath>
            <value>

<![CDATA[<webinf-first></enabled>false</enabled></webinf-first>]]>
            </value>
        </configuration>
    </configurations>
</descriptor>

```

```

<!-- For EAR -->
<descriptor>
  <uri>META-INF/application.xml</uri>
  <configurations>
    <configuration>
      <action>REPLACE</action>
      <xpath>/javaee:application/javaee:library-directory</xpath>
      <value>
        <![CDATA[<library-directory>mylib</library-directory>]]>
      </value>
    </configuration>
  </configurations>
</descriptor>
<descriptor>
  <uri>ejb.jar/META-INF/ejb-jar.xml</uri>
  <configurations>
    <configuration>
      <action>REPLACE</action>
      <xpath>/javaee:ejb-jar/javaee:enterprise-beans/
javaee:session/javaee:ejb-class</xpath>
      <value>
        <![CDATA[<ejb-class>HelloBean</ejb-class>]]>
      </value>
    </configuration>
  </configurations>
</descriptor>
<descriptor>
  <uri>web.war/WEB-INF/web.xml</uri>
  <configurations>
    <configuration>
      <action>REPLACE</action>
      <xpath>/javaee:web-app/javaee:servlet-mapping/
javaee:servlet-name</xpath>
      <value>
        <![CDATA[<servlet-name>HelloServlet</servlet-name>]]>
      </value>
    </configuration>
  </configurations>
</descriptor>
</descriptors>
</jeus-deployment-plan>

```

以下は、<descriptor>の下位タグについての説明です。

- <descriptor>

<descriptor>は複数の<configuration>タグで構成されています。

– <configuration>

<configuration>は、ターゲットDDに適用する1つの設定単位として、<action>、<xpath>、<value>タグで構成されています。

タグ	説明
<action>	<p>DDの特定タグについて、どの設定を変更するのかを明示します。</p> <p>以下は、設定値についての説明です</p> <ul style="list-style-type: none"> – DELETE : DDの特定のタグを削除します – REPLACE : DDの特定のタグを任意のタグに代替します – APPEND_CHILD : DDの特定のタグの最後の子(last child)として、任意のタグを追加します – INSERT_BEFORE : DDの特定のタグの前の兄弟(previous sibling)として、任意のタグを挿入します
<xpath>	<p>DDの特定のタグをxpath expressionで指定します。この際、<xpath>標準に基づいて、<xpath>パスに位置するすべてのタグは、XML名前空間を通じて修飾名で表現される必要があります。そのため、デプロイメント・プランに各DDのXML名前空間を名前空間の接頭辞と一緒に宣言します。</p> <p>たとえば、JEUS DDにある任意のタグを<xpath>に指定する場合、<xpath>パスに位置するすべてのタグは、JEUS XML名前空間 (http://www.tmaxsoft.com/xml/ns/jeus)と一緒に修飾名で表現される必要があります。デプロイメント・プランでは、JEUS XML名前空間を「jeus」接頭辞とマッチングさせたため、<xpath>パスに位置するすべてのタグ名の前に「jeus」接頭辞を付けて修飾名を表現します</p>
<value>	<p><value>は、<action>値がREPLACE、APPEND_CHILD、INSERT_BEFOREの場合にのみ有効です。</p> <p><action>値に基づく設定値は以下のとおりです。</p> <ul style="list-style-type: none"> – REPLACE : <xpath>に指定したタグを代替する新しいタグを設定します – APPEND_CHILD : <xpath>に指定したタグに最後の子として追加するタグを設定します – INSERT_BEFORE : <xpath>に指定したタグの前の兄弟として挿入するタグを設定します <p><value>値として設定されているタグは、一般的にdepthを持つXML fragment形式を想定するため、<value>値は特別にCDATAセクションで表現します。CDATAセクションにアクションを実行するために必要なXMLフラグメントを入力します</p>

デプロイメント・プランの動作方式

上の作成例を基に、タグ別の動作方式について説明します。

- <descriptor>

デプロイメント・プランは、複数の<descriptor>タグで構成されています

<descriptor>単位で対象となるDDを指定し、これのために<uri>タグを使用します。アプリケーション・ファイルをルートと想定し、それによって決まるDDの相対パスを<uri>値にして、対象のDDを決定します。

– たとえば、スタンドアロンEJBモジュールの標準DDは、アプリケーション・ルートから常にMETA-INF/ejb-jar.xmlに存在しており、JEUS DDはMETA-INF/jeus-ejb-dd.xmlに存在しています。そのため、上記のデプロイメント・プランにおける最初の<descriptor>は、その<uri>値からスタンドアロンEJBモジュールの標準DD用であり、2番目の<descriptor>は、その<uri>値からスタンドアロンEJBモジュールのJEUS DD用となります。スタンドアロンWebアプリケーションやEARについても同様な規則が適用されます。

– <uri>値が「ejb.jar/META-INF/ejb-jar.xml」の<descriptor>は、EARに属するEJBモジュール(ファイル名がejb.jar)用であり、<uri>値が「web.war/WEB-INF/web.xml」の<descriptor>はEARに属するWEBモジュール(ファイル名がweb.war)用です。

– 1つのデプロイメント・プランは、複数の異なるアプリケーションのデプロイのために使用されます。デプロイするアプリケーションのDDと<uri>値がマッチングされ、有効な値と判断される<descriptor>のみ選択され、該当のアプリケーションをデプロイする際に適用される方式なので、それ以外の<descriptor>はデプロイに影響を与えません。

- <configuration>

以下は、最初の<descriptor>設定を例にして説明したものです。<descriptor>の設定によって、DDで実際に変更される事項についての説明です。

– 最初の<descriptor>は、前述どおりスタンドアロンEJBモジュールの標準DDを操作します。最初に行われる操作は、特定のタグの代替として、最初の<configuration>に表現されています。値が「ByeBean」という<ejb-name>タグを、値が「HiBean」の<ejb-name>タグで代替していることを確認できます。

– 2番目の<configuration>は、値が「HelloHomeLocal」の<local-home>タグの削除を表現します。

– 3番目の<configuration>は、2番目の<session>タグの最後の子として<transaction>タグの追加を表現します。

– 4番目の<configuration>は、<ejb-name>タグの値が「HelloBean」の<session>タグの前の兄弟として<session-type>タグの挿入を表現します。

4.5.2. デプロイメント・プランのインストール

デプロイメント・プランを利用してデプロイするためには、まずデプロイメント・プランをドメインにインストールします。アプリケーションと同様、デプロイに利用できるデプロイメント・プランもドメインにインストールされたものに限りです。デプロイメント・プランをインストールする際、ドメインでのデプロイメント・プランの識別子を設定することができます。

デプロイメント・プランは、WebAdminまたはコンソール・ツールを使用してインストールできます。

WebAdminの使用

以下は、WebAdminを使用してデプロイメント・プランをインストールする手順です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。**[plan]**ボタンをクリックすると、**[Deployment Plan]**画面に移動します。
2. **[Deployment Plan]**画面で**[install]**ボタンをクリックしてインストールするデプロイメント・プランのパスを設定し、適切な識別名を入力して**[確認]**ボタンをクリックします。
3. 入力した識別名でデプロイメント・プランがインストールされたことを確認できます。

コンソール・ツールの使用

コンソール・ツールで**install-deployment-plan**コマンドを使用してデプロイメント・プランをインストールできます。

