

JEUS サーバガイド

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 サーバガイド

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 | xix |
| 第1章 紹介 | 1 |
| 1.1. 構成要素 | 1 |
| 1.2. Domain Administration Server(DAS) | 3 |
| 1.3. Managed Server(MS) | 4 |
| 1.3.1. エンジン・サービス | 7 |
| 1.4. クラス・ローダーの構造 | 8 |
| 1.5. サーバー・ディレクトリー構造 | 9 |
| 1.6. ランチャー | 12 |
| 1.7. Managed ServerのINDEPENDENT状態 | 12 |
| 第2章 JEUSの設定 | 15 |
| 2.1. 概要 | 15 |
| 2.2. サーバーの追加 | 15 |
| 2.2.1. WebAdminの使用 | 15 |
| 2.2.2. コンソール・ツールの使用 | 31 |
| 2.3. サーバーの設定 | 33 |
| 2.3.1. 基本設定 | 33 |
| 2.3.2. リスナーの設定 | 42 |
| 2.3.3. スレッド・プールの設定 | 46 |
| 2.3.4. ライフサイクル呼び出しの設定 | 50 |
| 2.3.5. リソース参照の設定 | 57 |
| 第3章 JEUSサーバーの制御とモニタリング | 63 |
| 3.1. サーバーの制御とモニタリング | 63 |
| 3.1.1. Managed Serverのライフサイクル | 63 |
| 3.1.2. Managed Serverの起動 | 65 |
| 3.1.3. Managed Serverの終了 | 67 |
| 3.1.4. Managed Serverの一時停止 | 68 |
| 3.1.5. Managed Serverの再開 | 70 |
| 3.2. サーバー・エンジンの設定 | 73 |
| 3.2.1. エンジン使用可否の設定 | 73 |
| 3.2.2. エンジンの初期化時点の設定 | 77 |
| 3.3. スレッドのモニタリングと制御 | 78 |
| 3.3.1. スレッドのモニタリング | 78 |
| 3.3.2. スレッドの制御 | 84 |
| 3.4. メモリーのモニタリングと制御 | 88 |
| 3.4.1. メモリーのモニタリング | 88 |
| 3.4.2. メモリー使用量による制御 | 90 |
| 第4章 JNDIネーミング・サーバー | 91 |
| 4.1. 概要 | 91 |

| | | |
|------------|---|------------|
| 4.2. | 基本概念と構造 | 91 |
| 4.2.1. | 基本概念 | 91 |
| 4.2.2. | バインドされたオブジェクトの確認 | 92 |
| 4.2.3. | JNDIネーミング・サーバーのアーキテクチャー | 93 |
| 4.2.4. | JNDIクラスタリング | 95 |
| 4.3. | JNDIネーミング・サーバーの設定 | 97 |
| 4.3.1. | JNSServerの設定 | 97 |
| 4.3.2. | JNSClientの設定 | 101 |
| 4.4. | クラスター環境でのJNDI | 101 |
| 4.4.1. | ルックアップ | 102 |
| 4.5. | JNDIのプログラミング | 102 |
| 4.5.1. | JEUSの環境設定 | 103 |
| 4.5.2. | 初期コンテキストのプロパティ設定 | 103 |
| 4.5.3. | コンテキストを使用した名前付きオブジェクトのルックアップ | 105 |
| 4.5.4. | 名前付きオブジェクトの使用 | 106 |
| 4.5.5. | コンテキストのクローズ | 106 |
| 4.5.6. | クラスタリング・コンテキストの作成 | 107 |
| 4.5.7. | リモート・ルックアップの実行 | 107 |
| 第5章 | 外部リソース | 109 |
| 5.1. | リソースの種類 | 109 |
| 5.2. | リソースの設定 | 110 |
| 5.2.1. | データソースの設定 | 111 |
| 5.2.2. | メールソースの設定 | 111 |
| 5.2.3. | URLソースの設定 | 112 |
| 5.2.4. | カスタム・リソースの設定 | 113 |
| 5.2.5. | 外部リソースの設定 | 118 |
| 5.2.6. | 外部リソースの設定 | 120 |
| 第6章 | DBコネクション・プールとJDBC | 125 |
| 6.1. | 概要 | 125 |
| 6.2. | データソースとJDBCコネクション・プーリング | 125 |
| 6.2.1. | JDBCドライバー | 126 |
| 6.2.2. | JDBCコネクション・プール | 126 |
| 6.2.3. | データソース | 127 |
| 6.2.4. | クラスター・データソース | 129 |
| 6.3. | データソースとコネクション・プールの管理 | 132 |
| 6.4. | データソース設定 | 134 |
| 6.4.1. | 基本設定 | 134 |
| 6.4.2. | コネクション・プールの設定 | 138 |
| 6.5. | クラスター・データソースの設定 | 146 |
| 6.5.1. | クラスター・データソースの設定 | 147 |
| 6.5.2. | クラスター・データソースに属するコンポーネント・データソースの設定 | 151 |
| 6.6. | データソース関連設定の動的変更 | 152 |

| | | |
|------------|------------------------------------|------------|
| 6.6.1. | データソースの追加 | 152 |
| 6.6.2. | サーバーへのデータソースの登録 | 156 |
| 6.6.3. | サーバーからのデータソースの削除 | 160 |
| 6.6.4. | クラスターへのデータソースの登録 | 163 |
| 6.6.5. | クラスターからのデータソースの削除 | 168 |
| 6.6.6. | クラスターへのサーバーの追加 | 173 |
| 6.6.7. | クラスターからのサーバーの削除 | 177 |
| 6.6.8. | クラスターの削除 | 180 |
| 6.6.9. | データソースの削除 | 184 |
| 6.6.10. | データソース設定の変更 | 187 |
| 6.6.11. | データソース設定の確認 | 195 |
| 6.7. | クラスター・データソース関連設定の動的変更 | 198 |
| 6.7.1. | クラスター・データソースの追加 | 199 |
| 6.7.2. | サーバーへのクラスター・データソースの登録 | 202 |
| 6.7.3. | サーバーからのクラスター・データソースの削除 | 205 |
| 6.7.4. | クラスターへのクラスター・データソースの登録 | 209 |
| 6.7.5. | クラスターからのクラスター・データソースの削除 | 213 |
| 6.7.6. | クラスターへのサーバーの追加 | 217 |
| 6.7.7. | クラスターからのサーバーの削除 | 217 |
| 6.7.8. | クラスターの削除 | 218 |
| 6.7.9. | クラスター・データソースの削除 | 218 |
| 6.7.10. | クラスター・データソース設定の変更 | 223 |
| 6.7.11. | クラスター・データソース設定の確認 | 226 |
| 6.8. | JDBCコネクション・プールのモニタリング | 228 |
| 6.8.1. | JDBCコネクション・プール・リストの確認 | 229 |
| 6.8.2. | 特定のJDBCコネクション・プールの詳細情報の確認 | 232 |
| 6.9. | JDBCコネクション・プールの制御 | 234 |
| 6.9.1. | コネクション・プールの作成 | 235 |
| 6.9.2. | コネクション・プールの無効化 | 238 |
| 6.9.3. | コネクション・プールの有効化 | 240 |
| 6.9.4. | コネクション・プールのコネクションの切り替え | 243 |
| 6.9.5. | コネクション・プールのコネクション数の最小化 | 246 |
| 6.10. | JEUS JDBCプログラミング | 249 |
| 6.10.1. | データソースからのコネクションの取得 | 249 |
| 6.10.2. | トランザクション・プログラミング・ルール | 249 |
| 6.10.3. | JDBCドライバのコネクション実装インスタンスの取得 | 250 |
| 6.10.4. | スタンドアロン・クライアントでのコネクション・プール | 250 |
| 6.11. | グローバル・トランザクション(XA)とコネクションの共有 | 251 |
| 6.12. | 複数ユーザーに対するコネクション・プーリング・サービス | 251 |
| 第7章 | トランザクション・マネージャー | 253 |
| 7.1. | 概要 | 253 |
| 7.1.1. | アプリケーション | 254 |

| | | |
|------------|--|------------|
| 7.1.2. | JEUSTランザクション・マネージャー | 255 |
| 7.1.3. | リソース・マネージャー | 256 |
| 7.2. | サーバー・ランザクション・マネージャーの設定 | 257 |
| 7.2.1. | ワーカー・スレッド・プールの設定 | 258 |
| 7.2.2. | タイムアウトの設定 | 261 |
| 7.2.3. | ルート・コーディネーターとサブ・コーディネーターに関する設定 | 263 |
| 7.2.4. | ランザクション結合の設定 | 264 |
| 7.3. | クライアント・ランザクション・マネージャーの設定 | 264 |
| 7.3.1. | ランザクション・マネージャーの使用可否の設定 | 264 |
| 7.3.2. | ランザクション・マネージャーのタイプの設定 | 265 |
| 7.3.3. | ランザクション・マネージャーのTCP/IPポートの設定 | 265 |
| 7.3.4. | ワーカー・スレッド・プールの設定 | 265 |
| 7.3.5. | タイムアウトの設定 | 266 |
| 7.4. | ランザクション・アプリケーションの作成 | 267 |
| 7.4.1. | ローカル・ランザクション(Local Transaction) | 268 |
| 7.4.2. | クライアント管理ランザクション(Client Managed Transaction) | 270 |
| 7.4.3. | Bean管理ランザクション(Bean Managed Transaction) | 272 |
| 7.4.4. | コンテナ管理ランザクション(Container Managed Transaction) | 274 |
| 7.4.5. | ランザクション・マネージャーの使用 | 276 |
| 7.5. | ランザクションのリカバリー | 277 |
| 7.5.1. | ランザクションのリカバリー手順 | 277 |
| 7.5.2. | リカバリー関連のログ・ファイル | 279 |
| 7.5.3. | リカバリー関連の設定 | 281 |
| 7.5.4. | リソース・マネージャーの障害 | 281 |
| 7.6. | ランザクション・プロファイル機能 | 282 |
| 7.7. | IP帯域が異なるサーバー間のランザクション通信問題 | 284 |
| 第8章 | ロギング | 285 |
| 8.1. | 概要 | 285 |
| 8.2. | JEUSロガーの基本構造 | 286 |
| 8.2.1. | 概要 | 287 |
| 8.2.2. | ランチャー・ロガー | 289 |
| 8.2.3. | サーバー・ロガー | 290 |
| 8.2.4. | アクセス・ロガー | 292 |
| 8.2.5. | ユーザー・ロガー | 293 |
| 8.2.6. | ロガー・リスト | 293 |
| 8.2.7. | ログ・メッセージのモジュール名 | 297 |
| 8.3. | ロギングの設定 | 298 |
| 8.3.1. | ロガー情報の確認 | 298 |
| 8.3.2. | ロガーの動的設定 | 301 |
| 8.3.3. | 標準出力と標準エラーのログ形式の出力 | 307 |
| 8.3.4. | ロガーの設定 | 310 |
| 8.3.5. | ログ・ファイルのローテーションの設定 | 316 |

| | |
|---|------------|
| 8.3.6. プロパティの設定 | 318 |
| 付録 A. JEUSで使用するポート | 319 |
| A.1. サーバー・ポート | 319 |
| 付録 B. JDBCデータソースの構成例 | 321 |
| B.1. 概要 | 321 |
| B.2. Oracle Thin(Type4)の構成例 | 322 |
| B.2.1. Oracle Thinコネクション・プール・データソース | 322 |
| B.2.2. Oracle Thin XAデータソース | 324 |
| B.3. Oracle OCI (Type2)の構成例 | 326 |
| B.3.1. Oracle OCIコネクション・プール・データソース | 326 |
| B.4. DB2の構成例 | 328 |
| B.4.1. DB2 Type4(JCC)コネクション・プール・データソース | 328 |
| B.4.2. DB2 Type4(JCC) XAデータソース | 330 |
| B.4.3. DB2 Type2(JCC) XAデータソース | 332 |
| B.5. Sybaseの構成例 | 334 |
| B.5.1. Sybase jConnect 5.xコネクション・プール・データソース | 334 |
| B.5.2. Sybase jConnect 6.x XAデータソース | 336 |
| B.6. MSSQLの構成例 | 338 |
| B.6.1. MSSQL 2005コネクション・プール・データソース | 338 |
| B.7. Informixの構成例 | 340 |
| B.7.1. Informixコネクション・プール・データソース | 340 |
| B.8. Tiberoの構成例 | 342 |
| B.8.1. Tiberoコネクション・プール・データソース | 344 |
| B.9. MySQL 5.xの構成例 | 344 |
| B.9.1. MySQL Connector/Jコネクション・プール・データソース | 344 |
| 索引 | 347 |

図目次

| | | |
|----------|---|----|
| [図 1.1] | JEUSの構成要素 | 1 |
| [図 1.2] | Isolated方式クラス・ローダーの階層構造 | 9 |
| [図 1.3] | サーバー・ディレクトリー構造 | 10 |
| [図 2.1] | WebAdminのサーバー・リスト画面 | 16 |
| [図 2.2] | WebAdminのサーバー設定画面 - サーバーの追加 | 17 |
| [図 2.3] | WebAdminのサーバー設定画面 - 詳細設定 | 18 |
| [図 2.4] | のサーバー設定画面 - サーバー追加の結果 | 20 |
| [図 2.5] | サーバーの追加 - System Thread Poolの設定 | 21 |
| [図 2.6] | サーバーの追加 - System Thread Pool設定の保存 | 22 |
| [図 2.7] | サーバーの追加 - デフォルト・リスナーの追加 | 23 |
| [図 2.8] | サーバーの追加 - デフォルト・リスナー情報の設定 | 24 |
| [図 2.9] | サーバーの追加 - デフォルト・リスナー情報設定の保存 | 25 |
| [図 2.10] | サーバーの追加 - デフォルト・リスナーの設定 | 26 |
| [図 2.11] | サーバーの追加 - デフォルト・リスナー設定の保存 | 26 |
| [図 2.12] | サーバーの追加 - Webエンジンの設定 | 27 |
| [図 2.13] | サーバーの追加 - 設定情報の反映(1) | 28 |
| [図 2.14] | サーバーの追加 - 設定情報の反映(2) | 28 |
| [図 2.15] | サーバーの追加 - 設定情報の反映(3) | 29 |
| [図 2.16] | サーバーの追加 - 新規サーバーの確認 | 30 |
| [図 2.17] | サーバーの追加 - 新規サーバーの実行(1) | 30 |
| [図 2.18] | サーバーの追加 - 新規サーバーの実行(2) | 31 |
| [図 2.19] | デフォルト・リスナーの設定 | 43 |
| [図 2.20] | 個別リスナーの設定 | 44 |
| [図 2.21] | 個別リスナーのSSL設定 | 45 |
| [図 2.22] | WebAdminによるライフサイクル呼び出しの設定(1) | 51 |
| [図 2.23] | WebAdminによるライフサイクル呼び出しの設定(2) | 52 |
| [図 2.24] | WebAdminによるライフサイクル呼び出しの設定(3) | 53 |
| [図 2.25] | WebAdminによるライフサイクル呼び出しの設定(4) | 54 |
| [図 2.26] | WebAdminによるライフサイクル呼び出しの設定(5) | 55 |
| [図 2.27] | WebAdminによるライフサイクル呼び出しの設定(6) | 56 |
| [図 2.28] | WebAdminによるライフサイクル呼び出しの設定(7) | 56 |
| [図 2.29] | WebAdminによるライフサイクル呼び出しの設定(8) | 57 |
| [図 2.30] | WebAdminによるリソース参照の設定(1) | 58 |
| [図 2.31] | WebAdminによるリソース参照の設定(2) | 59 |
| [図 2.32] | WebAdminによるリソース参照の設定(3) | 60 |
| [図 2.33] | WebAdminによるリソース参照の設定(4) | 61 |
| [図 2.34] | WebAdminによるリソース参照の設定(6) | 61 |
| [図 3.1] | MSの状態転移図 | 63 |
| [図 3.2] | ドメインのサーバー・リストの照会 | 68 |
| [図 3.3] | ドメインのサーバーの一時停止 | 69 |

| | | |
|----------|--|-----|
| [図 3.4] | ドメインのサーバー・リスト | 71 |
| [図 3.5] | ドメインのサーバーの再開 | 71 |
| [図 3.6] | サーバー内部のエンジン設定画面 | 74 |
| [図 3.7] | サーバー内部のエンジン設定結果画面 | 75 |
| [図 3.8] | スレッド・モニタリング画面 | 79 |
| [図 3.9] | WebAdminによるスレッド情報の確認 | 79 |
| [図 3.10] | WebAdminによる特定のスレッドのスタック・トレースの照会 | 83 |
| [図 3.11] | System Information画面 | 88 |
| [図 3.12] | WebAdminによるメモリー情報の確認 | 89 |
| [図 4.1] | JNDIリストの照会 | 92 |
| [図 4.2] | JNDIの詳細照会 | 93 |
| [図 4.3] | JNSServerとJNSClientの関係 | 93 |
| [図 4.4] | クラスタリング環境のJEUS JNDIアーキテクチャー | 96 |
| [図 4.5] | ネーミング・サーバーの設定 | 97 |
| [図 4.6] | 共用スレッド・プールの設定 | 98 |
| [図 4.7] | サービス専用スレッド・プールの設定 | 99 |
| [図 5.1] | WebAdminによるリソースの設定 | 111 |
| [図 5.2] | WebAdminによるメールソースの設定(1) | 112 |
| [図 5.3] | WebAdminによるメールソースの設定(2) | 112 |
| [図 5.4] | WebAdminによるURLソースの設定(1) | 113 |
| [図 5.5] | WebAdminによるURLソースの設定(2) | 113 |
| [図 5.6] | WebAdminによるカスタム・リソースの設定(1) | 116 |
| [図 5.7] | WebAdminによるカスタム・リソースの設定(2) | 116 |
| [図 5.8] | WebAdminによるカスタム・リソースの設定(3) | 117 |
| [図 5.9] | WebAdminによるカスタム・リソースの設定(4) | 117 |
| [図 5.10] | WebAdminによる外部リソースの設定 | 118 |
| [図 5.11] | WebAdminによるJMSソースの追加 | 119 |
| [図 5.12] | WebAdminによる外部リソースの設定(1) | 122 |
| [図 5.13] | WebAdminによる外部リソースの設定(2) | 123 |
| [図 5.14] | WebAdminによる外部リソースの設定(3) | 123 |
| [図 5.15] | WebAdminによる外部リソースの設定(4) | 123 |
| [図 6.1] | JEUSのコネクション・プーリング | 126 |
| [図 6.2] | RACに対するクラスター・データソースのフェイルオーバー | 129 |
| [図 6.3] | JEUSDメイン構造におけるデータソースとコネクション・プールの管理 | 133 |
| [図 6.4] | データソースの基本設定画面(1) | 134 |
| [図 6.5] | データソースの基本設定画面(2) | 135 |
| [図 6.6] | コネクション・プール設定画面(1) | 138 |
| [図 6.7] | コネクション・プール設定画面(2) | 139 |
| [図 6.8] | コネクション・プール設定画面(3) | 140 |
| [図 6.9] | クラスター・データソースの設定画面(1) | 147 |
| [図 6.10] | クラスター・データソースの設定画面(2) | 148 |
| [図 6.11] | ドメインへのデータソースの追加(1) | 153 |
| [図 6.12] | ドメインへのデータソースの追加(2) | 154 |

