

# JEUS アプリケーションクライアントガイド

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 アプリケーションクライアントガイド

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
<b>第1章 アプリケーション・クライアント .....</b>	<b>1</b>
1.1. 概要 .....	1
1.2. プログラムの作成 .....	1
1.2.1. プログラムの構造 .....	1
1.2.2. 例 .....	2
1.3. デプロイメント記述子(DD) .....	3
1.3.1. DDの作成 .....	3
1.3.2. DDの生成 .....	4
1.4. パッケージング .....	5
1.5. デプロイ .....	5
1.6. 実行 .....	6
1.6.1. コンソールでの実行 .....	6
<b>第2章 高度なアプリケーション・クライアント .....</b>	<b>9</b>
2.1. 概要 .....	9
2.2. 依存性注入 .....	9
2.2.1. EJB参照の注入 .....	11
2.2.2. リソースの注入 .....	11
2.2.3. その他の注入 .....	12
2.3. 依存性注入を使用できないクライアント .....	12
2.4. セキュリティー設定 .....	14
2.5. トランザクション .....	15
<b>第3章 アプレット・クライアント .....</b>	<b>17</b>
3.1. 概要 .....	17
3.2. プログラムの作成 .....	17
3.2.1. 例 .....	17
3.2.2. HTMLの例 .....	18
3.3. デプロイ .....	19
3.4. 実行 .....	20
3.4.1. Webブラウザでの実行 .....	20
3.4.2. アプレット・ビューアでの実行 .....	20
<b>第4章 JNLPクライアント .....</b>	<b>21</b>
4.1. 概要 .....	21
4.2. プログラムの作成 .....	21
4.3. プログラムの実行 .....	24
4.3.1. クライアントの実行 .....	24
4.3.2. クライアント・コンテナでの実行 .....	24
<b>用語集 .....</b>	<b>27</b>



# 図目次

[図 1.1] アプリケーション・クライアント・アーキテクチャ .....	2
---------------------------------------	---





# 表目次

[表 1.1] サービス別に追加で必要なJEUSライブラリー .....	6
--------------------------------------	---



## 例目次

[例 1.1]	<<HelloClient.java>> .....	2
[例 1.2]	<<jeus-client-dd.xml>> .....	4
[例 2.1]	<<EJB参照の注入>> .....	10
[例 2.2]	<<リソースの注入>> .....	12
[例 2.3]	<<HelloClient.java>> .....	13
[例 2.4]	<<jeus-client-dd.xml>> .....	14
[例 2.5]	<<Client.java>> .....	14
[例 3.1]	<<HelloClient.java>> .....	17
[例 3.2]	<<index.html>> .....	19
[例 4.1]	<<JNLPサーブレット : web.xml>> .....	22
[例 4.2]	<<HelloClient.java>> .....	23
[例 4.3]	<<HelloClient.jnlp>> .....	24
[例 4.4]	<<HelloClient.java>> .....	25
[例 4.5]	<<HelloClient.jnlp>> .....	26



# このガイドについて

## 対象読者

本書は、JEUS<sup>®</sup>(以下、JEUS)ベースでアプリケーション・クライアントを開発する開発者を対象としています。同書では、JEUSでのアプリケーション・クライアントの概念とその使用方法、およびJNLPの概念について説明します。

アプリケーション・クライアントとは、以下のような実行可能な形式のプログラムのことをいいます。

- Javaアプリケーション・クライアント
- アプレット
- JNLPクライアント

---

### 参考

本書で使用した例は、JEUS\_HOME/samples/clientディレクトリーでソース全体を確認することが可能であり、かつ直接実行することもできます。

---

## 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 Webサービスガイド

Webサービスを使用するクライアントの場合は、Webサービスのための追加作業が必要です。

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<sup>™</sup>(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 アプリケーション・クライアント」](#)

JEUSサーバーとは別のJVMで実行されるアプリケーション・クライアントについて説明します。

- [「第2章 高度なアプリケーション・クライアント」](#)

クライアント・コンテナなしで動作するクライアントとアプリケーション・クライアントが利用できるサービスについて説明します。

- [「第3章 アプレット・クライアント」](#)

JEUSでのアプレット・プログラムの作成、設定、および実行方法について説明します。

- [「第4章 JNLPクライアント」](#)

クライアントと、これに関するJNLPファイルをWebアプリケーションにデプロイする方法と、実際にユーザーがクライアントを実行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
<b>参考</b>	参照/注意事項
<b>注</b>	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 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 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 Webサービスガイド	JEUSでのWebサービス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
JEUS セキュリティガイド	JEUSでのセキュリティー・システムの設定と運用方法およびセキュリティー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 & デプロイメントガイド	Java EEアプリケーションをJEUSにデプロイするための様々な方法とツールについて記述しています

## 参考文献

- The Java EE 7 Technologies(<http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-142185.html>)
- The JNLP Specification
- The Java WebStart Specification

## 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](mailto: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](mailto:info@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/us\\_en/home](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](mailto: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](mailto:info@tmaxsoft.com.cn)  
Web (Chinese): [http://www.tmaxsoft.com/cn\\_en/home\\_cn\\_en](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](mailto:info.bra@tmaxsoft.com)  
Web (Portuguese): [http://www.tmaxsoft.com/br\\_en/home\\_br\\_en](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](mailto:info.rus@tmaxsoft.com)  
Web (Russian): [http://www.tmaxsoft.com/ru\\_ru/home\\_ru\\_ru](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](mailto:info.sg@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/sg\\_en/home\\_sg\\_en](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](mailto:info.uk@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/gb\\_en/home\\_gb\\_en](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](mailto:info.canada@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/ca\\_en/home\\_ca\\_en](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](mailto:info.aus@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/au\\_en/home\\_au\\_en](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](mailto:info.india@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/in\\_en/home\\_in\\_en](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](mailto:cslee@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/tr\\_en/home\\_tr\\_en](http://www.tmaxsoft.com/tr_en/home_tr_en)