```
[DAS]domain1.adminServer>installdp -path /home/dev/plans/jeus-deployment-plan.xml  
-name plan1  
Installing the deployment plan [plan1] was successful.
```

参考

install-deployment-planコマンドの使用方法についての詳細は、『*JEUS リファレンスガイド*』の「4.2.6.8. install-deployment-plan」を参照してください。

4.5.3. インストールしたデプロイメント・プランの確認

WebAdminまたはコンソール・ツールを使用してインストールしたデプロイメント・プランを確認できます。各デプロイメント・プランは、インストール時に設定したデプロイメント・プランの識別子で区別され、デプロイメント・プランごとにそれが適用されたアプリケーションの一覧が表示されます。デプロイメント・プランのファイルの内容も画面に出力できます。

WebAdminの使用

以下は、WebAdminを使用してインストールしたデプロイメント・プランを確認する手順です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。**[plan]**ボタンをクリックすると、**[Deployment Plan]**画面に移動します。
2. **[Deployment Plan]**画面で、インストールされたデプロイメント・プランの一覧が表示されます。
3. 表示された一覧からデプロイメント・プランの名前をクリックすると、その内容を確認できます。

コンソール・ツールの使用

コンソール・ツールで**deployment-plan-info**コマンドを使用して、ドメインにインストールしたデプロイメント・プランが適用されたアプリケーションの一覧を表示できます。特定のデプロイメント・プランを指定すると、該当するデプロイメント・プランの内容が画面に表示されます。

● デプロイメント・プランの一覧表示