| | | |
|----------|---------------------------------|-----|
| [図 6.13] | ドメインへのデータソースの追加(3) | 155 |
| [図 6.14] | サーバーへのデータソースの登録(1) | 157 |
| [図 6.15] | サーバーへのデータソースの登録(2) | 157 |
| [図 6.16] | サーバーへのデータソースの登録(3) | 158 |
| [図 6.17] | サーバーにJNDIバインドされたデータソースの確認 | 159 |
| [図 6.18] | サーバーからのデータソースの削除(1) | 161 |
| [図 6.19] | サーバーからのデータソースの削除(2) | 161 |
| [図 6.20] | サーバーからのデータソースの削除(3) | 162 |
| [図 6.21] | サーバーからJNDIアンバインドされたデータソースの確認 | 162 |
| [図 6.22] | クラスターへのデータソースの登録(1) | 164 |
| [図 6.23] | クラスターへのデータソースの登録(2) | 164 |
| [図 6.24] | クラスターへのデータソースの登録(3) | 165 |
| [図 6.25] | サーバーにJNDIバインドされたデータソースの確認(1) | 166 |
| [図 6.26] | サーバーにJNDIバインドされたデータソースの確認(2) | 166 |
| [図 6.27] | クラスターからのデータソースの削除(1) | 169 |
| [図 6.28] | クラスターからのデータソースの削除(2) | 169 |
| [図 6.29] | クラスターからのデータソースの削除(3) | 170 |
| [図 6.30] | サーバーからJNDIアンバインドされたデータソースの確認(1) | 171 |
| [図 6.31] | サーバーからJNDIアンバインドされたデータソースの確認(2) | 171 |
| [図 6.32] | クラスターへのサーバーの追加(1) | 174 |
| [図 6.33] | クラスターへのサーバーの追加(2) | 174 |
| [図 6.34] | クラスターへのサーバーの追加(3) | 175 |
| [図 6.35] | サーバーにJNDIバインドされたデータソースの確認 | 175 |
| [図 6.36] | クラスターからのサーバーの削除(1) | 177 |
| [図 6.37] | クラスターからのサーバーの削除(2) | 178 |
| [図 6.38] | クラスターからのサーバーの削除(3) | 178 |
| [図 6.39] | サーバーからJNDIアンバインドされたデータソースの確認 | 179 |
| [図 6.40] | クラスターの削除(1) | 181 |
| [図 6.41] | クラスターの削除(2) | 181 |
| [図 6.42] | サーバーからJNDIアンバインドされたデータソースの確認(1) | 182 |
| [図 6.43] | サーバーからJNDIアンバインドされたデータソースの確認(2) | 182 |
| [図 6.44] | データソースの削除(1) | 185 |
| [図 6.45] | データソースの削除(2) | 185 |
| [図 6.46] | サーバーからJNDIアンバインドされたデータソースの確認 | 186 |
| [図 6.47] | コネクション・プールの作成(1) | 188 |
| [図 6.48] | コネクション・プールの作成(2) | 189 |
| [図 6.49] | コネクション・プールの作成(3) | 189 |
| [図 6.50] | データソース設定の変更(1) | 190 |
| [図 6.51] | データソース設定の変更(2) | 190 |
| [図 6.52] | データソース設定の変更(3) | 191 |
| [図 6.53] | コネクション・プールのランタイム情報の確認 | 192 |
| [図 6.54] | データソース設定の確認(1) | 195 |
| [図 6.55] | データソース設定の確認(2) | 196 |

| | | |
|----------|--------------------------------------|-----|
| [図 6.56] | クラスター・データソースの追加(1) | 199 |
| [図 6.57] | クラスター・データソースの追加(2) | 200 |
| [図 6.58] | クラスター・データソースの追加(3) | 201 |
| [図 6.59] | サーバーへのクラスター・データソースの登録(1) | 203 |
| [図 6.60] | サーバーへのクラスター・データソースの登録(2) | 203 |
| [図 6.61] | サーバーへのクラスター・データソースの登録(3) | 204 |
| [図 6.62] | サーバーにJNDIバインドされたクラスター・データソースの確認 | 204 |
| [図 6.63] | サーバーからのクラスター・データソースの削除(1) | 206 |
| [図 6.64] | サーバーからのクラスター・データソースの削除(2) | 207 |
| [図 6.65] | サーバーからのクラスター・データソースの削除(3) | 207 |
| [図 6.66] | サーバーからJNDIアンバインドされたクラスター・データソースの確認 | 208 |
| [図 6.67] | クラスターへのクラスター・データソースの登録(1) | 210 |
| [図 6.68] | クラスターへのクラスター・データソースの登録(2) | 210 |
| [図 6.69] | クラスターへのクラスター・データソースの登録(3) | 211 |
| [図 6.70] | サーバーにJNDIバインドされたクラスター・データソースの確認 | 212 |
| [図 6.71] | クラスターからのクラスター・データソースの削除(1) | 214 |
| [図 6.72] | クラスターからのクラスター・データソースの削除(2) | 214 |
| [図 6.73] | クラスターからのクラスター・データソースの削除(3) | 215 |
| [図 6.74] | サーバーからJNDIアンバインドされたクラスター・データソースの確認 | 216 |
| [図 6.75] | クラスター・データソースの削除(1) | 219 |
| [図 6.76] | クラスター・データソースの削除(2) | 220 |
| [図 6.77] | サーバーにJNDIアンバインドされたクラスター・データソースの確認(1) | 221 |
| [図 6.78] | サーバーにJNDIアンバインドされたクラスター・データソースの確認(2) | 221 |
| [図 6.79] | クラスター・データソース設定の変更(1) | 224 |
| [図 6.80] | クラスター・データソース設定の変更(2) | 224 |
| [図 6.81] | クラスター・データソース設定の変更(3) | 225 |
| [図 6.82] | クラスター・データソース設定の確認(1) | 226 |
| [図 6.83] | クラスター・データソース設定の確認(2) | 227 |
| [図 6.84] | コネクション・プール・リストの確認(1) | 229 |
| [図 6.85] | コネクション・プール・リストの確認(2) | 230 |
| [図 6.86] | 特定のコネクション・プールの詳細情報の確認(1) | 233 |
| [図 6.87] | 特定のコネクション・プールの詳細情報の確認(2)) | 233 |
| [図 6.88] | コネクション・プールの作成(1) | 236 |
| [図 6.89] | コネクション・プールの作成(2) | 236 |
| [図 6.90] | コネクション・プールの作成(3) | 237 |
| [図 6.91] | コネクション・プールの無効化(1) | 238 |
| [図 6.92] | コネクション・プールの無効化(2) | 239 |
| [図 6.93] | コネクション・プールの無効化(3) | 239 |
| [図 6.94] | コネクション・プールの有効化(1) | 241 |
| [図 6.95] | コネクション・プールの有効化(2) | 241 |
| [図 6.96] | コネクション・プールの有効化(3) | 242 |
| [図 6.97] | コネクション・プールのコネクションの切り替え(1) | 244 |
| [図 6.98] | コネクション・プールのコネクションの切り替え(2) | 244 |

| | | |
|-----------|--|-----|
| [図 6.99] | コネクション・プールのコネクションの切り替え(3) | 245 |
| [図 6.100] | コネクション・プールのコネクション数の最小化(1) | 246 |
| [図 6.101] | コネクション・プールのコネクション数の最小化(2) | 247 |
| [図 6.102] | コネクション・プールのコネクション数の最小化(3) | 248 |
| [図 7.1] | ルート・コーディネーターとサブ・コーディネーターの関係 | 256 |
| [図 7.2] | WebAdminによる共用スレッド・プールの設定 | 259 |
| [図 7.3] | WebAdminによる専用スレッド・プールの設定 | 261 |
| [図 7.4] | WebAdminによるタイムアウトの変更 | 261 |
| [図 7.5] | 単純化されたトランザクションのリカバリー手順 | 277 |
| [図 7.6] | WebAdminによる自動トランザクション・リカバリーの設定 | 280 |
| [図 7.7] | WebAdminによるトランザクション・ログ・ディレクトリーの変更 | 280 |
| [図 8.1] | ログ・ディレクトリー | 287 |
| [図 8.2] | Log Homeパスの設定 | 289 |
| [図 8.3] | WebAdminによるjeusロガー情報の確認 | 299 |
| [図 8.4] | WebAdminによるロガーの動的設定(1) | 301 |
| [図 8.5] | WebAdminによるロガーの動的設定(2) | 302 |
| [図 8.6] | WebAdminによるロガーの動的設定(3) | 302 |
| [図 8.7] | WebAdminによるロガーの動的設定(4) | 303 |
| [図 8.8] | WebAdminによるロガーの動的設定(5) | 303 |
| [図 8.9] | WebAdminによるロガーの動的設定(6) | 304 |
| [図 8.10] | WebAdminによるロガーの動的設定(7) | 305 |
| [図 8.11] | WebAdminで標準出力と標準エラーをJEUSのログ形式で出力する設定 | 308 |
| [図 8.12] | WebAdminのロガー設定画面 | 310 |
| [図 8.13] | WebAdminによるログ・ファイルのローテーションの設定 | 317 |
| [図 B.1] | Oracle Thinコネクション・プール・データソースの構成例 | 322 |
| [図 B.2] | Oracle Thinコネクション・プール・データソースの構成例 | 323 |
| [図 B.3] | Oracle Thin XAデータソースの構成例 | 324 |
| [図 B.4] | Oracle Thin XAデータソースの構成例 | 325 |
| [図 B.5] | Oracle OCIコネクション・プール・データソースの構成例 | 326 |
| [図 B.6] | Oracle OCIコネクション・プール・データソースの構成例 | 327 |
| [図 B.7] | DB2 Type4コネクション・プール・データソースの構成例 | 328 |
| [図 B.8] | DB2 Type4コネクション・プール・データソースの構成例 | 329 |
| [図 B.9] | DB2 Type4 XAデータソースの構成例 | 330 |
| [図 B.10] | DB2 Type4 XAデータソースの構成例 | 331 |
| [図 B.11] | DB2 Type2 XAデータソースの構成例 | 332 |
| [図 B.12] | DB2 Type2 XAデータソースの構成例 | 333 |
| [図 B.13] | Sybase jConnect 5.xコネクション・プール・データソースの構成例 | 334 |
| [図 B.14] | Sybase jConnect 5.xコネクション・プール・データソースの構成例 | 335 |
| [図 B.15] | Sybase jConnect 6.x XAデータソースの構成例 | 336 |
| [図 B.16] | Sybase jConnect 6.x XAデータソースの構成例 | 337 |
| [図 B.17] | MSSQL 2005コネクション・プール・データソースの構成例 | 338 |
| [図 B.18] | MSSQL 2005コネクション・プール・データソースの構成例 | 339 |
| [図 B.19] | Informixコネクション・プール・データソースの構成例 | 340 |

| | | |
|----------|--|-----|
| [図 B.20] | Informixコネクション・プール・データソースの構成例 | 341 |
| [図 B.21] | Tiberoコネクション・プール・データソースの構成例 | 342 |
| [図 B.22] | Tiberoコネクション・プール・データソースの構成例 | 343 |
| [図 B.23] | MySQL Connector/Jコネクション・プール・データソースの構成例 | 344 |
| [図 B.24] | MySQL Connector/Jコネクション・プール・データソースの構成例 | 345 |

例目次

| | | |
|---------|--|-----|
| [例 2.1] | コンソール・ツールによるサーバーの追加 | 31 |
| [例 2.2] | コンソール・ツールによる動的な設定変更 | 34 |
| [例 2.3] | <<LifecycleTester.java>> | 50 |
| [例 5.1] | カスタム・リソースの例題クラス | 114 |
| [例 5.2] | カスタム・リソースを作成するファクトリー・クラスの例 | 114 |
| [例 5.3] | コンソール・ツールによるカスタム・リソースの照会・追加・削除 | 117 |
| [例 5.4] | jeus.external.ResourceBootstrapper | 121 |
| [例 5.5] | コンソール・ツールによる外部リソースの照会・追加・削除 | 124 |
| [例 6.1] | jeus.jdbc.connectionpool.JEUSConnectionChecker | 143 |
| [例 6.2] | jeus.jdbc.helper.DataSourceSelector | 150 |
| [例 6.3] | DataSourceSelectorの実装例 | 151 |
| [例 7.1] | <<Client.java>> | 268 |
| [例 7.2] | <<Client.java>> | 270 |
| [例 7.3] | <<ProductManagerEJB.java>> | 272 |
| [例 7.4] | <<Client.java>> | 272 |
| [例 7.5] | <<ProductManagerEJB.java>> | 273 |
| [例 7.6] | <<Client.java>> | 275 |
| [例 7.7] | <<ProductManagerEJB.java>> | 275 |
| [例 7.8] | <<CoordinatorProfileListener.java>> | 282 |
| [例 7.9] | <<TransactionInfo.java>> | 282 |
| [例 8.1] | 設定ファイルのエラーによる起動失敗時にランチャーが残したログ・メッセージ | 293 |
| [例 8.2] | サーバー・ログ・メッセージの出力例 | 293 |
| [例 8.3] | ログ・フォーマッターを使用したサーバー・ログ・メッセージの出力例 | 293 |
| [例 8.4] | access.logの記録例 | 293 |
| [例 8.5] | コンソール・ツールによるjeusロガー情報の確認 | 300 |
| [例 8.6] | コンソール・ツールによるロガーの追加/変更/削除 | 306 |
| [例 8.7] | コンソール・ツールで標準出力と標準エラーをJEUSのログ形式で出力する設定 | 308 |

このガイドについて

対象読者

本書は、JEUSの全般的な構造とサービスについて説明しており、JEUSに関する基本的な知識を必要とするユーザーを対象としています。

前提知識

本書では、JEUSの構造を理解するための基本的な内容とJEUSが提供するサービスについて説明します。JEUSの設定と制御を正しく行うためには、事前にJEUSの構造を理解する必要があります。ただし、本書ですべての内容を詳しく説明することは困難なため、詳細内容についてはテーマ別のガイドを参照してください。

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

最初に説明する「[第1章 紹介](#)」は、JEUSを理解するのに最も重要な内容です。なお、以降の章ではサービスごとに必要な内容や設定方法について説明しますので、必要に応じて参照してください。

また、コンソール・ツールの使用方法について説明するときは、便宜上、省略されたコマンドおよびオプションの名前を使用します。それらの詳しい使用方法については、コンソール・ツールでコマンド名を引数としてhelpを実行するか、『*JEUS リファレンスガイド*』の「4.2. jeusadmin」を参照してください。

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、JEUSサーバーの設定とサーバーを構成するサービスで構成されています。

以下のように、計8章と2つの付録から構成されています。

- [「第1章 紹介」](#)

JEUSの構成要素と構成要素が提供するサービスについて説明します。

- [「第2章 JEUSの設定」](#)

JEUSの構成要素の設定方法について説明します。

- [「第3章 JEUSサーバーの制御とモニタリング」](#)

JEUSを制御およびモニタリングする方法について説明します。

- [「第4章 JNDIネーミング・サーバー」](#)

JEUS JNDIの基本的な内容とアプリケーションの開発方法について説明します。

- [「第5章 外部リソース」](#)

JEUSと連携して使用できる外部リソースについて説明します。

- [「第6章 DBコネクション・プールとJDBC」](#)

JEUSで提供するコネクション・プールと付加機能、使用方法について説明します。

- [「第7章 トランザクション・マネージャー」](#)

JEUSのトランザクション・マネージャーとその周辺要素について説明します。

- [「第8章 ロギング」](#)

JEUSのロギング・システムについて説明します。

- [「付録 A. JEUSで使用するポート」](#)

JEUSで使用するポートについて説明します。

- [「付録 B. JDBCデータソースの構成例」](#)

主なJDBCドライバーに対するコネクション・プールの設定例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 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 セッション管理ガイド | JEUSで運用するセッション・マネージャーとセッション・サーバーの設定および管理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 MQガイド | JEUSメッセージベース・システム(JMS)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 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 |
| JEUS JCAガイド | JEUSとレガシーシステムを接続するためのコネクタ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 JMXガイド | JEUS JMX(Java Management eXtensions)モジュールを使用したJEUSの管理について記述しています |
| JEUS スケジューラガイド | JEUSのスケジューラ機能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 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 ノードマネージャガイド | JEUSノードマネージャーの概念と動作方式を理解するための基本的な内容について記述しています |

参照文献

- domain.xml設定に関するXMLリファレンス: JEUS_HOME/docs/manuals/xml-reference/index.html

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

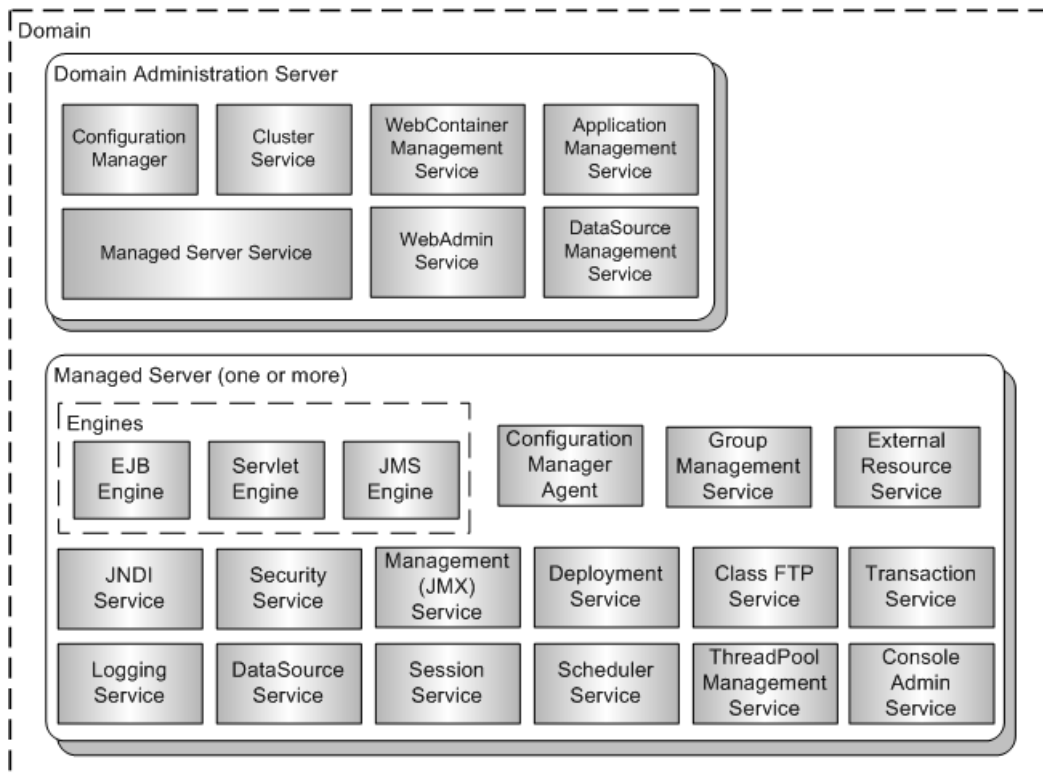
第1章 紹介

本章では、JEUSサーバーの構成要素とその構成要素が提供するサービスについて説明します。

1.1. 構成要素

以下は、JEUSの主要構成要素です。

[図 1.1] JEUSの構成要素



ドメインは、Domain Administration Server(以下、DAS)とManaged Server(以下、MS)で構成されます。

- **ドメイン(Domain)**

サーバーとクラスターを管理するグループです。ドメインについての詳細は、『JEUSドメインガイド』を参照してください。

- **Domain Administration Server(DAS)**

DASは、ドメインに1つだけ存在するドメイン管理サーバーです。ドメイン全体の設定とアプリケーションを管理します。また、ドメインに属する複数の**MS(Managed Server)**を管理・制御します。

以下は、DASの主要サービスです。

- JEUS WebAdmin(以下、WebAdmin)サービス
- 動的設定反映サービス
- ドメイン・アプリケーション管理サービス
- ドメイン・データソース管理サービス
- クラスター管理サービス

● Managed Server(MS)

MSは、ドメインに1つ以上存在できる構成要素です。ユーザーがデプロイするアプリケーションをサービスし、アプリケーションが必要とするサービスを提供します。

以下は、MSの主要サービスです。

- EJB、WEB、JMSエンジン・サービス
- JNDIサービス
- マネージメント・サービス
- セキュリティー・サービス
- HTTPセッション・クラスタリング・サービス
- トランザクション・サービス
- データベース(以下、DB)またはエンタープライズ情報システム(EIS : Enterprise Information System) 接続サービス
- スケジューラー・サービスなど

参考

ドメインの構成要素についての詳細は、『JEUS ドメインガイド』の「2.2. ドメインの構成」を参照してください。

1.2. Domain Administration Server(DAS)

Domain Administration Server(DAS)はドメインを管理するサーバーで、ドメインに1つだけ存在します。DASはドメインの設定を管理し、ドメインでサービスされるアプリケーションを管理・制御します。また、ドメインに属するMSを管理します。

参考

DASの概念についての詳しい説明は、『JEUS ドメインガイド』の「1.3.1. Domain Administration Server(DAS)」を参照してください。

以下は、DASで利用できる主要サービスです。ドメイン単位で管理するサービスであるため、DASでのみ実行されます。以下のサービスを使用するためには、DASと接続されたコンソール・ツール(jeusadmin)またはWebAdminを利用します。

- **WebAdminサービス**

WebAdminはドメインを管理するためのWebベースの管理コンソールです。WebAdminを使用することで、より便利にサーバーとアプリケーションを管理・制御することができます。

WebAdminはWebアプリケーションの形式でDASにデプロイされます。WebAdminのデプロイが完了していなくても、DASは起動を完了し、jeusadminにより制御とサービスが可能です。

WebAdminについての詳細は、『JEUS WebAdminガイド』を参照してください。

- **動的設定反映サービス**

DASでは動的設定反映サービスにより、ドメインの設定がドメインに属するすべてのサーバーで同一に維持されるように管理します。ドメインの設定の変更はDASを通じてのみ可能です。

WebAdminまたはコンソール・ツールを使って、このサービスのコマンドを実行できます。ドメインのすべての設定変更は動的設定反映サービスからロックを取得する必要があります。設定を変更した後、反映(Activate Changes)を実行すると、ドメインに設定の変更が適用されます。このとき、動的反映が可能な項目は運用中のサーバーを再起動しなくても変更した設定をサーバーに適用できますが、動的反映が不可能な項目は運用中のサーバーを再起動しないと適用されません。