# 第1章 アプリケーション・クライアント

本章では、JEUSサーバーとは別のJVMで実行されるアプリケーション・クライアントについて説明します。

## 1.1. 概要

一般的に、Java EEアプリケーションを呼び出したり、Java EEのサービスの提供を受けたりするためには、JEUSのクライアント・コンテナで駆動するアプリケーション・クライアントのモジュールを使用します。

アプリケーション・クライアントは、Java EE環境を使用するスタンドアロン・クライアントで、Java EEの仕様に定義されているアプリケーション・クライアント・コンテナで駆動するアプリケーションです。アプリケーション・クライアント・モジュールはJEUSクライアントの一形態です。これは、クライアントまたはサーバー・システムや、テストおよびデバッグ時に有用なクライアントです。

JEUSアプリケーション・クライアントは、クライアント・コンテナを使用して、ネーミング・サービス、スケジューラー、セキュリティなどのJEUSサービスを利用することができます。クライアント・コンテナを使用しない場合にも、JEUSクライアント・ライブラリーを用いることで、JNDI、セキュリティなどJava EEサービスの一部を利用できますが、依存性注入、JEUSスケジューラーなどのサービスは利用できません。

---

### 参考

1. Java EEベースのアプリケーション・クライアントの詳細については、Java EEの仕様を参照してください。
  2. JEUS XMLスキーマの詳細については、『JEUS XMLリファレンスガイド』の「jeus-client-dd.xml」設定を参照してください。
- 

## 1.2. プログラムの作成

本節では、JEUSクライアントとサーバー環境のアーキテクチャについて説明した後、簡単なサンプル・コードを作成します。

### 1.2.1. プログラムの構造

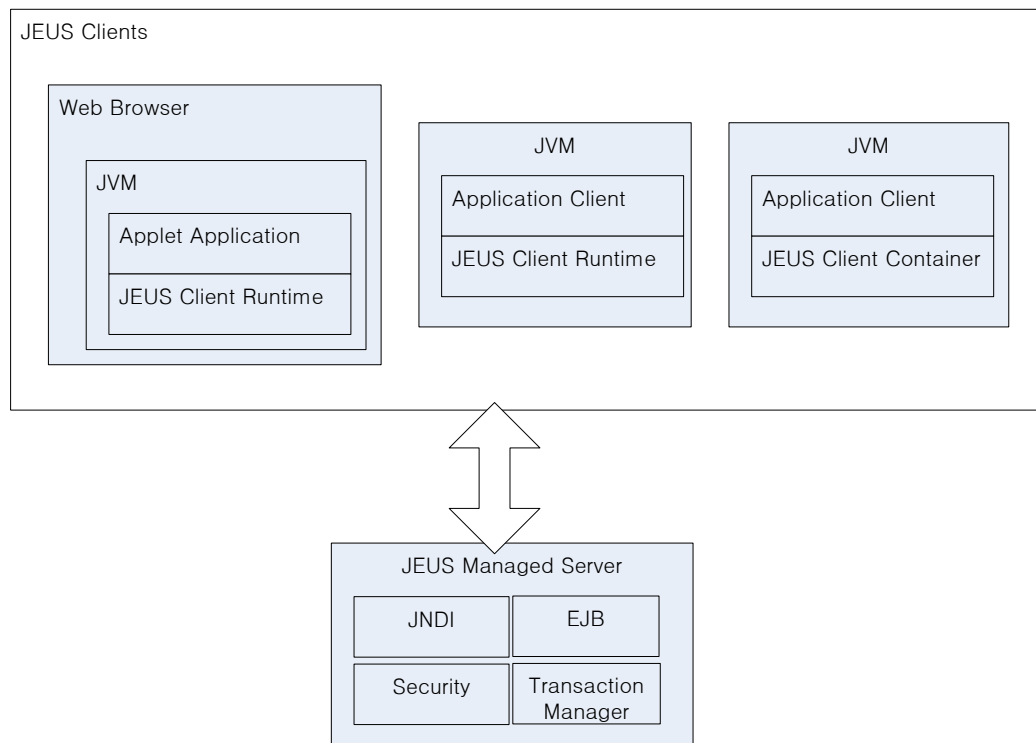
アプリケーション・クライアントは、サーバーとは別のJVMで実行されるクライアント・プログラムです。

アプリケーション・クライアントは、main()メソッドを呼び出して実行し、仮想マシンの終了と共に終了します。他のJava EEアプリケーション・コンポーネントと同様に、アプリケーション・クライアントはシステム・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コンテナに比べてシ

システム・リソースの使用量が少ないです。また、JEUSが提供するJava EEサービスを一般クライアント・アプリケーションで利用できる環境を提供します。

このサービスには、JNDI、スケジューラー、セキュリティ、そしてJNDIサービスによってJEUSにバインディングされているアプリケーション・コンポーネント(EJB)およびシステム・リソース(JDBCデータ・ソース、JMSコネクション・ファクトリーなど)の使用が含まれます。

**[図 1.1] アプリケーション・クライアント・アーキテクチャ**



## 1.2.2. 例

アプリケーション・クライアントは、一般的なJavaプログラムと同様に、publicで宣言されたmain()メソッドを保持する必要があります。

以下は、Java EE 7のスタイルで作成された例で、注入によってEJBを呼び出します。(JEUS\_HOME/samples/client/hello/hello-clientの例を参照)

**[例 1.1] <<HelloClient.java>>**

```
package helloejb;

import java.io.*;
import javax.ejb.EJB;

public class HelloClient {
```



```
@EJB(mappedName="helloejb.Hello")
private static Hello hello;

public static void main(String[] args) {
    System.out.println("EJB output : " + hello.sayHello());
}
}
```

---

#### 参考

Java EEアプリケーション・クライアント・プログラミングの詳細については、Java EE 7 Technologies(<http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-142185.html>)を参照してください。

---

## 1.3. デプロイメント記述子(DD)

本節では、デプロイメント記述子(Deployment Descriptor、以下DD)の作成方法について説明します。

### 1.3.1. DDの作成

DDの作成方法は以下の2つに分けられます。

- Java EE DD

Java EEの仕様にはクライアント・モジュールについてのDDが定義されていますが、これはWeb Application Server(WAS)に関係のない標準的な設定を含んでいます。