```
[DAS]domain1.adminServer>dpinfo
The list of deployment plans installed in the domain and the applications to
which each deployment plan applies
=====
+-----+-----+
|          Deployment plan          | Applications |
+-----+-----+
| plan1                             |              |
+-----+-----+
=====
```

● 特定のデプロイメント・プランの内容表示

```
[DAS]domain1.adminServer>dpinfo -name plan1
<?xml version="1.0" encoding="UTF-8"?>
<jeus-deployment-plan xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
  ...
</jeus-deployment-plan>
```

参考

deployment-plan-infoコマンドの使用方法的詳細については、『*JEUS リファレンスガイド*』の「4.2.6.5. deployment-plan-info」を参照してください。

4.5.4. デプロイメント・プランを適用したデプロイ

WebAdminまたはコンソール・ツールを使用してインストールしたデプロイメント・プランを適用してデプロイを実行できます。

WebAdminの使用

以下は、WebAdminを使用してデプロイメント・プランを適用したデプロイの手順です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。デプロイするアプリケーションを選択して**[deploy]**ボタンをクリックします。
2. デプロイ項目設定画面で「**Plan**」項目にインストールされたプランの識別名を入力して**[確認]**ボタンをクリックします。
3. 画面上段にデプロイが完了したというメッセージが表示され、アプリケーションがRUNNING状態に変更されたことを確認できます。

コンソール・ツールの使用

コンソール・ツールで**deploy-application**コマンドを使用してデプロイメント・プランを利用したデプロイを実行できます。

```
[DAS]domain1.adminServer>deploy webapp -all -plan plan1  
deploy the application for the application [webapp] succeeded.
```

4.5.5. アプリケーションに適用されたデプロイメント・プランの確認

WebAdminまたはコンソール・ツールを使用して、特定のアプリケーションに適用されたデプロイメント・プランを確認することができます。

WebAdminの使用

以下は、WebAdminを使用してアプリケーションに適用されたデプロイメント・プランを確認する手順です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。情報を確認するアプリケーションIDをクリックすると、詳細情報を確認できます。
2. 選択したアプリケーションの詳細情報には、適用されたデプロイメント・プランも含まれます。

コンソール・ツールの使用

コンソール・ツールで**application-info**コマンドを使用して、アプリケーションに適用されたデプロイメント・プランを確認できます。

```
[DAS]domain1.adminServer>appinfo -id webapp -detail
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Appli| Applic|State|Server|Cluster|      Running      | Applicati| Applicati| Plan|
| cation| ation |      |Target|Targets| Servers           | on Path  | on Time  | Name |
| ID   | Type  |      |s     |        |                   |          |          |      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| webapp| WAR   | RUNN| ALL  | ALL  | server1,serve| ${INSTALL| Tue May | plan1|
|        |       | ING |      |      | r2,server3,adm|_HOME}/web| 28      |      |
|        |       |     |      |      | inServer      | app/deploy| 22:45:13 |      |
|        |       |     |      |      |                | ment_plan_| KST 2013 |      |
|        |       |     |      |      |                | web.war   |          |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
=====
```

4.5.6. デプロイメント・プランのアンインストール

WebAdminまたはコンソール・ツールを使用して、ドメインからデプロイメント・プランをアンインストールできます。アンインストールされたデプロイメント・プランはドメインで無効になり、デプロイ時にも使用できません。

WebAdminの使用

以下は、WebAdminを使用してインストールされたデプロイメント・プランをアンインストールする手順です。

1. WebAdminの左側のメニューで**[Applications]**を選択すると、**[Deployed Application]**画面に移動します。**[plan]**ボタンをクリックすると、**[Deployment Plan]**画面に移動します。
2. **[Deployment Plan]**画面にインストールされたデプロイメント・プランの一覧が表示されます。削除するデプロイメント・プランを選択して**[uninstall]**ボタンをクリックします。
3. 画面上段にデプロイメント・プランのアンインストールが完了したというメッセージが表示されます。

コンソール・ツールの使用

コンソール・ツールで**uninstall-deployment-plan**コマンドを使用して、デプロイメント・プランをアンインストールできます。