ドメインで設定の変更を担当するサービスの詳細については、『JEUS ドメインガイド』の「第3章 ドメインの設定変更」を参照してください。

- **ドメイン・アプリケーション管理サービス**

DASではアプリケーション管理サービスにより、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管理します。MSでアプリケーションをサービスするには、まずドメインにアプリケーションを登録する必要があります。それから、アプリケーションをサービスする対象をターゲットとして設定し、DASでアプリケーションのdeployコマンドを実行します。ドメイン・アプリケーション管理サービスでは、DASでのアプリケーションの

状態とMSでのアプリケーションの状態が一致するようにします。なお、DASでアプリケーション・ファイルが変更されたら、MSでも変更されたファイルを再転送してもらって同期化します。

ドメインでのアプリケーションの管理方法については、『*JEUS アプリケーション&デプロイメントガイド*』の「第1章 ドメイン環境でのアプリケーション管理」を参照してください。

- **ドメイン・データソース管理サービス**

DASではデータソース管理サービスにより、ドメインに登録されたすべてのデータソースを管理します。MSでデータソースを使用するには、ドメインに登録されたデータソースを指定します。ドメインでデータソースを登録、管理する方法については、「[第6章 DBコネクション・プールとJDBC](#)」を参照してください。

- **クラスター管理サービス**

DASではクラスター管理サービスにより、ドメインに属するクラスターを管理します。クラスターは同じサービスを行う複数のサーバーの集合であり、サービスのロード・バランシングと障害状況におけるフェイルオーバー機能を提供します。クラスターについての詳細は、『*JEUS ドメインガイド*』の「第5章 JEUSクラスターリング」を参照してください。

- **Managed Server(MS)サービス**

DASはMSで実行されるすべてのサービスを実行でき、MSと同様にアプリケーション・サービスを行うことができます。規模の小さい運用系や開発系など、1つのサーバーでサービスが可能な状況では、DASでMSと同様のアプリケーション・サービスを実行できます。MSのサービスについての詳細は、「[1.3. Managed Server\(MS\)](#)」を参照してください。

参考

DASがMSの役割を果たすことは可能ですが、推奨しません。開発系や規模のとても小さい運用系では、ドメインにDASだけ存在しても無理なくサービスを実行できることがあります。ほとんどの場合は、DASは管理の役割に限るようにして、アプリケーション・サービスはMSを別途設けて実行するのが一般的です。

1.3. Managed Server(MS)

Managed Server(MS)は、実際にアプリケーションをサービスするためのエンジンと様々なサービスを行うサーバー・インスタンスを意味します。MSはドメインに複数存在することができます。MSの主な役割は、ユーザーがデプロイするアプリケーションをサービスし、アプリケーションが必要とするリソースやサービスを提供することです。

以下は、MSの主要サービスに関する説明です。

- **エンジン・サービス**

MSの代表的なサービスです。サーブレット、EJBコンポーネントを管理するエンジンとJava Message Service(JMS)を提供するエンジン・サービスがあります。エンジン・サービスについての詳細は、「[1.3.1. エンジン・サービス](#)」を参照してください。

● JNDIサービス

JNDIサービスは、Java EEで定義されたJNDI標準のメカニズムを提供することです。JEUSに登録された多様なオブジェクトを決められた名前で検索して使用できる方法を提供します。

JEUSサーバーでは、このようなサービスを提供するオブジェクトをJNDIネーミング・サーバーといいます。クラスタリング環境では、それぞれのMSで動作するJNDIネーミング・サーバー同士が情報を交換するため、まるで1つのJNDIネーミング・サーバが存在するように動作します。JNDIサービスの詳細については、「[第4章 JNDIネーミング・サーバー](#)」を参照してください。

● マネージメント・サービス

マネージメント・サービスは、Java Management Extensions(JMX)によりサービス、コンポーネント、アプリケーションを管理およびモニタリングするサービスです。

JMXに記述されたJMX Remote API Connector(RMI Connector/JMXMP Connector)、HTMLアダプター、SNMPアダプターなどの管理オブジェクトをJEUSマネージャーに設定し、これらによりJEUSの管理およびモニタリング情報に接続する方法を提供します。マネージメント・サービスの設定方法とそれぞれの管理オブジェクトの詳細については『JEUS JMXガイド』を参照してください。

● セキュリティー・サービス

セキュリティー・サービスは、アプリケーションと内部コンポーネントのセキュリティー要求に対する応答と認証作業を行います。

セキュリティー・サービスの詳細については『JEUS セキュリティガイド』を参照してください。

● HTTPセッション・クラスタリング・サービス

HTTPセッション・クラスタリング・サービスは、サーブレット・エンジン間でHTTPセッションを維持するサービスです。

JEUSは分散式HTTPセッション・クラスタリングをサポートします。このサービスにより、サーブレット・エンジンで障害が発生しても、HTTPセッションが維持されます。分散式HTTPセッション・クラスタリングは、サーバーを運用時にHTTPセッションのメモリー負担が小さい効率的な分散方式です。JEUSで提供するHTTPセッション・クラスタリング・サービスの詳細については、『*JEUS セッション管理ガイド*』の「第2章 分散セッション・サーバー」を参照してください。

このサービスはクラスタリングに参加する場合、自動で提供されます。クラスタリングに参加しなくても、ドメインに属するすべてのサーバーでセッションの維持を提供するサービスがありますが、このサービスには制限事項があります。詳しい内容は、『*JEUS セッション管理ガイド*』の「2.8.2. ドメイン・スコープのセッション・クラスター・モード」を参照してください。

● クラスFTPサービス

EJB 2.xの場合、リモート・クライアントでEJBを呼び出すには、基本的にRMIスタブ・クラスが必要になります。このとき、リモート・クライアントにRMIスタブ・クラスを事前にパッケージングしなくても、クラスFTPサービスを利用して必要なクラスを転送してもらって使用することができます。

クラスFTPサービスを利用しない場合は、RMIスタブ・クラスをリモート・クライアントのクラス・パス(classpath)に格納します。その例については、『JEUS EJBガイド』の「第11章 EJBクライアント」を参照してください。

参考

実際にクラス・ファイルを転送するプロトコルは、FTPプロトコルではなく、HTTPプロトコルを使用します。

EJB 3.xでは、ダイナミック・プロキシ方式を使用するため、このサービスを使用しません。EJB 2.xでも基本的にはダイナミック・プロキシ方式を使用し、設定によってRMIスタブ方式を使用できます。

● スケジューラー・サービス

スケジューラー・サービスは、あらかじめ指定した時間に特定の作業を実行させるサービスです。スケジューラー・サービスについての詳細は、『JEUS スケジューラガイド』を参照してください。

● ロギング・サービス

サーバーで発生するイベントやエラーをファイルに記録するサービスです。JEUSではJava Logging Technologyでサポートするロガーのレベルとハンドラーなどを設定できます。ロギング・サービスについての詳細は、「[第8章 ロギング](#)」を参照してください。

● データベース接続サービス

サーバーではアプリケーションがJDBCコネクション・プールを使ってデータベースにアクセスできるようにサポートします。詳細については、「[第6章 DBコネクション・プールとJDBC](#)」を参照してください。

● トランザクション・サービス

サーバーではトランザクション・マネージャーを使ってアプリケーションがトランザクションを使用できるようにサポートします。トランザクション・マネージャーについての詳細は、「[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。

● 外部リソース

サーバーではアプリケーションと様々な種類の外部リソースとの接続を提供します。

| 外部リソース | 説明 |
|-------------|---------------------------------|
| Data Source | データベースと接続します(JDBC標準で定義したデータソース) |
| Mail Source | メール・サーバーと接続します |
| URL Source | URLソースと接続します |

| 外部リソース | 説明 |
|-----------------|---|
| Message Bridge | JMSデスティネーション間で使用されるブリッジです |
| Custom Resource | JavaBean形式のカスタム・リソースをJNDIリポジトリに登録します |
| External Source | TP Monitor(Tmax)、IBM MQなどのエンタープライズ情報システム(EIS)と接続します(JCAリソース・アダプターをデプロイする方式とは区分されます) |
| JAXR Source | XMLレジストリー・ソースです |

外部リソースに対する接続情報はユーザーが直接追加することができます。JNDIネーミング・サーバーに登録して、アプリケーションでJNDIルックアップ操作により利用できます。外部リソースはドメインに設定されますが、ドメインに属するすべてのサーバーでこのサービスを利用することになるため、サーバーのサービスとして分類されます。

クラスタリング環境では、JNDIネーミング・サーバーを利用してクラスターに属するすべてのサーバーが同じリソース情報を共有して使用することになります。リソースの設定に関する内容については、[「第5章 外部リソース」](#)を参照してください。

● エンタープライズ情報システム(EIS)接続サービス

アプリケーションがデプロイされたJCAリソース・アダプターや外部リソース登録情報を使ってEISにアクセスできます。JCAリソース・アダプターについての詳細は、『JEUS JCAガイド』を参照してください。

1.3.1. エンジン・サービス

エンジンは、Java EEで定義されたEJBコンテナ、Webコンテナにマッピングされる概念であり、ユーザーがデプロイしたコンポーネントを管理およびサービスします。エンジンはサーバーに含まれるサービスであり、ユーザーが設定しなくても、サーバーの起動時に常に基本設定により実行されます。

サーバーの運用中に、WebAdminやコンソール・ツールを使ってエンジンの設定を変更することができます。各エンジンで提供する基本設定と動作的変更が可能な設定については、各エンジンのマニュアルを参照してください。

サーバーで提供するサービス・エンジンは以下の3つの種類があります。

| エンジン | 説明 |
|-----------------------------|--|
| EJBエンジン | EJBコンポーネントを管理、実行するEJBコンテナに当たります。詳細については、『JEUS EJBガイド』を参照してください |
| Webエンジン (またはサーブレット・エンジン) | Webコンテナに当たります。WebクライアントまたはWebサーバーから要求を受けて、ユーザーがデプロイしたサーブレット(またはJSP、JSF)により動的なWebコンテンツを作成します。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 |

| エンジン | 説明 |
|---------|---|
| JMSエンジン | Java Message Service(JMS)を提供します。詳細については、『JEUS MQガイド』を参照してください |

参考

JEUS 6まではエンジン・コンテナにエンジンの設定がない場合は、エンジンが実行されず、アプリケーションがサービスされませんでした。たとえば、Webエンジンを設定していない場合、Webアプリケーションをサービスするには、Webエンジンを追加してJEUSを再起動しなければなりませんでした。ただし、JEUS 7からはMSを起動すると、基本設定でエンジンが実行されるので、ユーザーが設定していなくても、どのアプリケーションのタイプもデプロイしてサービスすることができます。

1.4. クラス・ローダーの構造

本節では、JEUSでサポートする独立したクラス・ローダーについて説明します。

参考

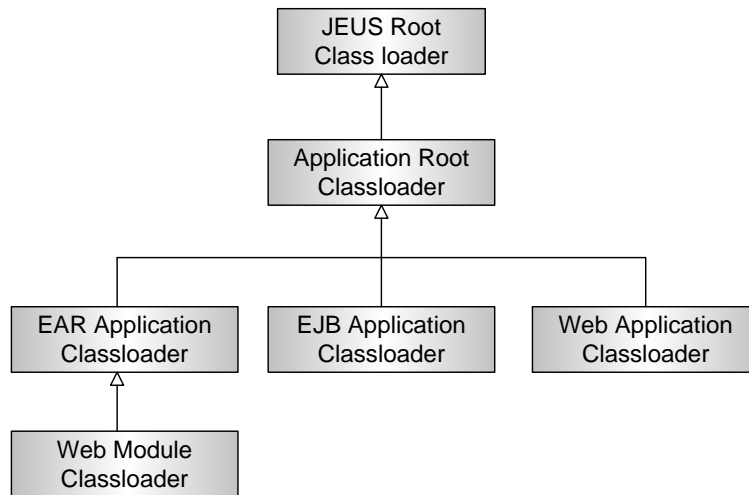
JEUS 5以下のバージョンでは、基本的に共有クラス・ローダー(Shared Class Loader)(以下、Shared方式)を使用するため、下位互換性のためにShared方式を継続して提供しています。しかし、Shared方式の使用は推奨していないため、デフォルトではIsolated方式が使用されます。

本ガイドでは、共有クラス・ローダーについては説明していません。

JEUSはアプリケーション間でクラスが重複することで発生する問題を防ぐために、アプリケーションごとに異なるクラス・ローダーを使用します。これを**独立したクラス・ローダー(Isolated Class Loader)**(以下、Isolated方式)といいます。Java EE標準では基本的にこの方式を使用することを推奨しており、JEUSもそれに準拠しています。

以下は、Isolated方式の階層構造です。

[図 1.2] Isolated方式クラス・ローダーの階層構造



この構造では、各アプリケーションのWebモジュールのクラス・ローダーが、それぞれのアプリケーションのEJBクラス・ローダーの下位に存在します。そして、特定のアプリケーション・クラス・ローダーは別のアプリケーション・クラス・ローダーにクラスを要求しません。Java EE標準では、アプリケーションが他のアプリケーションのインターフェースを必要とする場合、そのインターフェースを一緒にパッケージングするように規定しているため、別のアプリケーションのクラス・ローダーを参照することはできません。

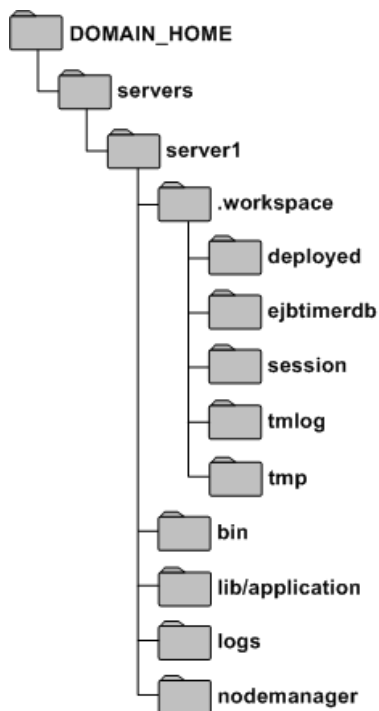
クラスを共有して使用しないため、1つのアプリケーションを再デプロイするとき、他のアプリケーションも一緒に再デプロイする必要がありません。このクラス・ローダー構造を有効に使用するためには、関連しているアプリケーション同士をEARでパッケージングして1つのアプリケーションにする作業が必要です。

1.5. サーバー・ディレクトリー構造

サーバーは、DOMAIN_HOMEの下位に自身のディレクトリーを持ちます。DOMAIN_HOMEの下位のserversディレクトリーには、ドメインに属するサーバーのディレクトリーが存在します。サーバーごとに、DOMAIN_HOME/serversの下位にサーバー名のディレクトリーを作成し、そこに運用中に必要な情報やサーバーの開始に必要な情報などを保存します。これをSERVER_HOMEといいます。

SERVER_HOMEディレクトリー構造は以下のとおりです。

【図 1.3】 サーバー・ディレクトリー構造



.workspace

JEUSが使用するサーバー別の空間であり、変更してはいけません。

以下は、下位ディレクトリーについての説明です。

| ディレクトリー | 説明 |
|------------|---|
| deployed | <p>サーバーにデプロイされたアプリケーションの圧縮を解凍したイメージ・ファイルとデプロイ時に生成されるファイルが含まれます。このディレクトリーの下位には、アプリケーションIDのディレクトリーが生成され、DASから受信したアプリケーション・ファイルが保存されます。アプリケーションがアーカイブされた場合、アプリケーションIDの下位に圧縮を解凍します(デプロイ・イメージ)。</p> <p>deployedディレクトリーの下位には「_generated_」というディレクトリーが生成され、アプリケーションをデプロイ時に生成されるファイルが含まれます</p> |
| ejbtimerdb | <p>EJBエンジンのタイマー・サービスでデータベースの設定がない場合、内部的にJEUSに組み込まれた、ファイル・ベースのデータベースのApache derbyを使用するためのディレクトリーです。タイマー・サービスについての詳細は、『JEUS EJBガイド』の「第10章 EJBタイマー・サービス」を参照してください</p> |
| session | <p>分散式セッション・サーバーで使用するファイル・データベースが含まれます。分散式セッション・サーバーは、パッシベーションされたセッションとプライマリー・セッション・サーバーが送信したバックアップ・セッションをこのファイル・データベースに保存します。詳細については、『JEUS セッション管理ガイド』の「第2章 分散セッション・サーバー」を参照してください</p> |

| ディレクトリー | 説明 |
|---------|--|
| tmlog | <p>トランザクションの回復のために残すトランザクション・ログが保存されるディレクトリーです。</p> <p>「_serverName_LOCATION_type_port_ipaddress_virtualport名」で下位ディレクトリーが生成されます。生成されるディレクトリー名のtypeとvirtualhostには、以下の値が設定されます。</p> <ul style="list-style-type: none"> – type : IPv4の場合は0が設定され、IPv6の場合には1が設定されます – virtualport : サーバー名のハッシュ値が設定されます <p>詳細は、「7.5.2. リカバリー関連のログ・ファイル」を参照してください</p> |
| tmp | <p>サーバーの運用中に一時的に保存するファイルがある場合に使用するディレクトリーです。ユーザーが特にアクセスすることはありません</p> |

bin

サーバーの起動/終了スクリプトが含まれます。JEUS_HOME/binのスクリプトと同じ機能をしますが、ドメイン名とサーバー名を設定する必要がありません。

DASの場合はstartDomainAdminServer/stopserverが、MSの場合はstartManagedServer/stopserverが含まれます。

lib/application

サーバーに適用するアプリケーション・ライブラリーが含まれます。

DOMAIN_HOME/lib/applicationにあるドメイン範囲のライブラリーと衝突する場合、このディレクトリーにあるファイルが適用されます。そのような場合は、ドメインのアプリケーション・ライブラリーと衝突したという警告メッセージが表示され、ドメイン範囲のライブラリーは該当サーバーで無視されます。lib/application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](#)』の「3.3.1. lib/application ディレクトリー」を参照してください。

logs

サーバーのランチャー・ログ、サーバー・ログ、アクセス・ログのファイルが保存されます。それぞれのログ・ファイルはロテーション・ルールによって、「ログ名_日付.log00000」で生成されます。00000は1から99999までの数字を5桁で表現したものです。ロギングについての詳細は、「[第8章 ロギング](#)」を参照してください。

nodemanager

ノード・マネージャー(Node Manager)で必要な設定情報とログが保存されるディレクトリーです。詳細は、『[JEUS ノードマネージャガイド](#)』を参照してください。

1.6. ランチャー

JEUS 7からDASとMSは両方ともランチャーにより起動されます。スクリプトでサーバーを起動するときも、DASによりMSを起動するときも、ランチャー・プロセスを実行します。ランチャーは、サーバーを起動するための準備を行い、また実際にサーバーを起動します。

ランチャーの役割は大きく次の2があります。

- DASからドメイン設定ファイルを受信する
- サーバーを起動する

ランチャーはまずDASからドメインの設定ファイルを受信します。対象ファイルは、**domain.xml**と**セキュリティ設定ファイル**です。設定ファイルの取得が完了したら、ランチャーは取得した設定ファイルをベースにサーバーJVMを起動します。サーバーを起動するときは、起動するサーバーに設定されたJVMオプションやクラスパスを設定ファイルから読み込んで、サーバーJVMを作成するときにオプションとして設定します。

ランチャー・プロセスがDASからドメインの設定ファイルを受信することに失敗した場合は、マシンにキャッシュされているドメイン設定ファイルが存在すれば、そのファイルをベースにサーバーJVMを起動します。ランチャーが設定の取得に失敗し、マシンにキャッシュされた設定ファイルも存在しない場合は、サーバーは起動されません。

ランチャー・プロセスは、上述した2つの役割のほかに、サーバー起動ログをロギングする役割もします。

サーバーの起動時に発生したログは、ランチャー・ログにもロギングされます。ランチャー・プロセスはサーバーJVMを起動し、サーバーの起動が完了するまで待機しながら、サーバーを起動時に発生したログをロギングします。もしサーバーがロガーを初期化する前に起動に失敗した場合は、ランチャー・ログで起動に失敗した原因を確認することができます。

一般に、ランチャー・プロセスはサーバーのJVMを起動し、サーバーの起動が完了したことを確認すると終了します。ランチャーにより起動されたサーバー・プロセスはバックグラウンドで実行されることに留意してください。またサーバー・プロセスは、ランチャー・プロセスの子プロセスとして起動されるため、自身のコンソール画面を持ちません。

したがって、サーバーのログをコンソール画面に出力したい場合は、サーバーの実行時に-verboseオプションを追加する必要があります。-verboseオプションを使用すると、ランチャー・プロセスはサーバーが終了するまで終了されずに、サーバーのログを画面に出力します。ランチャー・ログについての詳細は、[「8.2.2. ランチャー・ロガー」](#)を参照してください。

1.7. Managed ServerのINDEPENDENT状態

INDEPENDENTモードは、MSがDASの管理を受けずに起動および運用されるモードです。つまり、起動時にDASのURL情報がいない場合、あるいはDASが障害状況の場合にこの状態になります。