Java EE 5からは、従来とは異なり、標準DDファイルがなくてもアノテーションを用いて設定を行うことができます。したがって、上記の[例 1.1]もapplication-client.xmlが別途なくても動作します。

標準XMLディスクリプタの詳細については、Java EEの仕様を参照してください。

- JEUS DD

サーバーとアプリケーション・クライアントの通信に必要なクライアント・モジュールの情報を保持しているXML設定です。

クライアントのためのDDはMETA-INF/jeus-client-dd.xmlです。DDを使用してアプリケーション・クライアントをクライアント・コンテナにデプロイする際にどのサービスを使用するのかを指定できます。また、アプリケーション・クライアントを実行する場合も、DDを修正するだけで該当アプリケーション・クライアントに適用できます。

一般的にクライアントが使用するリソースなどを指定するために使用しますが、上の[例 1.1]ではこのDDが不要であるため、省略しました。

---

## 参考

jeus-client-dd.xmlの詳細については『JEUS XMLリファレンスガイド』を参照してください。

---

### 1.3.2. DDの生成

DDを生成する際、必要な場合はJEUSでクライアント・モジュールをデプロイする前にjeus-client-dd.xmlファイルを作成します。

以下は、DDを作成したXMLファイルの例です。

XMLでクライアントが使用する環境変数、EJBアプリケーションがバインディングされた名前、バインディングされたJDBCデータ・ソースの名前、JMSキューのJNDIバインディングの名前を指定しています。環境変数以外は、application-client.xmlの<ref-name>に実際にバインディングされたJNDIの名前を<export-name>に指定します。

#### [例 1.2] <<jeus-client-dd.xml>>

```
<jeus-client-dd>
  <env>
    <name>year</name>
    <type>java.lang.Integer</type>
    <value>2002</value>
  </env>
  <ejb-ref>
    <jndi-info>
      <ref-name>count</ref-name>
      <export-name>count_bean</export-name>
    </jndi-info>
  </ejb-ref>
  <res-ref>
    <jndi-info>
      <ref-name>datasource</ref-name>
      <export-name>Oracle_DataSource</export-name>
    </jndi-info>
  </res-ref>
  <res-env-ref>
    <jndi-info>
      <ref-name>jms/SalaryInfo</ref-name>
      <export-name>jms/salary_info_queue1</export-name>
    </jndi-info>
  </res-env-ref>
</jeus-client-dd>
```

---

## 参考

jeus-client-dd.xmlファイルの各要素の詳細については、『JEUS XMLリファレンスガイド』を参照してください。

---

## 1.4. パッケージング

クライアント・モジュールのパッケージングは、手動パッケージング方式とIDEを使用したパッケージング方式に分けられます。

### ● 手動パッケージング

必要な場合は、ユーザーのコンピュータにインストールされているテキスト・エディタまたはXMLエディタを利用してDD XMLファイルを作成し、必要なファイルを集めて、Javaが提供するJARツールでクライアント・モジュールのJARファイルを作成します。

アプリケーション・クライアントをパッケージングするには、アプリケーションを構成しているクラス・ファイルと、必要な場合は**application-client.xml**、**jeus-client-dd.xml**ファイルが含まれている必要があります。

コンソールでは**jar**コマンドを用いてクライアント・モジュールのJARファイルを作成します。標準仕様では、JARファイルのMANIFEST.MFに、このJARファイルの実行時に使用するメインクラスを指定できます。この場合、JEUSクライアント・コンテナはメインクラスを知らなくても自動的にこのクラスを実行します。

以下は、**jar**コマンドを用いてJARファイルを生成する例です。

```
jar cvf hello-client.jar *
```

### ● IDEを使用したパッケージング

EclipseやNetBeans、IntelliJ IDEAなどのJava EE環境に対応するIDEツールを用いて作成します。この方法については、各IDEのヘルプを参照してください。

## 1.5. デプロイ

アプリケーション・クライアント・モジュールのデプロイは手動あるいはJEUS WebAdminを使用して行います。

モジュールの中には、必要に応じて、Java EEの標準DDファイルである**application-client.xml**とJEUSが提供する**jeus-client-dd.xml**が含まれています。アプリケーション・クライアントのモジュール・ファイルを作成後、該当ファイルを適切な場所に移動させます。

Webサービス・クライアントとして動作する機能がある場合、追加でコンソール・ツール(**jeusadmin**)やWebAdminを用いてデプロイするか、**appcompiler**を使用しなければ、Webサービスのスタブが作成されません。

---

## 参考

1. WebAdminの使用方法については、『JEUS WebAdminガイド』を参照してください。
  2. デプロイの詳細については、『JEUS アプリケーション & デプロイメントガイド』を参照してください。
- 

## 1.6. 実行

本節では、サービス別に追加で必要なJEUSライブラリーと、コンソールでモジュールを実行する方法について説明します。

アプリケーション・クライアントでWebサービスを使用する場合、ライブラリーの実行段階で基本ライブラリーのJEUS\_HOME/lib/client/clientcontainer.jar以外にも必要なライブラリーが存在します。これは、大部分がJEUS\_HOME/lib/system(以下、SYSTEM\_LIB\_DIR)の下に位置します。

以下は、追加で必要なJEUSライブラリーのサービス別一覧です。

**[表 1.1] サービス別に追加で必要なJEUSライブラリー**

サービス	JEUSライブラリー
JMS(Java Message Service)	– SYSTEM_LIB_DIR/jms.jar
Web Service	– SYSTEM_LIB_DIR/mail.jar – SYSTEM_LIB_DIR/jeus-ws.jar – JEUS_HOME/lib/shared/wsit-2.3/webservices-rt.jar – SYSTEM_LIB_DIR/resolver.jar
JMX(Java Management eXtensions)	– SYSTEM_LIB_DIR/jmxremote.jar