```
[DAS]domain1.adminServer>uninstalldp plan1  
Uninstalling the deployment plan was successful.
```

参考

uninstall-deployment-planコマンドの使用方法的詳細については、『*JEUS リファレンスガイド*』の「4.2.6.17. uninstall-deployment-plan」を参照してください。

4.5.7. デプロイメント・プランと再デプロイ

デプロイメント・プランを利用してデプロイしたアプリケーションは、再デプロイ時にも既存のデプロイメント・プランがそのまま適用されるのが一般的ですが、新しいデプロイメント・プランを設定して適用することも可能です。

用語集

Console(コンソール)

GUIインターフェースと反対概念のコマンドライン・インターフェースです(ターミナル・ウィンドウ)。

DD

Deployment Descriptor(デプロイメント記述子)の略語です。

jeus-application-dd.xml

JEUSアプリケーションのデプロイメント記述子ファイルです。

JEUSMain.xml

JEUS WAS環境ファイルです。

索引

シンボル

<action>, 83
<application>
 <classloading>, 40
 <java-security-permission>, 39
 <library-ref>, 39
 <role-permission>, 39
<configuration>, 83
<descriptor>, 82
<value>, 83
<xpath>, 83

A

add-application-target, 59
Application Target, 11
application-info, 59
 Applicatio Type, 76
 Application ID, 76
 Application Path, 76
 Cluster Target, 76
 Server Target, 76
 state, 76
application.xml, 53

D

DD. *See* デプロイメント記述子
deploy-application, 59
distribute-application, 59

E

embedable EJB, 10
Explodedモジュール, 18

J

Java EEの標準DD, 39

JEUS DD, 39
jeus-application-dd.xml, 39
jeusadmin, 58
 application-info, 72
 Run-time Deploy, 58
 See also Run-time Deploy
JEUSアプリケーションのディレクトリー構造
 _appdat.ser, 10
 genディレクトリー, 10
 デプロイ・イメージ・ディレクトリー, 9
 ドメインで管理するアプリケーション・リポジトリ, 8
 ドメインのINSTALL_HOMEディレクトリー, 8

L

lib, 37
library directory, 37

M

META-INF, 38, 54

R

redeploy-application, 59
remove-application-target, 59

S

start-application, 59
stop-application, 59

U

undeploy-application, 59

あ

永久デプロイ, 13
アプリケーション, 35
 .ear archiveの構成, 37
 JEUSアプリケーションのディレクトリー構造, 8
アプリケーションのID, 5
アプリケーションのサービス対象, 11
 仮想ホスト, 13
 クラスター, 11
 サーバー, 11

アプリケーション状態
 DISTRIBUTED, 6
 INSTALLED, 6
 RUNNING, 6
インストール, 1

か

仮想ホスト, 13
クラスター, 11
グレースフル・アンデプロイ, 25
グレースフル再デプロイ, 27

さ

自動再デプロイ, 17
サーバー, 11
ステージング・モード・デプロイ, 76

た

2フェーズ・デプロイメント
 第1フェーズ(配布), 2
デプロイ, 2, 54
 デプロイメント記述子, 38
デプロイメント, 38
デプロイメント記述子, 38
 EJBモジュールDD, 38
 WebアプリケーションDD, 39
 WebサービスDD, 39
 アプリケーション・クライアントDD, 39
 アプリケーションDD, 38
 リソース・アダプターDD, 39
デプロイ作業
 アプリケーションのサービス対象の削除, 55
 アプリケーションのサービス対象の追加, 55
 アンデプロイ, 55
 開始, 55
 再デプロイ, 55
 停止, 55
 デプロイ, 54
 配布, 54
2フェーズ・デプロイメント
 第2フェーズ(開始), 4

は

ブートタイム・デプロイ, 14
 DEPENDENT状態でのブートタイム・デプロイ, 14
 INDEPENDENT状態でのブートタイム・デプロイ, 16

ま

モジュール, 35
 EJBモジュール, 36
 Webアプリケーション・モジュール, 36
 アプリケーション・クライアント・モジュール, 36
 リソース・アダプター・モジュール, 36

ら

ランタイム・デプロイ, 13