MSを開始するときは、常にDASのURL情報をオプションとして与える必要があります。ランチャーはサーバーを実行時に設定したURL情報でDASのURLを確認するので、この情報がないと、DASから設定を取得することができません。DASのURLを与えなかった場合、起動するサーバーのマシンにキャッシュされた設定ファイルがあれば、サーバーはその設定ファイルをベースにしてINDEPENDENT状態で起動されます。しかし、サーバーのマシンに設定ファイルが存在しない場合は、サーバーは起動されません。

サーバーのマシンに設定ファイルが存在するとしても、以前使用されたファイルで、DASと同期化されていない可能性が高いため、サーバーを起動するときは常にDASのURLを与える必要があります。

DASのURLを与えなくてもいいのは、DASとMSが同じマシンに存在し、ドメイン設定ファイルを取得する必要がない場合だけです。DASが障害状況でないのにURLを与えなかった場合、サーバーはINDEPENDENT状態と認識しますが、MSでグループ管理サービスが開始された後は、DASと通信が行なわれ、再びDEPENDENTモードに復旧されます。

INDEPENDENT状態になるのは、URLを与えなかった場合より、DASが障害状況である場合に多く発生します。MSを起動するとき、DASが障害状況にあれば、MSのランチャーは設定の取得に失敗することになります。DASがフェイルバックされ、INDEPENDENT状態で起動されたMSがDEPENDENTモードに復旧されると、DASの設定を同期化し、デプロイに失敗したアプリケーションを再デプロイします。

参考

DASが復旧され、設定を同期化したとしても、動的に反映されない設定が変更された場合や、すでにデプロイされサービスされているアプリケーションが変更された場合は、MSを再起動する必要があります。

以下は、MSがINDEPENDENT状態で起動された場合のログです。

```
JEUS_HOME/bin$ ./startManagedServer -domain domain1 -server server1 -u administrator -p
adminadmin -
dasurl localhost:9736
*****
- JEUS Home           : /home/admin/JEUS8/
- JEUS Base Port      :
- Java Vendor         : Sun
- Added Java Option  :
*****
===== JEUS LICENSE INFORMATION =====
=== VERSION : JEUS 8.0 (Fix#0) (8.0.0.0-b27)
=== EDITION: Enterprise (Trial License)
=== NOTICE: This license restricts the number of allowed clients.
=== Max. Number of Clients: 5
=====
[2016.08.23 22:41:15][0] [launcher-1] [Launcher-0052] Receiving the configuration failed.
Attempting to start as INDEPENDENT.
<<__Exception__>>
java.io.IOException: Connection failed. host:localhost, port:9736, virtual
id:FileTransfer
    at jeus.net.SocketProxy.getConnection(SocketProxy.java:67)
```

```

    at jeus.net.SocketProxy.getConnection(SocketProxy.java:23)
    at
jeus.server.filetransfer.ConfigurationSynchronizer.connect(ConfigurationSynchronizer.java:123)

    at
jeus.server.filetransfer.ConfigurationSynchronizer.connect(ConfigurationSynchronizer.java:106)

    at
jeus.server.filetransfer.ConfigurationSynchronizer.checkConnection(ConfigurationSynchronizer.java:148)

    at
jeus.server.filetransfer.ConfigurationSynchronizer.downloadConfigFileFromDAS(ConfigurationSynchronizer.java:187)

    at
jeus.launcher.ManagedServerLauncher.receiveConfigurationFromDas(ManagedServerLauncher.java:174)

    at jeus.launcher.ManagedServerLauncher.updateXmIs(ManagedServerLauncher.java:57)
    at jeus.launcher.ManagedServerLauncher.readDescriptor(ManagedServerLauncher.java:39)
    at jeus.launcher.Launcher.start(Launcher.java:102)
    at jeus.launcher.ManagedServerLauncher.main(ManagedServerLauncher.java:35)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:606)
    at jeus.server.Bootstrapper.callMainMethod(Bootstrapper.java:586)
    at jeus.server.Bootstrapper.callMain(Bootstrapper.java:593)
    at jeus.server.Bootstrapper.main(Bootstrapper.java:149)
    at
jeus.server.ManagedServerLauncherBootstrapper.main(ManagedServerLauncherBootstrapper.java:10)
<<__Exception__>>
[2016.08.23 22:41:16][0] [launcher-1] [Launcher-0054] Starting the server using the local
configuration.
...
[2016.08.23 22:41:21][0] [server1-1] [SERVER-0249] Successfully started the
server INDEPENDENTLY
[2016.08.23 22:41:21][2] [server1-1] [SERVER-0248] The JEUS server is RUNNING.
[2016.08.23 22:41:21][2] [server1-1] [SERVER-0401] The elapsed time to start: 4079ms.
[2016.08.23 22:41:21][2] [launcher-10] [Launcher-0034] The server[server1] initialization
completed successfully[pid : 16748].
[2016.08.23 22:41:21][0] [launcher-1] [Launcher-0040] Successfully started the server.
The server state is now RUNNING.
JEUS_HOME/bin$

```

以降、DASが正常な状態になると、MSは再び設定を同期化し、デプロイに失敗したアプリケーションがある場合は再デプロイを行います。

```

[2016.08.23 22:52:32][2] [server1-15] [Domain-0101] Domain Administration Server recovered.
server1 is communicating with the domain administration server.
[2016.08.23 22:52:32][2] [server1-15] [SERVER-0201] Successfully connected to the Domain
Administration Server(localhost:9736).
[2016.08.23 22:52:33][2] [server1-15] [SERVER-0308] Resynchronized the configuration with
Domain Administration Server.

```

第2章 JEUSの設定

本章では、ドメインにサーバーを追加する方法と、サーバーを追加するときに必要な設定について説明します。また、サーバーの設定を変更するときの動作方法についても説明します。

2.1. 概要

JEUSでは、WebAdminとコンソール・ツールを使ってサーバーの設定を変更できます。サーバーのエンジンで動作するアプリケーションがセキュリティー認証と権限チェックを必要としない場合、JEUSでセキュリティー機能を使用しないように設定します。

また上記とは別途に、下位構成要素のチューニングについても確認する必要があります。関連説明については各構成要素の章を参照してください。

2.2. サーバーの追加

本節では、WebAdminとコンソール・ツールを使ってドメインにサーバーを追加する方法と、サーバーを追加するときに必要な最小の設定について説明します。サーバーの設定についての詳細は、「[2.3. サーバーの設定](#)」を参照してください。

WebAdminを使ってサーバーを追加する場合は、サーバーの全設定を追加できます。しかし、コンソール・ツールでコマンドを使ってサーバーを追加する場合には、サーバーの全設定を追加することはできません。

2.2.1. WebAdminの使用

以下では、WebAdminを使ってサーバーを追加する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。

【図 2.1】 WebAdminのサーバー・リスト画面

The screenshot shows the WebAdmin interface. On the left is a sidebar with a menu for 'jeus_domain' containing links to Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. Below the menu is a 'システム状態' (System Status) section with a progress bar and a list of status items: 0 Failed, 0 Standby, 1 Running (highlighted with a green bar), 0 Shutdown, 0 Suspended, and 0 Other. Below this is a 'Runtime Info' section with a 'LOCK & EDIT' button and a note about domain management. At the bottom of the sidebar is a '運用マニュアル' (Operation Manual) section with links to 'Domain' and 'Server'.

The main content area is titled 'Servers' and has a 'HISTORY' dropdown in the top right. Below the title is a message: 'ドメイン内でJEUSサーバを構成するとき、各サーバの設定を行います。' (When configuring JEUS servers within a domain, configure each server). Below this is a table of servers. The table has columns: Name, Status, Pid, Need To Restart, Command, and actions (Delete, Add, Duplicate). There is one server listed: 'adminServer (*)' with status 'RUNNING(00:04:46)', pid '29464', and 'Need To Restart' set to 'false'. The 'Command' column shows 'Start' and 'Stop' buttons. The 'actions' column shows 'Delete', 'Add', and 'Duplicate' buttons.

Below the servers table is a 'Server Templates' section with a table that has a single header 'Name' and an 'Add' button. Below the table is a message: '該当する内容が存在しません。' (No matching content exists).

2. サーバーを追加するには、先に設定変更のためのロックを取得しなければなりません。サーバーを追加する前に、画面左のメニューの下部にある[LOCK & EDIT]ボタンをクリックして動的な設定変更モードに切り替えます。
3. サーバー・リスト画面で(【図 2.1】)で[Add]ボタンをクリックすると、サーバー設定画面に移動します。サーバー設定画面で、追加するサーバーの名前と「Jvm Option」を設定して[確認]ボタンをクリックします。

[図 2.2] WebAdminのサーバー設定画面 - サーバーの追加

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

Activate Changes

Undo All Changes

変更された設定を保存、またはキャンセルする機能です。

運用マニュアル

国 Domain

国 Server もっと見る

Server

HISTORY

ドメイン内で使用するJEUSサーバの詳細設定を定義します。

ヘルプ

Basic Resource Engine

Basic Info Res Ref Naming Server System Thread Pool System Logging User Logging Tm Config

動的設定 必須項目 確認 再設定

| | |
|------------------------------------|---|
| Name * | server1 サーバ名です。 |
| Log Home | JEUSサーバで生成するログのデフォルトパスを指定します。同パスが設定されていても、ロガーのファイルハンドラに設定されているパスが絶対パスの場合にはそのパスが優先されます。デフォルトログのみならず、ローテーションされたバックアップログが生成されるパスにも同様に適用されます。 |
| Node Name | サーバが属するノード名を指定します。ノードにはマシン情報、ホスト情報などが含まれます。 |
| Group | サーバを管理するためのグループを設定します。WebAdminでグループ別にサーバを管理することができます。 |
| Action On Resource Leak | [デフォルト: Warning] コンポーネント(主に、Stateless Component・Servlet/JSP、Stateless Session Bean、MDB)で利用したリソース(JCA、JDBCコネクションなど)をログギングするか、リターンするかを設定します。デフォルトはログギング(警告)です。データソース別に設定する場合、Action On Connection Leakを設定します。 |
| Data Source Remote Lookup | [デフォルト: false] リモートJVMでのデータソースルックアップを可能にします。データソースルックアップにより、リモートJVMでコネクションプールを構成して使用していた既存のスタンドアロンクライアントをサポートできます。 |
| Engine Init On Startup | [デフォルト: true] サーバに設定されているWeb、EJB、JMSエンジンなどの初期化時点を設定します。 |
| Use Web Engine | [デフォルト: true] サーバにWebエンジンを使用するか否かを設定します。 |
| Use Ejb Engine | [デフォルト: true] サーバにEJBエンジンを使用するか否かを設定します。 |
| Use Jms Engine | [デフォルト: true] サーバにJMSエンジンを使用するか否かを設定します。 |
| Managed Executor Service | サーバで有効なManagedExecutorServiceのエクスポート名を設定します。 |
| Managed Scheduled Executor Service | サーバで有効なManagedScheduledExecutorServiceのエクスポート名を設定します。 |
| Context Service | サーバで有効なContextServiceのエクスポート名を設定します。 |
| Managed Thread Factory | サーバで有効なManagedThreadFactoryのエクスポート名を設定します。 |

必要な場合、サーバーの詳細設定を設定します。この設定は必須ではありません。

[図 2.3] WebAdminのサーバー設定画面 - 詳細設定

[詳細設定](#)

[すべてを開く](#)

Server

Use MEJB

☐

クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。

Class Ftp

☐

クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。

Log Stdout To Raw Format

☒

▼ User Interceptor

Preceding Command

Jeus Classloader Append Class Path

EX

/jeus/mylib/classes:/jeus/mylib/lib/mylib.jar

Jeus Classloader Append Dirs

Profile Class Option Name

Boot Classloader Append Class Path

EX

/jeus/mylib/classes:/jeus/mylib/lib/mylib.jar

▼ Enable Interop

☐

クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。

以下は、主要設定項目についての説明です。WebAdmin画面でより詳しい説明を参照できます。

| 項目 | 説明 |
|---------------------------|---|
| Log Home | 各種ログが保存されるパスを指定します。詳細は、 「8.2. JEUSロガーの基本構造」 を参照してください |
| Node Name | 追加するサーバーが属するノードの名前を設定します。ノード・マネージャについての詳細は、『JEUS ノードマネージャガイド』を参照してください |
| Group | サーバー管理のためのグループを設定します。WebAdminでグループごとにサーバーを管理することができます |
| Action On Resource Leak | この項目の設定については、 「2.3.1.2. Action On Resource Leakの設定」 を参照してください |
| Data Source Remote Lookup | リモートJVM上でのデータソース・ルックアップを可能にします。データソース・ルックアップを利用してリモートJVMでコネクション・プールを構成し、既存のスタンドアローン・クライアントをサポートすることができます |
| Engine Init On Startup | <p>サーバー内部のエンジン(Web、EJB、JMS)の初期化時点を設定します。詳細は、「3.2.2. エンジンの初期化時点の設定」を参照してください。</p> <p>チェックされている場合、サーバーが初期化される際に内部のエンジンも一緒に初期化され、チェックされていない場合、各エンジン別にアプリ</p> |

18 JEUS サーバガイド

| 項目 | 説明 |
|------------------------------------|--|
| | ケーションがデプロイされる際に初期化されます。ただし、JMSエンジンの場合、EJBアプリケーションがMDB(Message Driven Bean)の場合にアプリケーションがデプロイされる際に初期化されます。この設定はサーバーを再起動すると適用されます |
| Use Web Engine | <p>サーバー内部のWebアプリケーションを使用するための内部エンジンの使用可否を設定します。詳細は、「3.2.1. エンジン使用可否の設定」を参照してください。</p> <p>使用しないという設定をし、Webアプリケーションをデプロイすると、デプロイは失敗します。この設定はサーバーを再起動すると適用されます</p> |
| Use Ejb Engine | サーバー内部のEJBアプリケーションを使用するための内部エンジンの使用可否を設定します。詳細は、「 3.2.1. エンジン使用可否の設定 」を参照してください。 |
| Use Jms Engine | サーバー内部のJMSサービスを使用するための内部エンジンの使用可否を設定します。詳細は、「 3.2.1. エンジン使用可否の設定 」を参照してください |
| Managed Executor Service | Concurrency Utilities for Java EE(JSR-236)仕様で使用するManagedExecutorServiceを設定します。アプリケーション・サーバーがリソースおよびコンテキストを管理するExecutorService上でタスクを実行することができます。詳細は、『 <i>JEUS 並行処理ユーティリティガイド</i> 』の「第2章 Managed Objects」を参照してください |
| Managed Scheduled Executor Service | Concurrency Utilities for Java EE(JSR-236)仕様で使用するManagedScheduledExecutorServiceを設定します。アプリケーション・サーバーがリソースおよびコンテキストを管理するScheduledExecutorService上で定期的なタスクを実行することができます。詳細は、『 <i>JEUS 並行処理ユーティリティガイド</i> 』の「第2章 Managed Objects」を参照してください |
| Context Service | Concurrency Utilities for Java EE(JSR-236)仕様で使用するContextServiceを設定します。動的プロキシを利用してタスク自体のコンテキストを維持します。詳細は、『 <i>JEUS 並行処理ユーティリティガイド</i> 』の「第2章 Managed Objects」を参照してください |
| Managed Thread Factory | Concurrency Utilities for Java EE(JSR-236)仕様で使用するManagedThreadFactoryを設定します。該当のスレッド・ファクトリーによって作成されたスレッド上で動作するタスクのコンテキストが維持されます。詳細は、『 <i>JEUS 並行処理ユーティリティガイド</i> 』の「第2章 Managed Objects」を参照してください |

4. 変更した設定内容が保存され、結果メッセージが画面上部に表示されます。

[図 2.4] のサーバー設定画面 - サーバー追加の結果

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

Activate Changes

Undo All Changes

変更された設定を保存、またはキャンセルする機能です。

運用マニュアル

Domain

Server

もっと見る

Server

HISTORY

ドメイン内で使用するJEUSサーバの詳細設定を定義します。

追加されました。

ヘルプ

BasicResourceEngine

Basic InfoRes RefNaming ServerSystem Thread PoolSystem LoggingUser LoggingTm Config

動的設定必須項目

確認再設定

| | |
|------------------------------------|---|
| Auto Generated | <input type="checkbox"/> サーバが動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスタが削除されると、該当するサーバも自動的に削除されます。 |
| Name * | server1 サーバ名です。 |
| Log Home | <input type="text"/> JEUSサーバで生成するログのデフォルトパスを指定します。同パスが設定されていても、ローガのファイルハンドラに設定されているパスが絶対パスの場合にはそのパスが優先されます。デフォルトログのみならず、ローテーションされたバックアップログが生成されるパスにも同様に適用されます。 |
| Node Name | <input type="text"/> サーバが属するノード名を指定します。ノードにはマシン情報、ホスト情報などが含まれます。 |
| Group | <input type="text"/> サーバを管理するためのグループを設定します。WebAdminでグループ別にサーバを管理することができます。 |
| Action On Resource Leak | <input type="text"/> [デフォルト: Warning] コンポーネント(主に、Stateless Component・Servlet/JSP、Stateless Session Bean、MDB)で利用したリソース(JCA、JDBCコネクションなど)をログニングするか、リターンするかを設定します。デフォルトはログニング(警告)です。データソース別に設定する場合、Action On Connection Leakを設定します。 |
| Data Source Remote Lookup | <input type="checkbox"/> [デフォルト: false] リモートJVMでのデータソースルックアップを可能にします。データソースルックアップにより、リモートJVMでコネクションプールを構成して使用していた既存のスタンドアロンクライアントをサポートできます。 |
| Engine Init On Startup | <input checked="" type="checkbox"/> [デフォルト: true] サーバに設定されているWeb、EJB、JMSエンジンなどの初期化時点を設定します。 |
| Use Web Engine | <input checked="" type="checkbox"/> [デフォルト: true] サーバにWebエンジンを使用するか否かを設定します。 |
| Use Ejb Engine | <input checked="" type="checkbox"/> [デフォルト: true] サーバにEJBエンジンを使用するか否かを設定します。 |
| Use Jms Engine | <input checked="" type="checkbox"/> [デフォルト: true] サーバにJMSエンジンを使用するか否かを設定します。 |
| Managed Executor Service | <input type="text"/> サーバで有効なManagedExecutorServiceのエクスポート名を設定します。 |
| Managed Scheduled Executor Service | <input type="text"/> サーバで有効なManagedScheduledExecutorServiceのエクスポート名を設定します。 |
| Context Service | <input type="text"/> サーバで有効なContextServiceのエクスポート名を設定します。 |

5. 基本情報画面の上部のメニューで、[Basic] > [System Thread Pool]を選択すると、System Thread Pool設定画面に移動します。設定画面で「Max」値を変更して[確認]ボタンをクリックします。

[図 2.5] サーバーの追加 - System Thread Poolの設定

The screenshot shows the 'System Thread Pool' configuration page. On the left is a sidebar with a menu for 'jeus_domain' containing options like Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. Below the menu is a 'システム状態' (System Status) section showing counts for Failed, Standby, Running, Shutdown, Suspended, and Other states. There are also buttons for 'Runtime Info', 'Activate Changes', and 'Undo All Changes'. The main area is titled 'System Thread Pool' and has a 'HISTORY' dropdown. A warning message states that the settings are for the shared thread pool used by server services, and dedicated thread pools are used if set. Below this are tabs for 'Basic', 'Resource', and 'Engine'. The 'Basic' tab is active, showing a breadcrumb trail: 'Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config'. There are buttons for '動的設定' (Dynamic Settings), '必須項目' (Required Items), '確認' (Confirm), '再設定' (Reset), and '削除' (Delete). The 'Min' value is set to 0 (default) and the 'Max' value is set to 200 (default). Below this is a '詳細設定' (Detailed Settings) section with fields for 'Keep Alive Time', 'Queue Size', 'Stuck Thread Handling' (including 'Max Stuck Thread Time', 'Action On Stuck Thread', and 'Stuck Thread Check Period'). At the bottom right are buttons for '確認' (Confirm), '再設定' (Reset), and '削除' (Delete).