---

## 参考

Webサービスの詳細については、『JEUS Webサービスガイド』を参照してください。

---

### 1.6.1. コンソールでの実行

コンソールでモジュールを実行するためには**appclient**を使用します。appclientはJEUS\_HOME/binに存在するスクリプトであり、クライアント・コンテナを使用してアプリケーション・クライアント・モジュールを実行します。

以下は、JEUSで提供するクライアント・コンテナのコマンドライン形式です。

- 使用法

```
appclient -client client_jar_path
          [-main main_class]
          [-cp classpath]
          application_arguments...
```

以下は、コマンド・オプションについての説明です。

オプション	説明
-client client_jar_path	実行するアプリケーション・クライアントのパスを指定します
[-main main_class]	アプリケーション・クライアントのメインクラスを指定します。クライアントのクラスパスに指定されたパスでMETA-INF/MANIFEST.MFの設定情報にメインクラスが指定されている場合、このオプションを使用する必要はありません
[-cp classpath]	必要な場合、クライアントの実行に必要なクラスパスを指定します

- 例

上記例を実行すると、以下のようになります。アプリケーション・クライアントが正常に実行されるためには、Hello EJBがデプロイされている必要があります。

```
JEUS_HOME/samples/client/hello/hello-client/dist$ appclient -client
hello-client.jar
[2016.08.03 14:44:46][0] [t-1] [CLIENT-0050] Starting the Application Client
Container - JEUS 8.0 (Fix#0)
EJB output : Hello EJB!
```

上記の内容のうち、JEUSログを出力しない場合は、以下の設定を追加で行います。

```
-Djeus.log.level=OFF
```

---

## 参考

ログ設定の詳細については、『JEUS サーバガイド』の「第8章 ロギング」を参照してください。

---



## 第2章 高度なアプリケーション・クライアント

本章では、前章で説明したアプリケーション・クライアントより高度なモジュール実装について説明します。

### 2.1. 概要

Java EEの仕様で提示するクライアント・コンテナを使用しなくても、一般Javaクラスを実行するように簡単にJEUSクライアントを実行できます。クライアント・コンテナが依存性注入(Dependency Injection)を行うリソースの種類とデフォルトのJNDIバインディングの名前について説明します。また、クライアント・コンテナなしで動作するクライアントとアプリケーション・クライアントが使用できるサービス(セキュリティの設定、トランザクション)について説明します。

クライアント・コンテナを使用せずにJEUSクライアントを実行する場合、コンテナの依存性注入サービスを受けられないことに注意してください。また、セキュリティ設定を行うことで、JEUSが提供する様々なサービスを実行できます。なお、トランザクション機能を使用すると、クライアントが使用したアプリケーションとリソースをグローバル・トランザクションで管理できます。

### 2.2. 依存性注入

本節で説明する注入についての内容は、アプリケーション・クライアント以外にも、WebアプリケーションやEJBアプリケーションなどにすべて適用されます。

注入できるリソースは、EJBオブジェクトと、JNDIでマッピングできる環境変数などがあります。基本的には、アプリケーション・コンポーネントのJNDIコンテキストであるjava:comp/envコンテキストで指定された名前を検索します。実際のリソースはJNDIグローバル・コンテキストにバインディングされているため、そのグローバル・バインディングの名前を把握している必要があります。

リソースは、独自のJNDIグローバル・バインディング名を持っています。EJBアプリケーションを例にすると、この名前は下記のいずれかの方法で設定されている必要があります。

- JEUSで認識するjeus-ejb-dd.xmlの<export-name>を使用
- 標準にあるejb-jar.xmlの<mapped-name>を使用
- EJBアプリケーションに指定されているアノテーションのmappedNameを使用
- 『JEUS EJBガイド』で提示されているJEUSのデフォルトのJNDI名でEJBがバインディング

クライアント・コンテナがない環境などでJNDIを直接使用してEJBを取得する場合は、デフォルトのJNDI名が決定されるルールを知る必要があります。このように、どの名前でバインディングされるかを開発者が確認

することが困難であるため、実際の開発では上記のいずれかの方法でJNDIバインディング名を明示するのが一般的です。

リソースを注入する側では、JEUS自体のDDであるjeus-ejb-dd.xml、jeus-web-dd.xml、jeus-client-dd.xmlなどで注入に使用するJNDIグローバル・バインディング名を指定するか、ejb-jar.xml、web.xml、application-client.xmlなどの標準DDのmapped-nameまたはアノテーションのmappedNameの指定値を使用してマッピングします。いずれの値も指定されていない場合は、JEUSの基本ルールに従った名前での注入を試行します。

実際に開発する際には、アノテーションのmappedNameでJNDIグローバル・バインディング名を指定するより、XMLを使用して指定することを推奨します。特に複数の場所で運用されるアプリケーションの場合は、その環境に適するグローバル名を使用する必要があるため、XMLを使用します。

注入はアノテーションをした変数とセッター・メソッドに対して行われますが、アノテーションをしない場合もXMLディスクリプタで指定した変数とセッター・メソッドに対して注入を行うことができます。

---

#### 参考

1. 注入の詳細については、Java EE 7 Technologies(<http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-142185.html>)を参照し、注入可能なリソースの詳細については、当該ガイドの「5. Resources, Naming and Injection」を参照してください。
  2. EJBアプリケーションのEJBコンテキスト注入などは『JEUS EJBガイド』を参照してください。
- 

以下は、上記の3つの方法のうち、アノテーションのmappedNameを使用したEJBアプリケーションをクライアントで注入する例です。

StatelessEJB1アプリケーションがJNDIグローバル・バインディング名に「MyEJB1」を使用しているので、クライアントでは@EJBアノテーションのmappedNameに同じ名前を指定します。また、クライアントで注入の代わりにJNDIルックアップを使用する場合、JNDIグローバル・バインディング名でそのままルックアップするか、クライアント・コンテナなどで動作するクライアントの場合はapplication-client.xmlなどを使用してアプリケーション・コンテキストに登録されているjava:comp/env/ejb/sless1という名前を使用できます。

#### [例 2.1] <<EJB参照の注入>>

```
import ejbl.RemoteSession;

@Stateless(name="StatelessEJB1", mappedName="MyEJB1")
public class StatelessEJB1 implements RemoteSession, LocalSession {...
}

...
@EJB(name="sless1", beanName="StatelessEJB1", mappedName="MyEJB1")
private RemoteSession sless1;

...
RemoteSession session = context.lookup("MyEJB1");
```



```
...
// with client container and application-client.xml descriptor
RemoteSession session = context.lookup("java:comp/env/ejb/sless1");
```

## 2.2.1. EJB参照の注入

EJB参照を注入するには、以下のようなバインディング名を使用します

- **変数タイプがビジネス・インターフェースの場合**

mappedNameが存在する場合、ルックアップする際に使用するグローバル名を以下の形式で設定します。

```
mappedName + "#" + Business_Interface_Name
```

上記例の場合、「MyEJB1」または「MyEJB1#ejb1.RemoteSession」を使用します。

EARやEJB JARなどでデプロイされた場合、ejb-jar.xmlにejb-linkが指定されているか、アノテーションにbeanNameがあるときは、同じアプリケーション内のEJBを探し、そのEJBのmappedNameをグローバル名として使用してルックアップします。このような情報がない場合は、同じアプリケーション内でビジネス・インターフェース名でEJBを探し、そのEJBのmappedNameをグローバル名として使用します。

最後に、ビジネス・インターフェース名でJNDIルックアップします。上記例の場合、「java:global/<module-name>/MyEJB1」または「java:global/<module-name>/MyEJB1!ejb1.RemoteSession」という名前でルックアップします。この方法は、EJBのデプロイ時にmappedNameがない場合にデフォルトのバインディング名を使用する方法です。

- **変数タイプがEJBHome/EJBObjectインターフェースの下位インターフェースの場合**

mappedNameが存在する場合、**mappedName**をグローバル名に使用します。

EARやEJB JARなどでデプロイされた場合、ejb-jar.xmlにEJB-linkが指定されているか、アノテーションにbeanNameがあるときは、同じアプリケーション内のEJBを探し、そのEJBのmappedNameを使用してルックアップします。このような情報がない場合、同じアプリケーション内で変数タイプのインターフェース名でEJBを探し、そのEJBのmappedNameをグローバル名として使用します。

最後に、ビジネス・インターフェース名でJNDIルックアップします。

## 2.2.2. リソースの注入

リソースを注入するには@Resourceアノテーションを使用します。

- mappedNameが指定されている場合、この名前をリソースのJNDIグローバル・バインディング名としてルックアップします。

- mappedNameが指定されていない場合、@Resourceのname属性値をJNDIグローバル・バインディング名として使用します。

name属性値が指定されていない場合は、仕様に従って、以下の形式で使用されます。

アプリケーション・クラスの名前 + / + 変数またはセッター・メソッドのプロパティ名

以下の例では、「jdbc/DB2名」でJNDIルックアップを実行します。name属性が指定されていない場合は、「test.Client/myDataSource3」でルックアップします。

#### [例 2.2] <<リソースの注入>>

```
package test;

class Client {
    @Resource(name="jdbc/DB2") // default mapping if no mapped-name private
    javax.sql.DataSource myDataSource3;
    ...
}
```

---

#### 参考

nameが存在する場合、あるいはname属性が指定されていない場合のデフォルト値は仕様で定められています。この値はアプリケーション・コンテキストにマッピングされている名前、実際のJNDIグローバル・バインディング名はベンダーごとにルールが異なります。したがって、互換性のためにはmappedNameなどを使用することを推奨します。

---

### 2.2.3. その他の注入

その他にも、@WebServiceRef、@PersistenceUnit、@PersistenceContextなどのアノテーションを使用して、Webサービス・オブジェクト、EntityManagerオブジェクト、EntityManagerFactoryオブジェクトなどを注入によって取得できます。

---

#### 参考

その他の詳細については、Java EE 7 Technologies(<http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-142185.html>)を参照してください。

---

## 2.3. 依存性注入を使用できないクライアント

アプリケーション・コンテナを使用しなくても、JEUSのJNDIなどを通じてJEUSのリソースとアプリケーションを使用することができます。この場合、JEUS\_HOME/lib/clientのjclient.jarファイルをクラスパスに指定して作成したクライアントを実行します。

しかし、この場合は依存性注入は使用できないため、以下のようにソースを変更して使用する必要があります。例では、EJBアプリケーションのバインディング名にJEUSのデフォルトのバインディング名のルールに従った名前を使用しています。このクライアントは、クライアント・コンテナ上でも動作します。つまり、クライアント・コンテナを使用しないすべてのクライアントはクライアント・コンテナ上でも動作します。

**[例 2.3] <<HelloClient.java>>**

```
package helloejb;

import java.io.*;
import javax.ejb.EJB;

import javax.naming.Context;
import javax.naming.InitialContext;
import java.util.Hashtable;

/**
 * HelloEJB application client
 */
public class HelloClient {
    private static Hello hello;

    public static void main(String[] args) {
        try {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY, "jeus.jndi.JNSContextFactory");

            Context context = new InitialContext(env);
            hello = (Hello) context.lookup("helloejb.Hello");

            System.out.println("EJB output : " + hello.sayHello());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

クライアント・パッケージングは、XMLファイル进行处理するクライアント・コンテナで実行しないため、標準、JEUS XMLファイルなしで、JARファイルで作成します。デプロイ時には、同様に適切な場所にJARファイルをコピーします。実行時には一般のJavaクラスを実行するときと同じく、上のクライアント・クラスを実行します。

実行結果は、以下のとおりです。

```
$ java -cp /jeus/lib/client/jclient.jar;./hello-client.jar helloejb.HelloClient
[2012.05.23 22:45:51][2] [t-1] [Network-0405] [Endpoint] exporting Endpoint
(0:192.168.0.16:9756;-1:0x79F24F28)
[2012.05.23 22:45:51][2] [t-1] [Network-0002] [Acceptor] start to listen
```