変更した設定内容が保存され、結果メッセージが画面上部に表示されます。

[図 2.6] サーバーの追加 - System Thread Pool設定の保存

System Thread Pool

HISTORY

サーバーサービスが使用する共有スレッドプールに関する設定です。 サービス別にDedicatedスレッドプールを設定していない場合は、共有スレッドプールを使用することになります。

ヘルプ

変更されました。

BasicResourceEngine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

動的設定 必須項目

確認再設定削除

Min

[デフォルト: 0] スレッドプールで管理するスレッドの最小数です。

Max

200

[デフォルト: 100] スレッドプールで管理するスレッドの最大数です。

詳細設定

すべてを開く

Keep Alive Time

ms

Queue Size

Stuck Thread Handling

Max Stuck Thread Time

ms

Action On Stuck Thread

Stuck Thread Check Period

ms

確認再設定削除

6. System Thread Pool設定の保存が完了したら、次にサーバーのリスナー情報を設定します。

[Resource]>[Listener]メニューを選択すると、リスナー・リスト画面に移動します。リスナー・リスト画面で[Add]ボタンをクリックし、サーバーのデフォルト・リスナーを作成します。

[図 2.7] サーバーの追加 - デフォルト・リスナーの追加

Listeners

HISTORY

サーバのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

ヘルプ

Basic Resource Engine

Listener Jms Resource Jmx Manager Scheduler Lifecycle Invocation External Resource

動的設定 必須項目 確認 再設定 削除

Base

JNDI、セキュリティ、JMX、クラスFTPサービスなどが基本的に利用するリスナーを指定します。このとき、Listener項目で設定した名前と同様に設定する必要があります。設定していない場合は、9736ポートと基本設定を利用してリスナーの作成を試みます。

確認 再設定 削除

Listener

| Name | Listen Address | Listen Port | Add |
|------|----------------|-------------|-----|
|------|----------------|-------------|-----|

該当する内容が存在しません。

7. サーバーのデフォルト・リスナーの名前と、必要の場合はサーバーが参照するアドレス、ポート番号を入力して[確認]ボタンをクリックします。以下の画面ではListen Addressを別途入力しません。

[図 2.8] サーバーの追加 - デフォルト・リスナー情報の設定

Listener

HISTORY

サーバのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

BasicResourceEngine

ListenerJms ResourceJmx ManagerSchedulerLifecycle InvocationExternal Resource

動的設定必須項目

確認再設定

| | |
|---------------------|--------------------------|
| Name | BASE |
| Listen Address | |
| Listen Port | 9836 |
| Selectors | |
| Use Dual Selector | <input type="checkbox"/> |
| Backlog | |
| Keep Alive Timeout | ms |
| Read Timeout | ms |
| Reserved Thread Num | |

☐ Ssl

SSL属性を指定し、該当するリスナーを使用するすべてのサービスにSSLが適用されるように宣言します。

確認再設定

8. 変更した設定内容が保存され、画面上部に結果メッセージが表示されます。**リスナー**リストにリスナーが追加されたことが確認できます。

[図 2.9] サーバーの追加 - デフォルト・リスナー情報設定の保存

Listeners

HISTORY

サーバのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

追加されました。

BasicResourceEngine

ListenerJms ResourceJmx ManagerSchedulerLifecycle InvocationExternal Resource

動的設定必須項目

確認再設定削除

Base

JNDI、セキュリティ、JMX、クラスFTPサービスなどが基本的に利用するリスナーを指定します。このとき、Listener項目で設定した名前と同様に設定する必要があります。設定していない場合は、9736ポートと基本設定を利用してリスナーの作成を試みます。

確認再設定削除

Listener

| Name | Listen Address | Listen Port | Add |
|------|----------------|-------------|--------|
| BASE | | 9836 | Delete |

9. 追加したリスナーをデフォルト・リスナーとして設定するには、「Base」項目で追加したリスナーを選択します。

【図 2.10】サーバーの追加 - デフォルト・リスナーの設定

Listeners

HISTORY

サーバーのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

追加されました。

Basic Resource Engine

Listener Jms Resource Jmx Manager Scheduler Lifecycle Invocation External Resource

動的設定 必須項目 確認 再設定 削除

Base BASE

JNDI、セキュリティ、JMX、クラスFTPサービスなどが基本的に利用するリスナーを指定します。このとき、Listener項目で設定した名前と同様に設定する必要があります。設定していない場合は、9736ポートと基本設定を利用してリスナーの作成を試みます。

確認 再設定 削除

Listener

| Name | Listen Address | Listen Port | |
|------|----------------|-------------|------------|
| BASE | | 9836 | Add Delete |

【確認】ボタンをクリックすると、サーバーに変更結果が保存され、以下のようなメッセージが表示されます。設定情報は一時的に保存されるだけで、サーバーには反映されないということに注意してください。

【図 2.11】サーバーの追加 - デフォルト・リスナー設定の保存

Listeners

HISTORY

サーバーのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

変更されました。

Basic Resource Engine

Listener Jms Resource Jmx Manager Scheduler Lifecycle Invocation External Resource

動的設定 必須項目 確認 再設定 削除

Base BASE

JNDI、セキュリティ、JMX、クラスFTPサービスなどが基本的に利用するリスナーを指定します。このとき、Listener項目で設定した名前と同様に設定する必要があります。設定していない場合は、9736ポートと基本設定を利用してリスナーの作成を試みます。

確認 再設定 削除

Listener

| Name | Listen Address | Listen Port | |
|------|----------------|-------------|------------|
| BASE | | 9836 | Add Delete |

- 10 [Engine]タブで[Web Engine]、[Jms Engine]、[Ejb Engine]メニューを選択してエンジン情報を設定します。

[図 2.12] サーバーの追加 - Webエンジンの設定

Web Engine

HISTORY

Webエンジンは、J2EE Web/サーブレットアプリケーションが動作するための環境を提供します。J2EEの仕様でのWebコンテナに相当する機能です。サーバの起動時に実行され、1つのサーバには1つのWebエンジンのみサポートされます。

Basic

Resource

Engine

Web Engine | Jms Engine | Ejb Engine

Basic

Jsp Engine

Virtual Host

Web Connections

Access Log

Session Config

動的設定

必須項目

確認

再設定

| | |
|----------------------------|---|
| Attach Stacktrace On Error | <div><input type="checkbox"/></div> <div>[デフォルト: false] JEUSが返すエラーページにスタックトレースを添付するか否かを設定します。この機能は開発期間には有用ですが、運用環境では無効にする必要があります。Default Error Page項目を設定して使用する場合は、このオプションは意味がありません。</div> |
| Async Timeout Min Threads | <div></div> <div>[デフォルト: 1] Servlet 3.0の非同期サーブレットを使用する場合、タイムアウト処理を行うスレッドプールの最小数を指定します。0を指定した場合、タイムアウトが正常に動作しないことがあるため、1以上の値を設定してください。</div> |
| Properties | <div><div></div><div>key=value</div></div> <div>Webエンジンに適用される属性を設定します。</div> |
| Default Error Page | <div></div> <div>Webアプリケーションに別途のエラーページを設定していない場合に使用するエラーページを指定します。静的ページ(HTML、HTML)のみ設定でき、絶対パスで指定します。このページは転送またはリダイレクトされず、HTTP応答本文の内容として使用されます。</div> |

11. 設定が完了したら、[Activate Changes]ボタンをクリックして、これまで行ったサーバーの追加設定や設定の変更をサーバーに反映します。

[図 2.13] サーバーの追加 - 設定情報の反映(1)

The screenshot shows the 'Web Engine' configuration page. At the top, there's a 'Web Engine' header with a 'HISTORY' dropdown. Below it, a message states: 'Webエンジンは、J2EE Web/サーブレットアプリケーションが動作するための環境を提供します。J2EEの仕様でのWebコンテナに相当する機能です。サーバーの起動時に実行され、1つのサーバーには1つのWebエンジンのみサポートされます。' followed by a green '変更されました。' (Changed) button. The page has tabs for 'Basic', 'Resource', and 'Engine', with 'Engine' selected. Under 'Engine', there are sub-tabs: 'Web Engine', 'Jms Engine', and 'Ejb Engine'. Below these are more sub-tabs: 'Basic', 'Jsp Engine', 'Virtual Host', 'Web Connections', 'Access Log', and 'Session Config', with 'Basic' selected. A legend indicates '動的設定' (Dynamic Settings) with a blue icon and '必須項目' (Required Items) with a red star icon. The 'Attach Stacktrace On Error' option is unchecked, with a default value of 'false'. The 'Async Timeout Min Threads' is set to '1'. The 'Properties' section has a text area for setting properties, with a hint 'key=value'. The 'Default Error Page' section has a text area for specifying the error page. At the bottom right, there are '確認' (Confirm) and '再設定' (Reset) buttons.

[Activate Changes]ボタンをクリックすると、以下のようにサーバーへの反映を確認できるポップアップ・ウィンドウが表示されます。[確認]ボタンをクリックすると、変更内容が反映されます。

[図 2.14] サーバーの追加 - 設定情報の反映(2)

The screenshot shows a dialog box titled 'Activate Changes'. It contains the text: '変更したすべての内容をサーバーに適用します。' (Apply all changes to the server). Below this is a table with two columns: 'Description' and a text area. The text area contains the text: 'ドメイン設定の変更内容を書き込むことができます。' (You can write the domain configuration change content). At the bottom right, there are '確認' (Confirm) and 'キャンセル' (Cancel) buttons.

12 実際にDASで反映が完了すると、ドメインにサーバーが追加されます。そして、サーバーの追加設定の結果メッセージがWebAdmin画面の上部に出力されます。

[図 2.15] サーバーの追加 - 設定情報の反映(3)

The screenshot shows the JEUS WebAdmin interface. On the left, a sidebar lists the 'jeus_domain' tree with 'Servers' selected. The main area is titled 'Web Engine' and shows the configuration for 'Jms Engine'. The 'Basic' tab is active, displaying the 'Attach Stacktrace On Error' checkbox (unchecked) and the 'Async Timeout Min Threads' input field (set to 1). The 'Properties' section is empty. The 'Default Error Page' section is also empty. The top of the page shows a message indicating that the 'domain.xml' settings have been applied and the 'Web Engine' is activated.

反映に成功すると、設定変更ロックの設定を解除します。画面に再び[LOCK & EDIT]ボタンが表示されることが確認できます。ロックの設定に関する詳細は、『JEUS WebAdminガイド』を参照してください。

13 WebAdminの左のメニューで**[Servers]**を選択してサーバー・リスト照会画面に移動すると、新しく追加されたサーバーを確認できます。

[図 2.16] サーバーの追加 - 新規サーバーの確認



14 サーバー・リストで**[start]**ボタンをクリックしてサーバーを実行します。WebAdminでサーバーを起動するためには、該当のサーバーがインストールされているノードのノード・マネージャーが動作している必要があります。ノード・マネージャーについては、『JEUS ノードマネージャガイド』を参照してください。サーバーの**[start]**ボタンをクリックすると、以下のような設定画面が表示されます。サーバーの起動設定についての詳細は、「[3.1.2. Managed Serverの起動](#)」を参照してください。

[図 2.17] サーバーの追加 - 新規サーバーの実行(1)



サーバーが起動すると、画面上部で以下のようなサーバーの起動結果メッセージを確認できます。また、サーバーの状態がRUNNINGに変更されていることが確認できます。

[図 2.18] サーバーの追加 - 新規サーバーの実行(2)

2.2.2. コンソール・ツールの使用

コンソール・ツールでは、**add-server**コマンドを使ってサーバーを追加できます。しかし、このコマンドでは限られた設定しか追加できないため、サーバーを追加した後、別のコマンドで設定を変更する必要があります。エンジンの設定を変更する場合は、Webエンジン、EJBエンジン、JMSエンジンのコマンドを使用します。

参考

Webエンジン(『*JEUS リファレンスガイド*』の「4.2.8. Webエンジン関連コマンド」)、EJBエンジン(『*JEUS リファレンスガイド*』の「4.2.7. EJBエンジン関連コマンド」)、JMSエンジン(『*JEUS リファレンスガイド*』の「4.2.10. JMSエンジン関連コマンド」)の設定を変更するコマンドについては関連ガイドの説明を参照してください。

以下は、コンソール・ツールでサーバーを追加する例です。

[例 2.1] コンソール・ツールによるサーバーの追加

```
[DAS]domain1.adminServer>serverinfo
Information about Domain (domain1)
=====
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Server | Status | Node | PID | Clu | Latest | Need | Listen | Running |
|         |         | Name | ster |      | Start Time | to | Ports | Engines |
|         |         |      |      |      | / Shutdown | Restart |      |      |
|         |         |      |      |      | Time |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
| admin | RUNNING | nod | 116 | N/A | 2016-08-24 | false | base-0.0. | jms, |
| Server | (00:03:1 | e1 | 35 |      | (水)午前 |      | 0.0:9736 | ejb, web |
| (*) | 2) |      |      |      | 11:09:58 KST |      | http-serv |      |
|         |         |      |      |      |      |      | er-0.0.0.0 |      |
|         |         |      |      |      |      |      | :8088 |      |
|         |         |      |      |      |      |      | jms-0.0.0 |      |
|         |         |      |      |      |      |      | .0:9741 |      |
+-----+-----+-----+-----+-----+-----+-----+
| serve | RUNNING | nod | 118 | N/A | 2016-08-24 | false | base-0.0. | jms, |
| r1 | (00:01:4 | e1 | 08 |      | (水)午前 |      | 0.0:9836 | ejb, web |
|         | 8) |      |      |      | 11:11:22 KST |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
=====
[DAS]domain1.adminServer>>add-server server2 -addr 192.168.15.59 -baseport 9936 -node node1
-jvm "-Xmx512m -XX:MaxPermSize=128m"
Successfully performed the ADD operation for server (server2).
Check the results using "list-servers or add-server".
[DAS]domain1.adminServer>>modify-system-thread-pool server2 -max 200
Successfully performed the MODIFY operation for the system thread pool of the server (server2),
but all changes were non-dynamic. They will be applied after restarting.
Check the results using "modify-system-thread-pool server2 or show-system-thread-pool server2".
[DAS]domain1.adminServer>>serverinfo
Information about Domain (domain1)
=====
+-----+-----+-----+-----+-----+-----+-----+-----+
| Server | Status | Node | PID | Clu | Latest | Need | Listen | Running |
|         |         | Name | ster |      | Start Time | to | Ports | Engines |
|         |         |      |      |      | / Shutdown | Restart |      |      |
|         |         |      |      |      | Time |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
| admin | RUNNING | nod | 116 | N/A | 2016-08-24 | false | base-0.0. | jms, |
| Server | (01:35:1 | e1 | 35 |      | (水)午前 |      | 0.0:9736 | ejb, web |
| (*) | 2) |      |      |      | 11:09:58 KST |      | http-serv |      |
|         |         |      |      |      |      |      | er-0.0.0.0 |      |
|         |         |      |      |      |      |      | :8088 |      |
|         |         |      |      |      |      |      | jms-0.0.0 |      |
|         |         |      |      |      |      |      | .0:9741 |      |
+-----+-----+-----+-----+-----+-----+-----+
| serve | RUNNING | nod | 118 | N/A | 2016-08-24 | false | base-0.0. | jms, |
| r1 | (01:33:4 | e1 | 08 |      | (水)午前 |      | 0.0:9836 | ejb, web |
|         | 8) |      |      |      | 11:11:22 KST |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
| serve | SHUTDOWN | nod | N/A | N/A | N/A | N/A | N/A | N/A |
| r2 |         | e1 |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
=====

```



```
[DAS]domain1.adminServer>>start-server server2
The server(server2) was successfully started. The server is [RUNNING]
[DAS]domain1.adminServer>>serverinfo
Information about Domain (domain1)
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Server | Status | Node | PID | Clu | Latest | Need | Listen | Running |
|         |         | Name |     | ster | Start Time | to | Ports | Engines |
|         |         |     |     |     | / Shutdown | Restart |      |         |
|         |         |     |     |     | Time      |      |      |         |
+-----+-----+-----+-----+-----+-----+-----+-----+
| admin | RUNNING | nod | 116 | N/A | 2016-08-24 | false | base-0.0. | jms, |
| Server | (01:35:3 | e1 | 35 | | (水)午前 | | 0.0:9736 | ejb, web|
| (*) | 8) | | | | 11:09:58 KST | | http-serv | |
| | | | | | | | er-0.0.0.0 | |
| | | | | | | | :8088 | |
| | | | | | | | jms-0.0.0 | |
| | | | | | | | .0:9741 | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| serve | RUNNING | nod | 118 | N/A | 2016-08-24 | false | base-0.0. | jms, |
| r1 | (01:34:1 | e1 | 08 | | (水)午前 | | 0.0:9836 | ejb, web|
| | 3) | | | | 11:11:22 KST | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| serve | RUNNING | nod | 136 | N/A | 2016-08-24 | false | base-192. | jms, |
| r2 | (00:00:0 | e1 | 60 | | (水)午前 | | 168.15.59: | ejb, web|
| | 9) | | | | 12:45:27 KST | | 9936 | |
+-----+-----+-----+-----+-----+-----+-----+-----+
=====
```


2.3. サーバーの設定

WebAdminまたはコンソール・ツールを使用してサーバーを設定できます。コンソール・ツールを使用する場合は、WebAdminのように、サーバーの全設定を変更することはできません。コンソール・ツールでは、ユーザーが多く使用する主要設定だけ使用できます。コンソール・ツールで変更できない設定については、WebAdminで変更してください。

2.3.1. 基本設定

本節では、基本的な設定について説明します。ここで説明していない設定については、関連サービスに関するガイドを参照してください。

2.3.1.1. 基本設定の動的変更

サーバーの基本設定のうち、Class FTP、Use MEJB、Log Stdout To Raw Formatは動的に変更することができます。動的反映が可能な設定は、WebAdminではアイコン()により確認できます。コンソール・ツールでは、helpコマンドによりmodify-serverコマンドを実行すると照会できます。各コマンドの詳細は、『JEUS リファレンスガイド』を参照してください。

WebAdminの使用

以下では、WebAdminを使って動的変更が可能な基本設定を変更する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。
2. サーバー・リストでサーバーを選択すると、サーバー設定画面に移動します。設定画面で**[Basic]**タブを選択します。
3. 設定を変更するには、先にロックを取得しなければなりません。画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックして動的な設定変更モードに切り替えます。
4. **詳細設定**で「**Use MEJB**」、「**Log Stdout To Raw Format**」項目を設定します。
5. 設定が完了したら、**[確認]**ボタンをクリックします。設定情報をサーバーに反映するために、**[Activate Changes]**ボタンをクリックします。
6. 設定変更の結果メッセージが画面上部に出力されます。「**Use MEJB**」、「**Log Stdout To Raw Format**」の設定が変更され、サーバーに反映されていることが確認できます。

コンソール・ツールの使用

以下は、コンソール・ツールを使用して動的変更が可能な基本設定を変更する例です。

[例 2.2] コンソール・ツールによる動的な設定変更

```
[DAS]domain1.adminServer>>modify-server server2
Shows the current configuration.
server (server2)
=====
+-----+-----+
| Node           | node1           |
| JVM Configs    | -Xmx512m -XX:MaxPermSize=128m |
| Action On Resource Leak | WARNING        |
| Stdout to Raw Format | true           |
| MEJB           | false          |
| Class FTP      | false          |
| Server Log Home Directory | none          |
+-----+-----+
=====

[DAS]domain1.adminServer>>modify-server server2 -logStdoutToRawFormat false -mejb true
-classFtp true
Successfully performed the MODIFY operation for server (server2), but all changes were
non-dynamic. They will be applied after restarting.
Check the results using "list-servers server2 or modify-server server2".
```

```
[DAS]domain1.adminServer>>modify-server server2
Shows the current configuration.
server (server2)
=====
+-----+-----+
| Node | node1 |
| JVM Configs | -Xmx512m -XX:MaxPermSize=128m |
| Action On Resource Leak | WARNING |
| Stdout to Raw Format | false |
| MEJB | true |
| Class FTP | true |
| Server Log Home Directory | none |
+-----+-----+
=====
```

2.3.1.2. Action On Resource Leakの設定

Action On Resource Leakは、サーバーで使用されるリソースが閉じられたかどうかを確認して、設定された動作を行い、ユーザーにリソース・リークがあることを通知する機能です。サーバーでこのような機能を担当するのが**Invocation Manager**です。

Invocation Managerは、サーバーでServlet/JSP、EJBステートレス・セッションBean、MDBなどのステートレス・メソッドを呼び出す間に使用する外部リソースのJDBCコネクションとWebtコネクションを追跡し、コネクションが閉じられていない場合、モードによって使用するリソースに対しロギングを残すか、リソースを返す処理を行います。次の3つのモードから1つを選択できます。

| モード | 説明 |
|-----------|---|
| NoAction | 返されていないリソースがあっても何の動作も行わない |
| Warning | コンポーネントを呼び出した後に返されていないリソースに対してログを残す(デフォルト値) |
| AutoClose | コンポーネントを呼び出した後に返されていないリソースに対してログを残して閉じる |

WebAdminの使用

以下は、WebAdminを使ってAction On Resource Leak項目を設定する例です。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバー・リストでサーバーを選択すると、サーバー設定画面に移動します。設定画面で**[Basic]**タブを選択します。
2. 設定を変更するには、先にロックを取得しなければなりません。画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックして動的な設定変更モードに切り替えます。
3. 「**Action On Resource Leak**」項目の設定を変更して**[確認]**ボタンをクリックします。

4. 設定情報をサーバーに反映するには、**[Activate Changes]**ボタンをクリックします。

画面上部に結果メッセージが出力され、サーバーのAction On Resource Leak設定が変更されたことが確認できます。運用中のサーバーのAction On Resource Leak設定を変更すると、設定の変更は成功したが、サーバーに適用できないというメッセージが出力されます。変更したAction On Resource Leak設定を適用するには、サーバーを再起動する必要があります。