```
NonBlockingChannelAcceptor: /192.168.0.16:9756
EJB output : Hello EJB!
```

## 2.4. セキュリティー設定

クライアントでは、JEUSが提供するJava EEのサービスを使用する際、そのサービスに対する使用権限の有無を確認できるように、クライアントのユーザー名とパスワードをクライアント・ランタイムに指定する必要があります。このサービスには、EJBアプリケーション、JMSリソースなどがあります。これを指定する方法には以下の3つがあります。

- **jeus-client-dd.xmlを使用する方法**

jeus-client-dd.xmlの<security-info>を指定すると、クライアント・コンテナでアプリケーションを実行する前に、指定されたユーザー名とパスワードでログインします。以降、アプリケーションではJEUSのサービスを使用する際にこのユーザー名で認証を試みます。jeus-client-dd.xml設定の詳細については『JEUS XMLリファレンスガイド』を参照してください。

**[例 2.4] <<jeus-client-dd.xml>>**

```
<jeus-client-dd>
...
  <security-info>
    <provider-node-name>jeusNode</provider-node-name>
    <user>user1</user>
    <passwd>password1</passwd>
  </security-info>
...
</jeus-client-dd>
```

- **JNDIコンテキストを使用する方法**

アプリケーションでJNDIコンテキストを作成する際に、JNDIのプロパティを使用して、ユーザー名とパスワードでログインできます。以降、ログインしたユーザー名で認証が行われます。この方法はクライアント・コンテナを使用しない場合にも利用できます。JNDIの設定については、『JEUS サーバガイド』の「第4章 JNDIネーミング・サーバー」を参照してください。

**[例 2.5] <<Client.java>>**

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "jeus.jndi.JNSContextFactory");
env.put(Context.PROVIDER_URL, "192.168.0.16:9736");
env.put(Context.SECURITY_PRINCIPAL, "user1");
env.put(Context.SECURITY_CREDENTIALS, "password1");
Context context = new InitialContext(env);
```

- JEUSのセキュリティーAPIを直接使用する方法

JEUSのセキュリティーAPIを使用してログインできます。セキュリティーAPI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。

## 2.5. トランザクション

クライアントで使用するEJBアプリケーション、JDBCデータ・ソース、JMSコネクション・ファクトリーとデスティネーションなどを1つのグローバル・トランザクション、あるいはXAトランザクションとして使用するには、UserTransactionを使用します。

クライアントで使用するトランザクション・マネージャーは、リソースを直接管理するか否かによって、サーバー・トランザクション・マネージャーとクライアント・トランザクション・マネージャーに分けることができます。

---

### 参考

1. UserTransactionの詳細については、Java EE 7 Technologies(<http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-142185.html>)を参照してください。
  2. トランザクション・マネージャーの詳細については、『JEUS サーバガイド』の「第7章トランザクション・マネージャー」を参照してください。
-



## 第3章 アプレット・クライアント

本章では、JEUSでアプレット・プログラムを作成、設定、および実行する方法について説明します。

### 3.1. 概要

アプレットは、Webブラウザで実行されるJavaアプリケーションです。

Webブラウザで実行されるアプレットでJEUSのJava EEサービスを使用するためには、アプレット・コンテナを使用します。アプレットは基本的にアプリケーション・クライアントですが、JEUSでは軽量クライアント・コンテナをまだ提供していないため、JEUSのライブラリーに直接アクセスできないアプレットはJEUSのJava EEサービスを使用する方法で実装します。

アプレットを使用すると、ブラウザでJavaアプリケーションを実行できます。そして、JEUSの機能を使用するクライアントとして動作できます。

### 3.2. プログラムの作成

アプレットが駆動するのに必要なファイルは、Webアプリケーション内に存在します。HTMLのJAVA\_CODEBASEが「.」となっているため、このJARファイルはHTML文書がデPLOYされるWebアプリケーションでHTML文書と同じディレクトリーに存在する必要があります。

本節では、ユーザーが参考にできるサンプル例について説明します。

#### 3.2.1. 例

アプレット・アプリケーションはAppletまたはJAppletクラスを継承し、start()メソッドを実装する必要があります。

以下は、クライアント・コンテナを介さず、独立して実行されるクライアントの例です。したがって、依存性注入を使用せずにJNDI APIを用いて該当EJBを直接ルックアップします。

クライアント・コンテナと同じEJBを使用する場合、EJBはhelloejb.Helloというインターフェイス名をバインディング名として使用します。

**[例 3.1] <<HelloClient.java>>**

```
package helloejb;

import javax.naming.Context;
```

```
import javax.naming.InitialContext;
import java.applet.Applet;
import java.util.Hashtable;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;

import javax.swing.*;

public class HelloClient extends JApplet {
    public void start() {
        try {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY, "jeus.jndi.JNSContextFactory");

            Context context = new InitialContext(env);
            Hello hello = (Hello) context.lookup("helloejb.Hello");
            System.out.println("EJB output : " + hello.sayHello());

            JLabel label = new JLabel(hello.sayHello());
            label.setFont(new Font("Helvetica", Font.BOLD, 15));
            getContentPane().setLayout(new BorderLayout());
            getContentPane().add(label, BorderLayout.CENTER);
            setSize(500, 250);
            setVisible(true);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

---

#### 参考

JNDIにバインドされるEJBの名前についての詳細内容は、『JEUS EJBガイド』を参照してください。

---

### 3.2.2. HTMLの例

HTML文書には、呼び出すアプレットとそのアプレットのクラスが存在する位置を指定します。

例では、上のアプリケーション・クラスとhelloejb.HelloのEJBインターフェイスがhello-client.jarに含まれていると仮定しています。jclient.jarはJEUS\_HOME/lib/clientに存在するクライアント用のJEUSライブラリーです。



### [例 3.2] <<index.html>>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Hello JavaEE</title>
</head>
<body>
<center>
<h1>Hello JavaEE Sample Applet!</h1>
<APPLET CODE = "helloejb.HelloClient" JAVA_CODEBASE = "."
    ARCHIVE = "hello-client.jar,jclient.jar"
    WIDTH = 300 HEIGHT = 300/>
</APPLET>
</center>
</body>
</html>
```

---

#### 参考

HTML文書をブラウザに関係なく使用し、さらにブラウザが実行される場所にJDKがインストールされていない場合は、JDKをインストールするように誘導できます。HTML文書に変換するには、JDKのhtmlconverterを使用します。詳細については、[http://docs.oracle.com/javase/1.5.0/docs/guide/plugin/developer\\_guide/html\\_converter.html](http://docs.oracle.com/javase/1.5.0/docs/guide/plugin/developer_guide/html_converter.html)を参照してください。

---

## 3.3. デプロイ

アプレットは基本的にHTML文書からアクセスするため、このHTML文書をWebに送信するためにはWebアプリケーションが必要です。アプレットを実行前にWebアプリケーションとEJBのデプロイが完了している必要があります。

#### ● Webアプリケーションのデプロイ

Webアプリケーションを作成し、HTML文書および「ARCHIVE」に指定されているJARファイルを追加します。Webアプリケーションの作成とデプロイについての詳細は『JEUS アプリケーション&デプロイメントガイド』を参照してください。

#### ● EJBのデプロイ

Webアプリケーションのデプロイが完了すると、EJBアプリケーションをデプロイします。

## 3.4. 実行

Webブラウザを開いてHTML文書のURLを入力し、アプレットを実行します。アプレットの場合、Javaセキュリティモデルに従ってjava.policyに設定されたとおり、アクセスをコントロールします。したがって、アプレットを正常に実行するには、java.policyファイルにアプレットで使用するクラスへのパーミッションを提供する必要があります。

---

### 参考

java.policyの設定は、<http://docs.oracle.com/javase/1.5.0/docs/guide/deployment/deployment-guide/security.html> 文書を参照してください。

---

### 3.4.1. Webブラウザでの実行

Webブラウザは、HTMLページ内にある<applet>タグにアクセスします。例のアプレット画面はスイングを使用しているため、Webブラウザで以下のアドレスにアクセスできます。

```
http://host1:8088/hello/index.html
```

### 3.4.2. アプレット・ビューアでの実行

テスト過程では、ブラウザを介さずにJDKに含まれているアプレット・ビューアを用いてアプレットを実行することができます。これにより、例外などを即刻確認できるため、ブラウザでテストするよりも簡単に開発できます。

```
appletviewer index.html
```

## 第4章 JNLPクライアント

本章では、クライアントとこれについてのJNLPファイルをWebアプリケーションにデプロイする方法と、実際にユーザーがクライアントを実行する方法について説明します。

### 4.1. 概要

アプリケーションのサイズが大きく、バージョンが頻繁に変更される場合、ソフトウェア・コンポーネントのデプロイおよび実行に関するプロトコルであるJNLP(Java Network Launching Protocol)を使用します。このプロトコルを使用して、Webサーバーを通じてクライアント・アプリケーション(以下、クライアント)を自動的にダウンロードして実行できます。これをJava Web Startといいます。JNLPを通じてダウンロードしたクライアントもJEUSを使用できます。

JNLPを使用すると、簡単にクライアント・プログラムをデプロイすることができます。バージョンアップにより、必要なバイナリを自動的に取得できるので、継続的なデプロイが可能です。また、JDKホーム・ディレクトリーのsample/jnlpには様々なJNLPサンプルが存在しており、このディレクトリー下のREADMEファイルに従って実行すると、JNLPを簡単に練習することができます。

---

#### 参考

本書では、JNLPについての詳細情報は記述していません。そのため、JNLPに関してある程度の事前知識が必要です。JNLPを使用してJARファイルをデプロイする際は、Javaパーミッションの問題が常に起きるため、大抵は署名を行います。JEUSで提供するjclient.jar、clientcontainer.jarなどのファイルは署名されていないので、必ず他のJARファイルと共に署名してください。JNLPの詳細については、<http://docs.oracle.com/javase/7/docs/technotes/guides/javaws>を参照してください。

---

### 4.2. プログラムの作成

本節では、Webアプリケーションの構成方法について説明します。

#### Webアプリケーションの構成

JNLPファイルをWeb上で取得し、ファイルに定義されているJARファイルをWebサーバーからダウンロードするには、JNLPプロトコルを実装したサーブレットが必要です。JEUSではこれを直接提供していないので、JDK 7またはJDK 8が提供するサンプルJNLPサーブレットを使用します。

サーブレットを実装したjnlp-servlet.jarファイルは、JDK 7を基準にして以下のディレクトリーに位置します。  
(IBM JDK 7はlibディレクトリー)

```
sample/jnlp/servlet
```

jnlp-servlet.jarファイルをWebアプリケーションのWEB-INF/libにコピーし、以下のようにweb.xmlを作成すると、サーブレットがJNLPプロトコルによってJNLPファイルとリソース・ファイルをクライアントに送信します。

---

#### 参考

JNLPプロトコルによってリソース・ファイルを送信する「JnlpDownloadServlet」の詳細については、  
<https://docs.oracle.com/javase/7/docs/technotes/guides/javaws/developersguide/downloadServletguide.html>  
を参照してください。

---

以下は、JNLPサーブレットのためのweb.xmlの例です。

#### [例 4.1] <<JNLPサーブレット : web.xml>>

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <servlet-name>JnlpDownloadServlet</servlet-name>
    <servlet-class>jnlp.sample.servlet.JnlpDownloadServlet</servlet-class>
    <init-param>
      <param-name>logLevel</param-name>
      <param-value>DEBUG</param-value>
    </init-param>
    <init-param>
      <param-name>logPath</param-name>
      <param-value>jnlpdownloadServlet.log</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>JnlpDownloadServlet</servlet-name>
    <url-pattern>*.jnlp</url-pattern>
  </servlet-mapping>
</web-app>
```

以下は、HelloClientクラスのソースです。JNLPクライアントは一般クライアントと同一です。JNLPクライアントはコンソール画面がないため、ここではアプレットと同じくスイングを使用してGUI画面を作成しました。

**[例 4.2] <<HelloClient.java>>**