コンソール・ツールの使用

以下は、コンソール・ツールでmodify-serverコマンドを使ってAction On Resource Leak設定を変更する例です。

1. サーバーを起動する前に、**modify-server**コマンドにより設定を変更するサーバーの現在設定を照会します。

```
[DAS]domain1.adminServer>modify-server server2
Shows the current configuration.
server (server2)
=====
+-----+-----+
| Node           | node1 |
| JVM Configs    | -Xmx512m -XX:MaxPermSize=128m |
| Action On Resource Leak | WARNING |
| Stdout to Raw Format | false |
| MEJB           | true  |
| Class FTP      | true  |
| Server Log Home Directory | none |
+-----+-----+
=====
```

2. server1のAction On Resource Leak設定をAutoCloseに設定します。

```
[DAS]domain1.adminServer>modify-server server2 -actionOnResourceLeak AutoClose
Successfully performed the MODIFY operation for server (server2).
Check the results using "list-servers server2 or modify-server server2".
```

参考

動的に反映されないオプションは、サーバーを起動する前に変更する必要があります。運用中のサーバーでそのようなオプションの設定を変更した場合は、動的に反映されないため、サーバーを再起動しなければなりません。

3. **modify-server**または**list-servers**コマンドを使って、変更した設定が正しく反映されているか確認します。

```
[DAS]domain1.adminServer>modify-server server2
Shows the current configuration.
```

```
server (server2)
=====
+-----+-----+
| Node           | node1           |
| JVM Configs    | -Xmx512m -XX:MaxPermSize=128m |
| Action On Resource Leak | AUTO_CLOSE     |
| Stdout to Raw Format | false          |
| MEJB           | true            |
| Class FTP      | true            |
| Server Log Home Directory | none          |
+-----+-----+
=====
```

4. 「server2」を起動します。

```
[DAS]domain1.adminServer>start-server server2
The server(server2) was successfully started. The server is [RUNNING]
```

参考

コンソール・ツールまたはWebAdminでstart-serverコマンドを使用するためには、SSHを使用できる環境である必要があり、またノードが設定されている必要があります。ノード・マネージャーについての詳細は、『JEUS ノードマネージャガイド』を参照してください。

5. 「server2」のAction On Resource Leak設定を元の値のWARNINGに変更します。

サーバーが運用中であるため、設定の変更のみ行なわれ、実際にサーバーには反映されません。変更した設定をサーバーに反映するには、サーバーを再起動する必要があります。

```
modify-server server2 -actionOnResourceLeak Warning
Successfully performed the MODIFY operation for server (server2), but all changes
were non-dynamic. They will be applied after restarting.
Check the results using "list-servers server2 or modify-server server2".
```

6. modify-serverまたはlist-serversコマンドで変更した設定を確認します。

```
[DAS]domain1.adminServer>modify-server server2
Shows the current configuration.
server (server2)
=====
+-----+-----+
| Node           | node1           |
| JVM Configs    | -Xmx512m -XX:MaxPermSize=128m |
| Action On Resource Leak | WARNING         |
| Stdout to Raw Format | false          |
| MEJB           | true            |
| Class FTP      | true            |
| Server Log Home Directory | none          |
+-----+-----+
=====
```

```
+-----+
=====
```

参考

上記で使ったコンソール・ツールのコマンドの詳細については、『*JEUS リファレンスガイド*』の「4.2.3. サーバー管理関連コマンド」を参照してください。

2.3.1.3. Jvm Config設定

Jvm Configは、サーバーを実行するために個別のJVMに追加するパラメータを宣言するときに使われます。ここに設定した値をランチャー・プロセスがサーバーを起動する前に読み込み、サーバーJVMを作成するときにパラメータとして追加します。指定可能なJEUSパラメータのリストは、『*JEUS リファレンスガイド*』の「1.2. サーバーのシステム・プロパティ」を参照してください。標準のJVMパラメータも設定できます。

サーバーに適用するJVMオプションやシステム・プロパティの外に、JEUSで提供するシステム・プロパティもここに設定できます。主にJVMメモリーやオプションを設定します。これらの値はサーバーの運用環境に合う適切な値を設定する必要があります。Jvm Config関連の追加説明は、『*JEUS ドメインガイド*』の「3.6.2. サーバーのJVM設定の変更」を参照してください。

注

サーバーが運用中の場合は、この値を変更してもサーバーに反映されません。動的設定が可能な項目ではないため、運用中のサーバーでこの値を変更した場合は、サーバーを再起動して反映する必要があります。

WebAdminの使用

以下では、WebAdminを使ってJvm Configを設定する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバー・リストで設定情報を変更するサーバー(server1)を選択すると、サーバー設定画面に移動します。設定画面で**[Basic]**タブを選択します。
2. 設定を変更するには、先にロックを取得しなければなりません。画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックして動的な設定変更モードに切り替えます。
3. 「**Jvm Option**」項目の設定を変更して**[確認]**ボタンをクリックすると、変更した設定内容が保存され、結果メッセージが出力されます。
4. 設定情報をサーバーに反映するには、**[Activate Changes]**ボタンをクリックします。

画面上部に設定反映の結果メッセージが出力され、サーバーのJVM設定が変更されたことが確認できます。運用中のサーバーのJvm Configを変更した場合、設定は変更されたが変更したJVMの設定はサーバーに適用できないというメッセージが表示されます。変更したJVM設定をサーバーに反映するには、サーバーを再起動する必要があります。

コンソール・ツールの使用

コンソール・ツールのmodify-server、add-jvm-option、modify-jvm-option、remove-jvm-optionを使ってサーバーのJVM設定を変更できます。add-jvm-option、modify-jvm-option、remove-jvm-optionコマンドについての詳細は、『JEUS ドメインガイド』の「3.6.2. サーバーのJVM設定の変更」を参照してください。

以下は、コンソール・ツールでmodify-serverコマンドを使ってサーバーのJVM設定を変更する例です。

1. サーバーを起動する前に、**modify-server**コマンドにより設定を変更するサーバーの現在の設定を照会します。

```
[DAS]domain1.adminServer>modify-server server1
Shows the current configuration.
server (server1)
=====
+-----+-----+
| Node                | node1 |
| Action On Resource Leak | WARNING |
| Stdout to Raw Format  | true  |
| MEJB                 | false |
| Class FTP            | false |
| Server Log Home Directory | none  |
+-----+-----+
=====
```

2. server1にJVMオプションを追加します。JVMの最大ヒープ・メモリーを512MBに設定し、最大永続メモリーを128MBに設定します。

```
[DAS]domain1.adminServer>modify-server server1 -jvmOptions "-Xmx512m
-XX:MaxPermSize=128m"
Successfully performed the MODIFY operation for server (server1).
Check the results using "list-servers server1 or modify-server server1".
```

参考

動的反映が不可能なオプションは、サーバーを起動する前に変更する必要があります。運用中のサーバーでそのようなオプションの設定を変更した場合は、動的に反映されないため、サーバーを再起動しなければなりません。

3. **modify-server**または**list-servers**コマンドにより、変更した設定が正しく反映されているか確認します。

```
[DAS]domain1.adminServer>modify-server server1
Shows the current configuration.
server (server1)
=====
+-----+-----+
| Node                | node1                |
| JVM Configs         | -Xmx512m -XX:MaxPermSize=128m |
| Action On Resource Leak | WARNING              |
| Stdout to Raw Format  | true                 |
| MEJB                | false                |
| Class FTP           | false                |
| Server Log Home Directory | none                 |
+-----+-----+
=====
```

4. 「server1」を起動します。

```
[DAS]domain1.adminServer>start-server server1
The server(server1) was successfully started. The server is [RUNNING]
```

参考

コンソール・ツールまたはWebAdminでstart-serverコマンドを使用するためには、SSHを使用できる環境である必要があり、またノードが設定されている必要があります。ノード・マネージャーについての詳細は、『JEUS ノードマネージャガイド』を参照してください。

5. 「server1」にJVMオプションを追加します。

サーバーでOutOfMemoryErrorが発生したときに、ヒープ・ダンプ・ファイルを残すオプションを追加しました。この設定は、サーバーが運用中であるため設定の変更だけが行なわれ、実際にサーバーには反映されません。変更した設定をサーバーに反映するには、サーバーを再起動する必要があります。

```
[DAS]domain1.adminServer>modify-server server1 -jvmOptions
"-XX:+HeapDumpOnOutOfMemoryError"
Successfully performed the MODIFY operation for server (server1), but all changes
were non-dynamic. They will be applied after restarting.
Check the results using "list-servers server1 or modify-server server1".
```

6. modify-serverまたはlist-serversコマンドで変更した設定を確認します。

```
[DAS]domain1.adminServer>modify-server server1
Shows the current configuration.
server (server1)
=====
+-----+-----+
| Node                | node1                |
+-----+-----+
```


| | | |
|---------------------------|---------------------------------|--|
| JVM Configs | -Xmx512m -XX:MaxPermSize=128m, | |
| | -XX:+HeapDumpOnOutOfMemoryError | |
| +-----+ | | |
| Action On Resource Leak | WARNING | |
| +-----+ | | |
| Stdout to Raw Format | true | |
| +-----+ | | |
| MEJB | false | |
| +-----+ | | |
| Class FTP | false | |
| +-----+ | | |
| Server Log Home Directory | none | |
| +-----+ | | |
| ===== | | |

参考

上記で利用したコンソール・ツールのコマンドの詳細については、『*JEUS リファレンスガイド*』の「4.2.3. サーバー管理関連コマンド」の内容を参照してください。

2.3.1.4. クラス・パスの設定

本節では、サーバーに付加的なクラスパスを追加する方法を説明します。この設定はWebAdminでのみ行えます。

注

サーバーが運用中の場合は、この値を変更してもサーバーに反映されません。動的設定が可能な項目ではないため、運用中のサーバーでこの値を変更した場合は、サーバーを再起動して反映する必要があります。

WebAdminの使用

以下では、WebAdminを使ってサーバーにクラスパスを追加する設定を変更する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバー・リストで設定するサーバー(server1)を選択すると、サーバー設定画面に移動します。設定画面で**[Basic]**タブを選択します。
2. 設定を変更するには、先にロックを取得しなければなりません。画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックして動的な設定変更モードに切り替えます。
3. **詳細設定**で**User Interceptor**の下位項目にクラスパスを追加します。設定が終われば、**[確認]**ボタンをクリックします。

4. 設定情報をサーバーに反映するには、**[Activate Changes]**ボタンをクリックします。以下のように、画面上部に設定の変更結果メッセージが出力されます。サーバーにUser Interceptor設定が追加されたことが確認できます。

運用中のサーバーのUser Interceptor設定を変更すると、設定の変更は成功したが、変更したUser Interceptor設定をサーバーに適用できないというメッセージが出力されます。変更したUser Interceptor設定を適用するには、サーバーを再起動する必要があります。

2.3.2. リスナーの設定

本節では、サーバーで使用するネットワーク・リスナーの設定について説明します。

ネットワーク・リスナーの設定は、サーバーで実行されるシステム・サービスや各種のエンジンによって参照および使用されます。このリスナーはポート統合サービスが適用されているため、異なるサービスやエンジンで1つのリスナーを共有して使用するように設定することもできます。極端な例として、1つのデフォルト・リスナーだけを開いて、すべてのサービスをこのリスナーを通じてサービスすることも可能です。

注

サーバーが運用中の場合は、この値を変更してもサーバーに反映されません。動的設定が可能な項目ではないため、運用中のサーバーでこの値を変更した場合は、サーバーを再起動して反映する必要があります。

WebAdminの使用

以下では、WebAdminを使ってデフォルト・リスナーを設定する手順について説明します。デフォルト・リスナーは、設定されたリスナーのうち1つを選択します。JEUSの最も基本的なシステム・サービス、Webadminなどによって使用されます。

[図 2.19] デフォルト・リスナーの設定

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

1 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル

Domain

Server もっと見る

Listeners

HISTORY

サーバのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

Basic Resource Engine

Listener Jms Resource Jmx Manager Scheduler Lifecycle Invocation External Resource

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

Base BASE

JNDI、セキュリティ、JMX、クラスFTPサービスなどが基本的に利用するリスナーを指定します。このとき、Listener項目で設定した名前と同様に設定する必要があります。設定していない場合は、9736ポートと基本設定を利用してリスナーの作成を試みます。

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

Listener

| Name | Listen Address | Listen Port | |
|-------------|----------------|-------------|--------|
| BASE | 0.0.0.0 | 9736 | Delete |
| http-server | 0.0.0.0 | 8808 | Delete |
| jms-jmstest | 0.0.0.0 | 9741 | Delete |

新しいリスナーは[LOCK & EDIT]ボタンをクリックし、[Add]ボタンを押して追加することができます。個別リスナーの基本的な設定は、他のシステム・サービスやエンジンで参照する名前を設定し、他のリスナーと重複しないようにポート番号を設定します。

[図 2.20] 個別リスナーの設定

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

1 Shutdown

0 Suspended

0 Other

Runtime Info

Activate Changes

Undo All Changes

変更された設定を保存、またはキャンセルする機能です。

運用マニュアル

Domain

Server もっと見る

Listener

HISTORY

サーバのJEUSシステムが使用するソケットリスナーおよびソケットコネクションリクエスト処理に必要な各種属性を指定します。

ヘルプ

Basic Resource Engine

Listener Jms Resource Jmx Manager Scheduler Lifecycle Invocation External Resource

動的設定 必須項目 確認 再設定

Name *

TestListener

統合ポートリスナーを示す固有のIDを設定します。別の設定でこのリスナーを参照するときに使用します。

Listen Address

複数のIPが割り当てられている場合、統合ポートリスナーがバインドされるサービスIPアドレスを設定します。設定していない場合は、すべてのIPアドレスを利用してサービスが行われます。仮想マルチキャストを使用する場合は、ベースリスナーのIPアドレスを必ず指定します。

Listen Port

12345

[デフォルト: 9736] 統合ポートリスナーのポートを設定します。設定されていない場合は常に9736を使用するため、複数のサーバを同じマシンで使用する場合には必ず指定します。

Selectors

1

[デフォルト: 1] 統合ポートリスナーのI/O処理に使用するセクタ数を設定します。一般的にCPUのコア数で設定します。デフォルト値は1です。

Use Dual Selector

[デフォルト: false] 統合ポートリスナーの読み書きに使用するセクタを分離するか否か設定します。大量の読み書きが同時に必要な場合に設定してください。

Backlog

[デフォルト: 128] 統合ポートリスナーのバックログ値を指定します。

Keep Alive Timeout

ms

KeepAlive形式のクライアントが設定した時間の間何の要求もない場合に切断します。ベースリスナーには設定することを推奨しません。

Read Timeout

ms

[デフォルト: 30000] 新しく受け入れたソケットに対し、read回数がブロックされる時間を設定します。このソケットが目的とするサービスに割り当てられるため、プロトコルの判別時から、プロトコルを処理できるサービスにソケットが渡された後にもこのタイムアウトは継続して適用されます。

Reserved Thread Num

[デフォルト: 0] ポート統合サービスを処理するためのスレッドプールに関する追加設定です。基本的にシステムスレッドプールを使用しますが、特にこのサービスのためのスレッドを事前に割り当てたい場合のみ設定します。設定した値と他サービスの値との合計が、システムスレッドプールの最大値を超えないようにします。

以下は、リスナーに適用するSSL(Secure Socket Layer)を設定する方法です。リスナーを使用するサービスやエンジンがSSLでサービスを行います。(例: https)

[図 2.21] 個別リスナーのSSL設定

 ☒

SSL属性を指定し、該当するリスナーを使用するすべてのサービスにSSLが適用されるように宣言します。

| | |
|----------------------------|---|
| Client Auth | <div><div></div><div>[デフォルト: Unnecessary] クライアントの認証可否を設定します。</div></div> |
| SSL Protocol | <div><div></div><div>暗号化および復号化に使用されるSSLプロトコルを指定します。</div></div> |
| Cipher Suite | <div><div></div><div>SSLハンドシェイクの実行後、実際にデータを送信するときに使用する暗号スイートを指定します。</div></div> |
| Keystore File | <div><div></div><div>サーバの秘密鍵と、それにマッチングされる証明書を保存するファイルを指定します。Webサーバとは違って、PEM形式の秘密鍵ファイルはサポートしません。Javaキーストア形式のファイルを使用することをお勧めします。</div></div> |
| Keystore Pass | <div><div></div><div>入力</div><div>キーストアファイルを開くための暗号値です。この値を暗号化して保存するときは、{algorithm}ciphertextの形式で記述します。ex) {DES}FQrLbQ/D8O1DVS71L28rw==</div></div> |
| Keystore Keypassword | <div><div></div><div>入力</div><div>キーストアファイルに保存されたサーバ秘密鍵の暗号値です。この値を暗号化して保存するときは、{algorithm}ciphertextの形式で記述します。ex) {DES}FQrLbQ/D8O1DVS71L28rw==</div></div> |
| Keystore Type | <div><div></div><div>[デフォルト: JKS] キーストアファイルのタイプを指定します。</div></div> |
| Key Management Algorithm | <div><div></div><div>キーストアに保存するキーに対する管理アルゴリズムを設定します。</div></div> |
| Key Alias | <div><div></div><div>SSLを使用するときに用いられるサーバ証明書のキーエイリアス名を指定します。</div></div> |
| Truststore File | <div><div></div><div>サーバの証明書を保存するファイルを指定します。SSLサーバ認証機能、クライアント認証機能などに必要な証明書を保存します。Webサーバとは違って、PEM形式の証明書ファイルはサポートしません。Javaキーストア形式のファイルを使用することをお勧めします。</div></div> |
| Truststore Pass | <div><div></div><div>入力</div><div>トラストストアファイルを開くための暗号値です。この値を暗号化して保存するときは、{algorithm}ciphertextの形式で記述します。ex) {DES}FQrLbQ/D8O1DVS71L28rw==</div></div> |
| Truststore Type | <div><div></div><div>[デフォルト: JKS] トラストストアのタイプを指定します。</div></div> |
| Trust Management Algorithm | <div><div></div><div>トラストストアに保存するトラストに対する管理アルゴリズムを設定します。</div></div> |
| Crl File | <div><div></div><div>証明書失効リスト(Certificate Revocation Lists)を保存するファイルを指定します。</div></div> |

確認

再設定

2.3.3. スレッド・プールの設定

本節では、サーバーで使用する共用スレッド・プールの設定について説明します。

サーバーで使用されるサービスが専用のスレッド・プールを設定して使用しない限り、共用スレッド・プールを使用します。共用スレッド・プールを使用するサービスとして、トランザクション・サービス、JNDIサービス、スケジューラー・サービスがあります。これらのサービスは、設定によって共用スレッド・プールを使用することも、専用スレッド・プールを使用することもできます。共用スレッド・プールを使用する場合は、サービスで使用する最小スレッド数をあらかじめ割り当てることができます。

参考

アプリケーションの要求を処理するエンジンでは共用スレッド・プールを使用しません。サービスで専用スレッド・プールを使用する方法については、「[第4章 JNDIネーミング・サーバー](#)」、「[第7章 トランザクション・マネージャー](#)」、および『JEUS スケジューラガイド』を参照してください。

WebAdminの使用

以下では、WebAdminを使ってスレッド・プールを設定する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバー・リストでサーバーを選択すると、サーバー設定画面に移動します。設定画面で**[Basic]**タブを選択し、下位の**[System Thread Pool]**メニューを選択します。

2. システム・スレッド・プール設定を変更するには、先に設定を変更するためのロックを取得しなければなりません。

ロックを設定する前には、現在設定されているシステム・スレッド・プールの情報を確認できます。設定を変更するには、画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックしてロックを取得します。ロックを取得すると、システム・スレッド・プール設定を変更できる画面に切り替わります。

3. スレッド・プールの「**Max**」と「**Keep Alive Time**」の設定を変更して**[確認]**ボタンをクリックします。

以下は、主要設定項目についての説明です。

- 基本設定

| 項目 | 説明 |
|-----|---|
| Min | スレッド・プールで管理するスレッドの最小数です。必要なときにスレッドを作成します(デフォルト値: 0) |
| Max | スレッド・プールで管理するスレッドの最大数です(デフォルト値: 100) |

- 詳細設定

| 項目 | 説明 |
|-----------------|---|
| Keep Alive Time | スレッド・プールで使用しないスレッドを自動で削除する時間を設定します。 Minに設定した値以上のスレッドが、設定した時間の間使用されない場合、自動でスレッド・プールから削除されます。 共用スレッド・プールの場合、デフォルト値は5分です(単位: ms) |
| Queue size | スレッド・プールが処理するワーク・キューのサイズを指定します(デフォルト値: 4096個) |