```
package helloejb;

import javax.naming.Context;
import javax.naming.InitialContext;
import java.util.Hashtable;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;

import javax.swing.*;

public class HelloClient extends JFrame {
    public static void main(String[] args)
    {
        new HelloClient();
    }

    public HelloClient() {
        try {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY, "jeus.jndi.JNSContextFactory");

            Context context = new InitialContext(env);
            Hello hello = (Hello) context.lookup("helloejb.Hello");

            JLabel label = new JLabel(hello.sayHello());
            label.setFont(new Font("Helvetica", Font.BOLD, 15));
            getContentPane().setLayout(new BorderLayout());
            getContentPane().add(label, BorderLayout.CENTER);
            setSize(500, 250);
            setVisible(true);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Java Web Startは、HelloClientを実行するための情報が格納されているJNLPファイルを要求します。

以下の例では、HelloClientを実行するために必要なJARファイルと、使用するJavaバージョンなどが明示されています。<jnlp>タグでcodebase属性の「`$$codebase`」は、JNLPサーブレットを含むWebアプリケーションがデプロイされるコンテキストによって自動的に入れ替えられます。ここではJARファイルのhref値にパスが指定されていないため、JNLPファイルとJARファイルはどちらも同じディレクトリーに集めておく必要があります。

#### [例 4.3] <<HelloClient.jnlp>>

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0" codebase="$$codebase">
  <information>
    <title>HelloClient</title>
    <vendor>TmaxSoft</vendor>
  </information>
  <resources>
    <j2se version="1.5+" href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="hello-client.jar"/>
    <jar href="jclient.jar"/>
  </resources>
  <application-desc main-class="helloejb.HelloClient"/>
</jnlp>
```

---

#### 参考

「\$\$codebase」のような予約語は、OracleのJNLPサーブレットの実装によって提供されるもので、JNLP標準で定義されたものではありません。

---

## 4.3. プログラムの実行

本節では、クライアントの実行方法について説明します。

### 4.3.1. クライアントの実行

Webアプリケーションをデプロイ後、WebブラウザからJNLPファイルにアクセスするとJava Web Startが開始され、クライアントが実行されます。

以下は、Webアプリケーションのコンテキストがhelloの場合、JNLPファイルを要求するURLの例です。この場合、HelloClient.jnlpとHelloClient.jnlpに定義されたhello-client.jar、jclient.jarは、URLパスがappとなっているので、helloディレクトリーあるいはwarファイル内のappディレクトリーに位置します。

```
http://host1:8088/hello/app/HelloClient.jnlp
```

JNLPファイルをWebブラウザで要求した結果、指定されたクライアントが実行されます。Webブラウザとは別のJavaウィンドウが表示され、クライアントが実行されたことを確認できます。

### 4.3.2. クライアント・コンテナでの実行

JNLPクライアントの場合、クライアント・コンテナを介して依存性注入を使用できます。そのためには、上記とは異なる方法が必要です。

以下は、HelloClient.javaでInjectionを使用したコードの例です。

**[例 4.4] <<HelloClient.java>>**

```
package helloejb;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;

import javax.swing.*;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.ejb.EJB;

public class HelloClient extends JFrame {
    @EJB
    private static Hello hello;

    public static void main(String[] args) {
        new HelloClient();
    }

    public HelloClient() {
        try {
            JLabel label = new JLabel(hello.sayHello());
            label.setFont(new Font("Helvetica", Font.BOLD, 15));
            getContentPane().setLayout(new BorderLayout());
            getContentPane().add(label, BorderLayout.CENTER);
            setSize(500, 250);
            setVisible(true);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

JNLPファイルもこれに合わせて作成する必要があります。クライアントのメインクラスの代わりにJEUSのクライアント・コンテナ・クラスを記述し、そのクラスに渡すパラメータを設定します。パラメータについては「[1.6.1. コンソールでの実行](#)」を参照してください。

**<resources>**タグにはclient.jarの代わりにclientcontainer.jarを記述し、JEUSクライアント・コンテナがJava Web Startで実行するモードであることが分かるように、jeus.client.container.jwsプロパティをtrueに設定する必要があります。

また、Java Web Startプログラムの場合、パーミッションを確認するために**<security>**タグに<all-permission>または<j2ee-application-client-permissions>を記述し、JARファイルは必ず署名を行います。

#### [例 4.5] <<HelloClient.jnlp>>

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0" codebase="$$codebase">
  <information>
    <title>HelloClient</title>
    <vendor>TmaxSoft</vendor>
  </information>
  <security>
    <all-permissions/>
  </security>
  <resources>
    <j2se version="1.5+" href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="clientcontainer.jar"/>
    <jar href="hello-client.jar"/>
    <property name="jeus.client.container.jws" value="true"/>
  </resources>
  <application-desc main-class="jeus.client.container.ClientContainer">
    <argument>-main</argument>
    <argument>helloejb.HelloClient</argument>
  </application-desc>
</jnlp>
```

クライアント・コンテナを使用する場合と同様に、WebブラウザまたはJava Web Startを介してJNLPファイルをロードした場合、実行結果は同じです。

## JARファイルの署名

JNLPファイルに<security>タグがある場合、Java Web Startは必ず署名されたJARファイルを要求するため、例の**hello-client.jar**を署名する必要があります。

以下は、JDKで提供するkeytoolとjarsignerを使用してhello-client.jarを署名する例です。

```
keytool -genkey -alias helloclient -keypass 1234 -keystore helloks -storepass 1234
jarsigner -keystore helloks hello-client.jar helloclient
```



# 用語集

## Java Web Start

クライアントがJNLPリソースをアクセスおよびダウンロードできるソフトウェアです。

## JNLP

Java Network Launching Protocolの略語で、ソフトウェア・コンポーネントをデプロイするプロトコルです。