– Stuck Thread Handling

スレッドが一定時間以上継続して占有された状態である場合、そのスレッドに対して特定のアクションを行うための設定です。

| 項目 | 説明 |
|---------------------------|---|
| Max Stuck Thread Time | スレッドが占有されたと判断する基準になる設定です。設定した時間以上占有された場合、そのスレッドをスタック・スレッドとみなします (デフォルト値 : 1時間、単位: ms) |
| Action On Stuck Thread | 占有されたと判断されたスレッドに対して行うアクションを設定します。次の値を設定することができます。共用スレッド・プールのデフォルト値は IgnoreAndReplace です – None : スタック・スレッドに何のアクションも取りません – Interrupt : スタック・スレッドにインターラプト信号を送ります – IgnoreAndReplace : スタック・スレッドを無視して新しいスレッドに交代します |
| Stuck Thread Check Period | スレッドがスタック状態になったかチェックする周期を設定します (デフォルト値 : 5分、単位: ms) |

4. 設定情報をサーバーに反映するために、**[Activate Changes]**ボタンをクリックします。画面上部に結果メッセージが出力され、サーバーのシステム・スレッド・プール設定が変更されたことが確認できます。

コンソール・ツールの使用

以下は、コンソール・ツールで共用スレッド・プールを照会および変更する例です。

参考

例で使用されたコマンドについての詳細は、『*JEUS リファレンスガイド*』の「4.2.5. スレッド管理関連コマンド」を参照してください。サービスで使用するスレッド・プールの設定については、各サービスのマニュアルを参照してください。

以下は、スレッド・プールのmaxを100から200に変更し、keep alive timeを5分から10分に変更する例です。

```
[DAS]domain1.adminServer>show-system-thread-pool server1
Shows the current configuration.
the system thread pool of the server (server1)
=====
+-----+-----+
| Min                | 0                |
| Max                | 100              |
| Keep-Alive Time    | 300000           |
| Queue Size         | 4096             |
| Max Stuck Thread Time | 3600000          |
| Action On Stuck Thread | IGNORE_AND_REPLACE |
| Stuck Thread Check Period | 300000           |
| Reserved Threads for the Service transaction | 0                |
| Reserved Threads for the Service namingserver | 0                |
+-----+-----+
=====

[DAS]domain1.adminServer>modify-system-thread-pool server1 -max 200 -keep 600000
Successfully performed the MODIFY operation for the system thread pool of the server (server1).
Check the results using "modify-system-thread-pool server1 or show-system-thread-pool server1".
[DAS]domain1.adminServer>show-system-thread-pool server1
Shows the current configuration.
the system thread pool of the server (server1)
=====
+-----+-----+
| Min                | 0                |
| Max                | 200              |
| Keep-Alive Time    | 600000           |
| Queue Size         | 4096             |
| Max Stuck Thread Time | 3600000          |
| Action On Stuck Thread | IGNORE_AND_REPLACE |
| Stuck Thread Check Period | 300000           |
| Reserved Threads for the Service transaction | 0                |
| Reserved Threads for the Service namingserver | 0                |
+-----+-----+
=====
```


以下は、JNDIサービスで、共用スレッド・プールでスレッドをあらかじめ割り当てるように設定する例です。

```
[DAS]domain1.adminServer>show-system-thread-pool server1
Shows the current configuration.
the system thread pool of the server (server1)
=====
+-----+-----+
| Min                | 0                |
| Max                | 200              |
| Keep-Alive Time    | 600000           |
| Queue Size         | 4096             |
| Max Stuck Thread Time | 3600000          |
| Action On Stuck Thread | IGNORE_AND_REPLACE |
| Stuck Thread Check Period | 300000          |
| Reserved Threads for the Service transaction | 0                |
| Reserved Threads for the Service namingserver | 0                |
+-----+-----+
=====

[DAS]domain1.adminServer>modify-system-thread-pool server1 -service namingserver -r 10
Successfully performed the MODIFY operation for The namingserver thread pool of the server
(server1).., but all changes were non-dynamic. They will be applied after restarting.
Check the results using "show-system-thread-pool server1 -service namingserver or
modify-system-thread-pool server1 -service namingserver".

[DAS]domain1.adminServer>show-system-thread-pool server1 -service namingserver
Shows the current configuration.
the system thread pool of the server (server1)
=====
+-----+-----+
| Min                | 0                |
| Max                | 200              |
| Keep-Alive Time    | 600000           |
| Queue Size         | 4096             |
| Max Stuck Thread Time | 3600000          |
| Action On Stuck Thread | IGNORE_AND_REPLACE |
| Stuck Thread Check Period | 300000          |
| Reserved Threads for the Service transaction | 0                |
| Reserved Threads for the Service namingserver | 10               |
+-----+-----+
=====

[DAS]domain1.adminServer>modify-system-thread-pool server1 -service namingserver
Shows the current configuration.
The namingserver thread pool of the server (server1).
=====
+-----+-----+
| Reserved Threads for the Service namingserver | 10               |
+-----+-----+
=====
```

2.3.4. ライフサイクル呼び出しの設定

JEUSでは、サーバーのライフサイクルに合わせて目的の作業を行えるように、ライフサイクル呼び出し機能を提供しています。サーバーは起動・終了プロセスの各段階でユーザーが設定したイベントを呼び出します。ライフサイクル呼び出しは、WebAdminを使ってのみ設定できます。

注

サーバーが運用中の場合は、この値を変更してもサーバーに反映されません。動的設定が可能な項目ではないため、運用中のサーバーでこの値を変更した場合は、サーバーを再起動して反映する必要があります。

以下は、ライフサイクル呼び出しに一般Javaクラスを登録する例です。このクラスを、サーバーのライフサイクルに合わせて呼び出そうとする対象サーバーのSERVER_HOME/lib/applicationに格納します。

[例 2.3] <<LifeCycleTester.java>>

```
package lifecycle;

public class LifeCycleTester {
    public void boot() {
        System.out.println("Boot");
        // do somethig
    }

    public void beforeDeploy() {
        System.out.println("Before Deploy");
        // do somethig
    }

    public void afterDeploy() {
        System.out.println("After Deploy");
        // do somethig
    }

    public void ready() {
        System.out.println("Ready");
        // do somethig
        try {
            System.out.println("Sleeping for 15 seconds ....");
            Thread.sleep(15000L);
        } catch (Exception e) {
            //ignored
        }
    }
}
```

```

public void beforeUndeploy() {
    System.out.println("Before Undeploy");
    // do somethig
}

public void afterUndeploy() {
    System.out.println("After Undeploy");
    // do somethig
}
}

```

WebAdminの使用

以下では、上記の例をもとに、WebAdminを使ってサーバーのライフサイクル呼び出しにクラスを設定する手順について説明します。

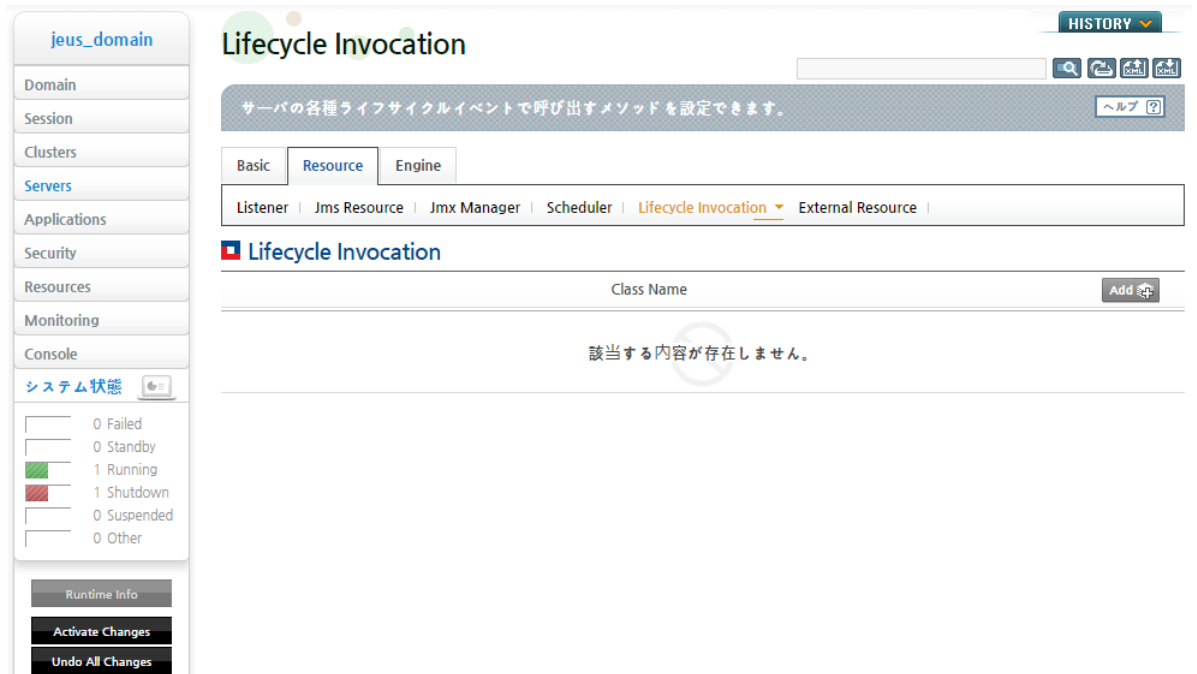
1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバー・リストで設定するサーバー(adminServer)を選択すると、サーバー設定画面に移動します。設定画面で**[Resource]**タブを選択して**[Lifecycle Invocation]**メニューを選択すると、現在設定されているライフサイクル呼び出し情報を照会する画面に移動します。

[図 2.22] WebAdminによるライフサイクル呼び出しの設定(1)



2. 画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックしてロックを取得した後、ライフサイクル呼び出しを追加するために、メイン領域の**[Add]**ボタンをクリックします。

[図 2.23] WebAdminによるライフサイクル呼び出しの設定(2)



- ライフサイクル呼び出しクラスとライフサイクル呼び出しタイプに合わせて実行するメソッドを追加します。
ライフサイクル呼び出しクラスを「test.lifecycle.invocation.LifecycleInvocation」に設定し、起動時にBOOTメソッドが呼び出されるように指定してから[確認]ボタンをクリックします。

[図 2.24] WebAdminによるライフサイクル呼び出しの設定(3)

Lifecycle Invocation

サーバの各種ライフサイクルイベントで呼び出すメソッドを設定できます。

Basic | **Resource** | Engine

Listener | Jms Resource | Jmx Manager | Scheduler | **Lifecycle Invocation** | External Resource

動的設定 * 必須項目 確認 再設定

Class Name * test.lifecycle.invocation.LifecycleInvocation EX com.tmax.event.ServerLifecycleListener
ライフサイクルイベントのコールバックメソッドが存在する完全修飾クラス名を指定します。

Invocation
クラス内の呼び出しに関する詳細情報を設定します。

Invocation Type * BOOT
メソッドが呼び出される時点を指定します。

Invocation Argument
メソッドを呼び出すときに使用する引数を指定します。

Invocation Method
呼び出しに使用されるメソッドを指定します。

Method Name * boot EX foo
メソッドの名前を指定します。

Method Param EX java.lang.String
メソッドのパラメータの完全修飾クラス名を指定します。

確認 再設定

以下は、設定項目についての説明です。

| 項目 | 説明 |
|-----------------|--|
| Invocation Type | <p>メソッドが呼び出される時点を指定します</p> <ul style="list-style-type: none"> BOOT : サーバーが起動し、エンジンが起動する前の時点です BEFORE_DEPLOY : サーバーが起動し、サーバーに登録されたアプリケーションを配布する前の時点です AFTER_DEPLOY : サーバーが起動し、サーバーに登録されたアプリケーションをすべて配布した後の時点です |

| 項目 | 説明 |
|---------------------|---|
| | <ul style="list-style-type: none"> – READY: サーバーが起動し、ターゲットに登録されたアプリケーションがすべてデプロイされ、サービスの準備が完了した時点です – BEFORE_UNDEPLOY : サーバーがstopコマンドを受信したときに、サーバーでサービス中のアプリケーションをアンデプロイする前の時点です – AFTER_UNDEPLOY : サーバーがstopコマンドを受信したときに、サーバーでサービス中のアプリケーションのアンデプロイをすべて完了した時点です |
| Invocation Argument | メソッドを呼び出すときに使用する引数を指定します |
| Method Name | 呼び出しに使用されるメソッドを指定します |
| Method Param | 呼び出しに使用されるメソッドのパラメータを指定します |

4. ライフサイクル呼び出しリストに追加したライフサイクル呼び出しクラスが表示されます。ライフサイクル呼び出しクラスで使用するライブラリーを設定します。ライフサイクル呼び出しタイプに合わせて実行するメソッドを追加するためにLibrary Refリストの[Add]ボタンをクリックします。

[図 2.25] WebAdminによるライフサイクル呼び出しの設定(4)

Lifecycle Invocation HISTORY

サーバーの各種ライフサイクルイベントで呼び出すメソッドを設定できます。 ヘルプ

追加されました。

Basic **Resource** Engine

Listener | Jms Resource | Jmx Manager | Scheduler | **Lifecycle Invocation** | External Resource

動的設定 必須項目

Class Name * test.lifecycle.invocation.LifecycleInvocation com.tmax.event.ServerLifecycleListener
ライフサイクルイベントのコールバックメソッドが存在する完全修飾クラス名を指定します。

Library Ref

Library Name Add

該当する内容が存在しません。

Invocation

Method Name Add

| | |
|------|--------|
| boot | Delete |
|------|--------|

5. クラスで参照するライブラリーの設定です。設定したライブラリーはlib/sharedに格納される必要があります。共有ライブラリーについての詳細は、『JEUS アプリケーション&デプロイメントガイド』の「3.3.2. 共有ライブラリー」を参照してください。

[図 2.26] WebAdminによるライフサイクル呼び出しの設定(5)

Library Ref

HISTORY

アプリケーションで使用する共有ライブラリ情報を設定します。

ヘルプ

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 * 必須項目

確認 再設定

Library Name *
共有ライブラリの名前を指定します。

Failon Error ☐
[デフォルト: false] 該当する共有ライブラリが見つからなかった場合、デプロイを失敗させるかどうかを指定します。デフォルト値はfalseです。

Specification Version
使用する共有ライブラリの仕様バージョンを指定します。

Value *
バージョン値を設定します。

Exact Match ☐
[デフォルト: false] 完全にマッチするバージョンが必要かどうかを指定します。

Implementation Version
使用する共有ライブラリの実装バージョンを指定します。

Value *
バージョン値を設定します。

Exact Match ☐
[デフォルト: false] 完全にマッチするバージョンが必要かどうかを指定します。

確認 再設定

以下は、設定項目についての説明です。

| 項目 | 説明 |
|------------------------|--------------------------|
| Library Name | 使用するライブラリーの名前を指定します |
| Specification Version | 設定したライブラリーの仕様バージョンを指定します |
| Implementation Version | 設定したライブラリーの実装バージョンを指定します |

6. 以下は、ライブラリーを追加した後、**[確認]**ボタンをクリックして保存したときの結果画面です。

[図 2.27] WebAdminによるライフサイクル呼び出しの設定(6)

Lifecycle Invocation

HISTORY

サーバの各種ライフサイクルイベントで呼び出すメソッドを設定できます。

追加されました。

BasicResourceEngine

ListenerJms ResourceJmx ManagerSchedulerLifecycle InvocationExternal Resource

動的設定必須項目

Class Name

test.lifecycle.invocation.LifecycleInvocation

com.tmax.event.ServerLifecycleListener

ライフサイクルイベントのコールバックメソッドが存在する完全修飾クラス名を指定します。

Library Ref

Library Name

Add

lib1

Delete

Invocation

Method Name

Add

boot

Delete

7. メソッドが呼び出される時点に応じて、ライフサイクル呼び出しメソッドを追加で設定します。

[図 2.28] WebAdminによるライフサイクル呼び出しの設定(7)

Lifecycle Invocation

HISTORY

サーバの各種ライフサイクルイベントで呼び出すメソッドを設定できます。

追加されました。

BasicResourceEngine

ListenerJms ResourceJmx ManagerSchedulerLifecycle InvocationExternal Resource

動的設定必須項目

Class Name

test.lifecycle.invocation.LifecycleInvocation

com.tmax.event.ServerLifecycleListener

ライフサイクルイベントのコールバックメソッドが存在する完全修飾クラス名を指定します。

Library Ref

Library Name

Add

lib1

Delete

Invocation

Method Name

Add

boot

Delete

before_deploy

Delete

after_deploy

Delete

8. ライフサイクル呼び出し設定の変更を完了するには、[**Activate Changes**]ボタンをクリックして変更した設定をサーバーに反映します。画面上部にライフサイクル呼び出し設定の変更結果が出力され、その内容を確認することができます。

[図 2.29] WebAdminによるライフサイクル呼び出しの設定(8)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

1 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル

Domain

Server

もっと見る

Lifecycle Invocation

HISTORY

サーバーの各種ライフサイクルイベントで呼び出すメソッドを設定できます。

domain.xmlの設定を変更しました。

domain.xml : PENDING

servers.server.[? name == 'adminServer'].lifecycleInvocation : PENDING

変更内容を適用するには、サーバーを再起動してください。

ヘルプ

Basic Resource Engine

Listener Jms Resource Jmx Manager Scheduler Lifecycle Invocation External Resource

動的設定 必須項目

Class Name * test.lifecycle.invocation.LifecycleInvocation com.tmax.event.ServerLifecycleListener

ライフサイクルイベントのコールバックメソッドが存在する完全修飾クラス名を指定します。

Library Ref

Library Name

lib1

Invocation

Method Name

boot

before_deploy

after_deploy

参考

運用中のサーバーのライフサイクル呼び出し設定を変更した場合は、上記のような結果メッセージが出力されます。変更したライフサイクル呼び出し設定をサーバーに反映するためには、サーバーを再起動する必要があります。

2.3.5. リソース参照の設定

サーバーのアプリケーションで共通して使用するリソースのマッピング情報を設定します。

アプリケーションで使用するリソースは、アプリケーションがサービスされるサーバーのJNDIリポジトリに名前が登録されます。リソース・マッピングを設定すると、登録される名前に関係なく、アプリケーションで常に同じ名前のリソースをルックアップして使用することができます。

注

サーバーがクラスターに含まれている場合、リソース参照(Resource Reference)の設定はクラスターに設定された値が優先して適用されます。

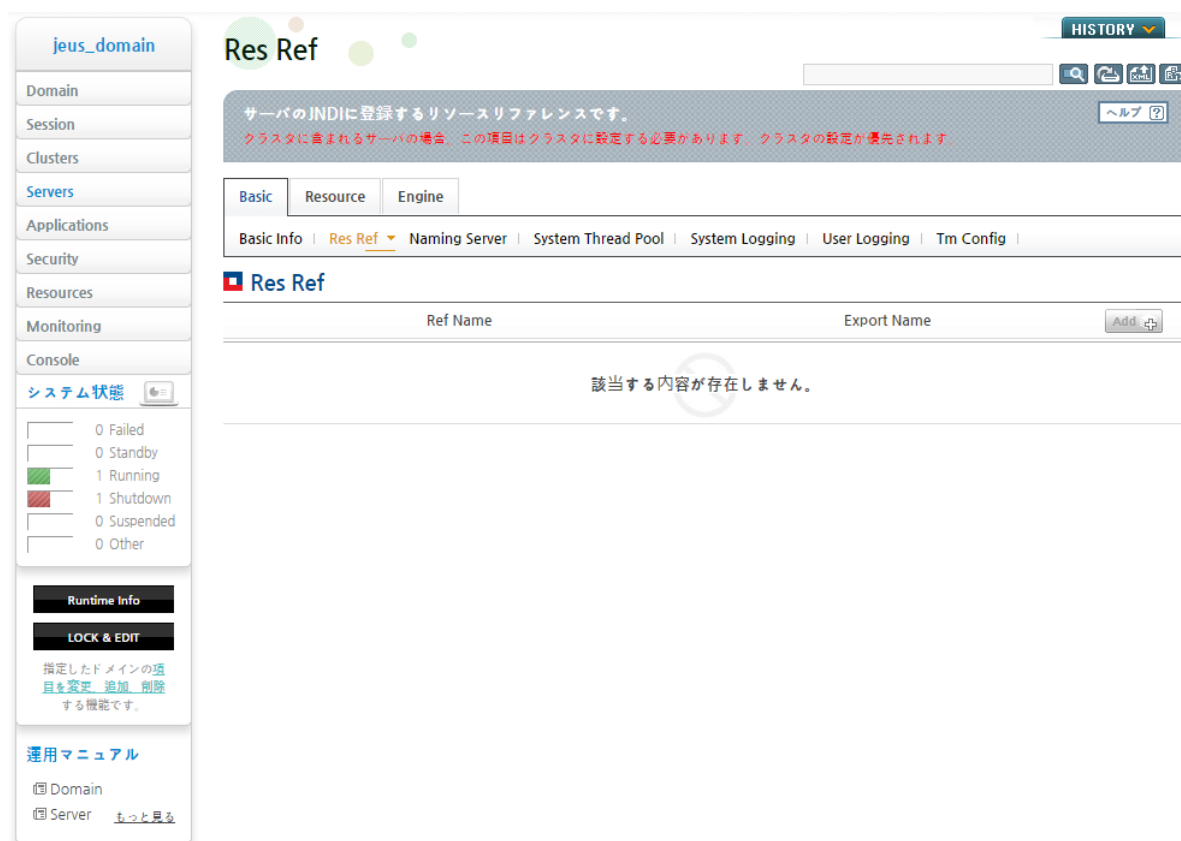
WebAdminの使用

以下では、WebAdminを使ってリソース参照を設定する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。

サーバー・リストでサーバーを選択すると、サーバー設定画面に移動します。設定画面で**[Basic]**タブを選択して**[Res Ref]**メニューを選択すると、**Res Ref**画面に移動します。

[図 2.30] WebAdminによるリソース参照の設定(1)



リソース参照の設定を変更するには、先に設定を変更するためのロックを取得しなければなりません。ロックを設定する前には、現在設定されているリソース参照の情報が確認できます。設定を変更するには、画面左のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックしてロックを取得します。ロックを取得すると、リソース参照の設定を変更できるモードに切り替わります。

2. リソース参照を追加するために、リソース参照リストの[Add]ボタンをクリックします。

[図 2.31] WebAdminによるリソース参照の設定(2)

The screenshot displays the WebAdmin interface for configuring resource references. The left sidebar shows the 'jeus_domain' menu with 'Servers' selected. The main area has tabs for 'Basic', 'Resource', and 'Engine', with 'Resource' active. Below the tabs is a breadcrumb trail: 'Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config'. The 'Res Ref' section has a table with columns 'Ref Name' and 'Export Name', and an 'Add' button. A message states '該当する内容が存在しません。' (No matching content exists). The bottom of the sidebar shows system status and runtime info.

jeus_domain

- Domain
- Session
- Clusters
- Servers**
- Applications
- Security
- Resources
- Monitoring
- Console

システム状態

- 0 Failed
- 0 Standby
- 1 Running
- 1 Shutdown
- 0 Suspended
- 0 Other

Runtime Info

Activate Changes

Undo All Changes

[変更された設定を保存、またはキャンセルする機能です。](#)

運用マニュアル

- Domain
- Server [もっと見る](#)

Res Ref

HISTORY

サーバのJNDIに登録するリソースリファレンスです。
クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。

ヘルプ

Basic | **Resource** | Engine

Basic Info | **Res Ref** | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

Res Ref

| Ref Name | Export Name |
|----------|-------------|
|----------|-------------|

Add

該当する内容が存在しません。

3. リソースをマッピングする「**Ref Name**」と実際にリソースがバインドされた「**Export Name**」を設定して[確認]ボタンをクリックします。

[図 2.32] WebAdminによるリソース参照の設定(3)

サーバのJNDIに登録するリソースリファレンスです。
クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。

Basic Resource Engine

Basic Info | **Res Ref** | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

Res Ref

| Ref Name | Export Name |
|----------------|-------------|
| 該当する内容が存在しません。 | |

JNDI Info

サーバのJNDIに登録する各リソース参照のエクスポート名と参照名を指定します。

動的設定 * 必須項目

| | | |
|-------------|-------------------------------------|-------------------|
| Ref Name * | jdbc/DB1 ソースコードで使用できる参照名を宣言します。 | EX ejb/AccountEJB |
| Export Name | db1 JEUS DDに定義された実際のJNDI名を設定します。 | EX ACCEJB |

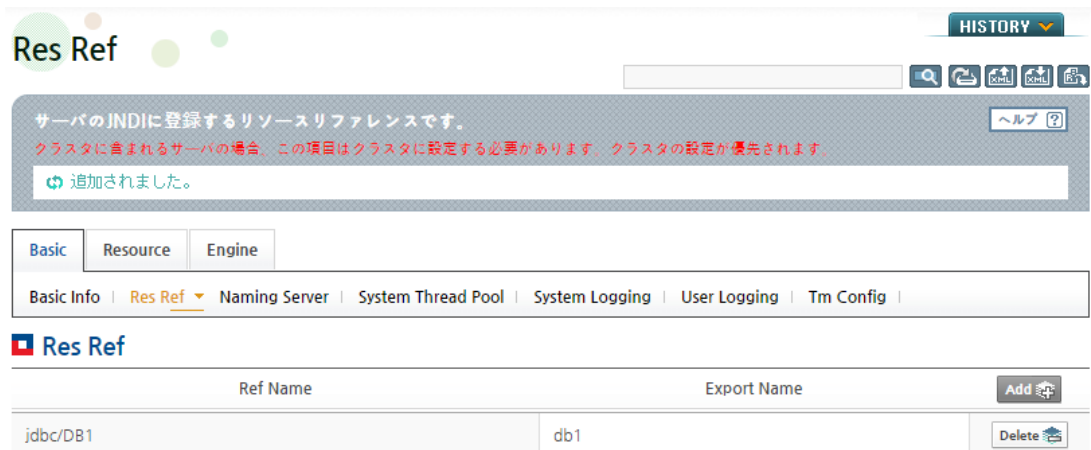
確認 再設定

以下は、設定項目についての説明です。

| 項目 | 説明 |
|-------------|---|
| Ref Name | マッピングするリソースの参照名を設定します。設定した名前はアプリケーションでルックアップの際に使用されます |
| Export Name | 実際にリソースがサーバーにバインドされている名前を指定します。サーバーのJNDIリポジトリに登録されているリソースのJNDI名です |

4. 以下は、リソース参照を追加した結果画面です。設定が完了すると、以下のように新規リソース・マッピング情報がリソースに追加されます。

【図 2.33】 WebAdminによるリソース参照の設定(4)

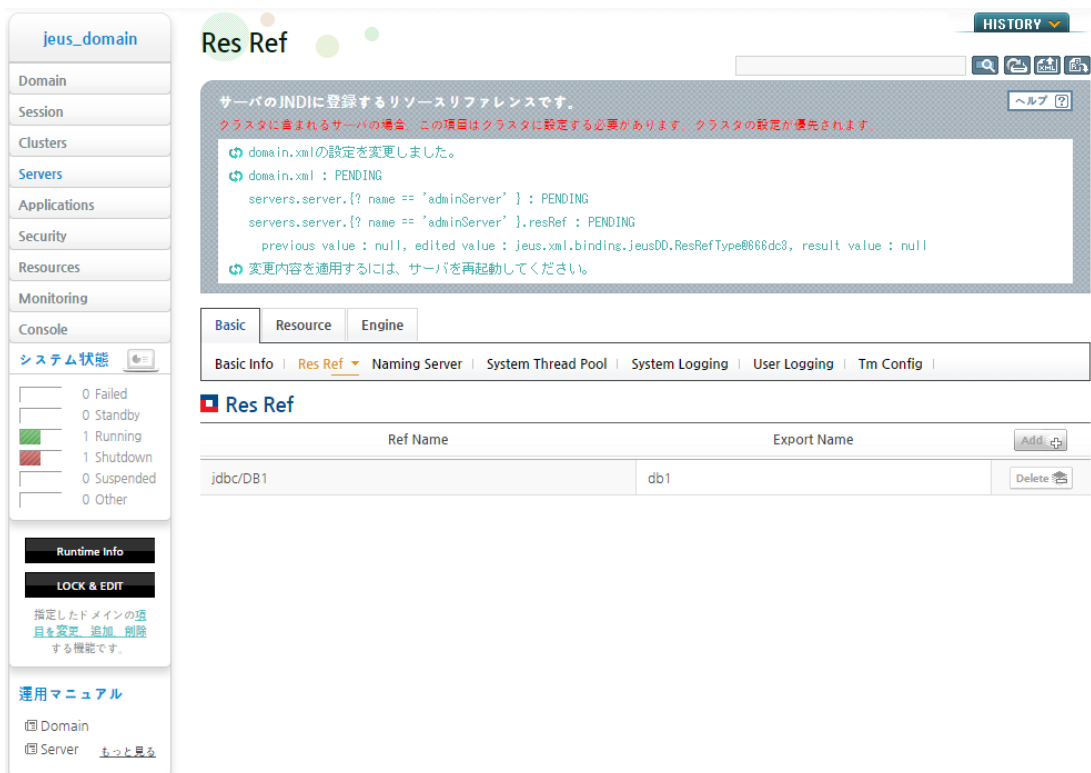


5. **[Activate Changes]**ボタンをクリックして設定の変更をサーバーに反映します。

6. 以下は、リソース・マッピング設定の変更結果です。

画面上部に結果メッセージが出力され、サーバーのリソース・マッピング設定が変更されたことが確認できます。運用中のサーバーのリソース・マッピング設定を変更した場合は、上記のような結果メッセージが出力されます。変更したリソース・マッピング設定をサーバーに反映するためには、サーバーを再起動する必要があります。

【図 2.34】 WebAdminによるリソース参照の設定(6)



第3章 JEUSサーバーの制御とモニタリング

本章では、JEUSサーバーを制御およびモニタリングする方法を説明します。ここで説明するサーバーはManaged Server(MS)に限定されます。また、管理を担当するDASが正常に動作していることを仮定します。サーバーの制御とモニタリングはWebAdminを使っても行えます。コマンド行に慣れていない方はWebAdminを使用することをお勧めします。

参考

サーバーの実行スクリプトとコンソール・ツールについての詳細は、『JEUS リファレンスガイド』の「第3章 JEUSサーバーの実行」と『JEUS リファレンスガイド』の「4.2.2. ローカル・コマンド」を参照してください。

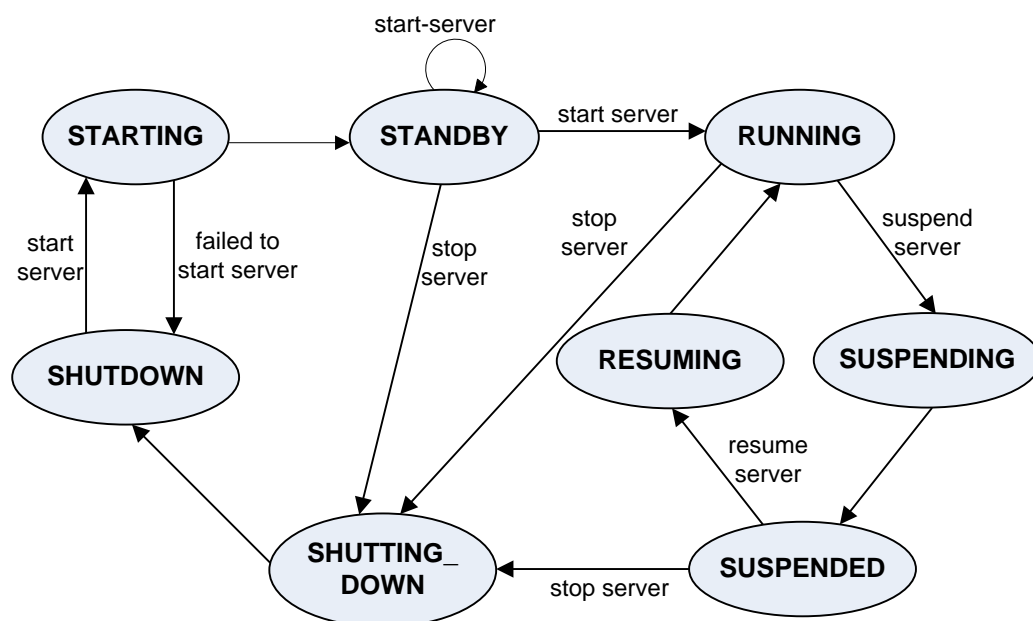
3.1. サーバーの制御とモニタリング

本節では、サーバーの制御とモニタリングについて説明します。

3.1.1. Managed Serverのライフサイクル

MSは、以下のような動作状態を持ちます。MSの状態はDASあるいはユーザーによって変更されます。

[図 3.1] MSの状態転移図



| 区分 | 説明 |
|---------------|---|
| SHUTDOWN | サーバーがまだ起動されていないか、正常に終了された状態です |
| STARTING | <p>サーバーが起動コマンドを受信して起動している状態です。</p> <p>サーバーは実行スクリプトまたはノード・マネージャーによって起動できます。この段階でサーバーはエンジン(WEB、EJB、JMS)の作成、JEUSサービス(セキュリティ、SCFなど)の実行、登録されたアプリケーションのデプロイなどを行います</p> |
| STADNBY | サーバーで実行されるJEUSサービスが開始された状態です。サーバーがこの状態で起動を完了した場合、サーバーに登録されたアプリケーションはデプロイに失敗したことになります。サーバーに登録されたアプリケーションの配布(distribute)に失敗した場合も、サーバーはSTANDBY状態となります |
| RUNNING | サーバーが起動を完了し、登録されたアプリケーションも正常にサービスされている状態です。STANDBY状態のサーバーを-forceオプションを使って起動した場合は、サーバーに登録されたすべてのアプリケーションがサービスされない可能性があります |
| SUSPENDING | <p>RUNNING状態のサーバーがsuspendコマンドを実行している状態です。</p> <p>この段階ではサーバーでサービスされているすべてのアプリケーションのサービスが中断されます。ただし、アプリケーションのサービスのためのリスナーは終了されません</p> |
| SUSPENDED | サーバーにデプロイされたアプリケーションをすべて停止させてサービスを中断した状態です。ただし、アプリケーションのサービスのためのリスナーは終了せずにそのままにします |
| RESUMING | <p>SUSPENDED状態のサーバーがresumeコマンドを実行している状態です。</p> <p>この状態では停止されたすべてのアプリケーションを再起動して、サービスが可能な状態にします。resumeコマンドが正しく実行されたら、サーバーはRUNNING状態になります</p> |
| SHUTTING_DOWN | <p>サーバーを終了している状態です。デプロイされたアプリケーションをアンデプロイし、起動時に実行させたすべてのJEUSサービスを終了させます。</p> <p>サーバーを終了時に適切なタイムアウトを設定することで、サービス中の要求の処理を保証できます。これをグレースフル・シャットダウン(Graceful Shutdown)といいます</p> |

参考

1. DASで管理するMSのライフサイクルの詳細については、『*JEUSドメインガイド*』の「4.4. サーバーのライフサイクル状態の確認」を参照してください。
 2. グレースフル・シャットダウン(Graceful Shutdown)についての詳細は、[「3.1.3. Managed Serverの終了」](#)を参照してください。
-

3.1.2. Managed Serverの起動

JEUS 7からDASとMSは基本的にランチャーにより起動されます。スクリプトまたはDASによるサーバー実行コマンドがランチャーを実行させ、ランチャーはサーバーを起動するための準備とサーバーの起動作業を行います。ランチャーについての詳細は[「1.6. ランチャー」](#)を参照してください。

コンソール・ツールまたはWebAdminを使ってDASによりサーバーを起動する方法については、『*JEUSドメインガイド*』の「4.2.2. Managed Server(MS)の起動」を参照してください。本節では、スクリプトによりサーバーを実行する例のみ説明します。

例

ランチャーはサーバーのJVMを起動し、サーバーの起動が完了したことを確認した後に終了されます。以下のように、ランチャー・ログによりサーバーが正常に起動されたことを確認できます。

```
JEUS_HOME/bin$ ./startManagedServer -domain domain1 -server server1 -u jeus -p jeus -dasurl
localhost:9736
*****
- JEUS Home           : /home/test/jeus
- JEUS Base Port      :
- Added Java Option   :
- Java Vendor         : Sun
*****
...
[2016.08.23 23:01:30][2] [launcher-1] [SERVER-0201] Successfully connected to the Domain
Administration Server(localhost:9736).
[2016.08.23 23:01:30][2] [launcher-1] [Launcher-0058] All local configurations are up-to-date.
...
[2016.08.23 23:01:36][2] [server1-1] [SERVER-0248] The JEUS server is RUNNING.
[2016.08.23 23:01:36][2] [server1-1] [SERVER-0401] The elapsed time to start: 4283ms.
[2016.08.23 23:01:36][2] [launcher-11] [Launcher-0034] The server[server1] initialization
completed successfully[pid : 20496].
[2016.08.23 23:01:36][0] [launcher-1] [Launcher-0040] Successfully started the server. The
server state is now RUNNING.
JEUS_HOME/bin$
```

MSが自身が持っているアプリケーション・ファイルを起動するとき、デプロイ時に問題が発生すると、STANDBY状態にとどまることがあります。MSが起動中にデプロイするアプリケーションは通常一度以上正常にサービスされたものです。そのため、このような問題が発生したら、ファイルが誤って変更された可能性が高いです。

特定のMSでJEUS内部ディレクトリーの.workspaceに存在するアプリケーションのxmlタグを誤って修正した場合、以下のようにSTANDBY状態にとどまることがあります。ドメイン構造の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』の「第1章ドメイン環境でのアプリケーション管理」を参照してください。

注

アプリケーションはDASで管理するのが基本ポリシーなので、MSの.workspaceディレクトリーに直接アクセスして作業しないようにします。

```
JEUS_HOME/bin$ ./startManagedServer -domain domain1 -server server1 -u administrator
-p adminadmin -dasurl 61.77.153.160:9736
...
[2012.05.12 17:12:48][2] [server1-1] [SERVER-0248] The JEUS server is STARTING.
[2012.05.12 17:12:48][2] [server1-1] [Deploy-0095] Distribute the application[de
ployment_helloear].
[2012.05.12 17:12:48][2] [server1-1] [SERVER-0201] Successfully connected to the
Domain Administration Server(61.77.153.160:9736).
[2012.05.12 17:12:48][0] [server1-1] [STDERR] java.lang.RuntimeException: Unexpe
cted close tag </application>; expected </module>
...
<<__!Exception__>>
[2012.05.12 17:12:48][1] [server1-1] [SERVER-0300] Distributing some registered
applications [deployment_helloear] failed.
[2012.05.12 17:12:48][2] [server1-1] [SERVER-0248] The JEUS server is STANDBY.
[2012.05.12 17:12:48][0] [server1-1] [SERVER-0250] Starting server (server1) fai
led. Staying in STANDBY.
[2012.05.12 17:12:48][2] [server1-1] [SERVER-0401] The elapsed time to start: 954lms.
[2012.05.12 17:12:48][2] [launcher-10] [Launcher-0034] The server[server1] initi
alization completed successfully[pid : 31068].
[2012.05.12 17:12:48][0] [launcher-1] [Launcher-0042] Comp
leted starting the server but the server state is still STANDBY.
JEUS_HOME/bin$
```

このような場合、以下のように**server-info**コマンドを使ってMSがSTANDBY状態であることが確認できま

```
[DAS]domain1.adminServer>server-info

Information of Domain (domain1)
=====
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

| Server | Status | Node | PID | Clu | Latest | Need | Listen | Running |
|--------|----------|------|------|-------|--------------|---------|------------|----------|
| | | Name | | ster | Start Time | to | Ports | Engines |
| | | | | | / Shutdown | Restart | | |
| | | | | | Time | | | |
| admin | RUNNING | nod | 5716 | N/A | 2016-08-24 | true | base-0.0. | jms, |
| Server | (04:24:1 | e1 | | | (水) 午後 | | 0.0:9736 | ejb, web |
| (*) | 0) | | | | 12:57:36 KST | | http-serv | |
| | | | | | | | er-0.0.0.0 | |
| | | | | | | | :8088 | |
| | | | | | | | jms-0.0.0 | |
| | | | | | | | .0:9741 | |
| serve | STANDBY | nod | 1820 | clu | 2016-08-24 | false | BASE-0.0. | jms, |
| r1 | (00:00:1 | e1 | | ster1 | (水) 午後 | | 0.0:9836 | ejb, web |
| | 8) | | | | 05:21:27 KST | | | |

3.1.3. Managed Serverの終了

MSは`jeusadmin`コマンドで終了できます。特定のMSに接続したコンソール・ツールで、`local-shutdown`コマンドを使って接続したMSを終了することもでき、コンソール・ツールやWebAdminからDASによるコマンドで終了することもできます。MSを終了する方法については、『*JEUS ドメインガイド*』の「4.3.1. Managed Server(MS)の停止」を参照してください。

参考

MSはDASによって管理するのが基本ポリシーなので、DASで終了することを推奨します。

例

MSを終了するとき、現在実行している要求の処理を保証するグレースフル・シャットダウン機能があります。サーバーの終了を要求した時点からは新しい要求を処理しませんが、現在実行している要求に対しては決められた時間まで処理が終わるのを待つことができます。

以下は、MS終了コマンドにタイムアウト・オプションを使って待機時間を設定する例です。

```
[DAS]domain1.adminServer>stop-server server1 -to 60000
Server [server1] was successfully stopped.
```

コンソール・ツールを使ってMSに直接接続した場合もタイムアウトを設定できます。

```
domain1.server1>local-shutdown -to 60000
The server [server1] has been shut down successfully.
offline>
```

正常に終了したサーバーでは以下のようなログが確認できます。

```
...
[2012.05.12 18:03:04][2] [server1-63] [SERVER-0248] The JEUS server is SHUTDOWN.
[2012.05.12 18:03:04][0] [server1-63] [SERVER-0265] The JEUS server has exited.
[2012.05.12 18:03:04][0] [server1-1] [SERVER-0099] The server[server1] has been
shut down.
[2012.05.12 18:03:04][0] [server1-9] [SERVER-0565] The JVM process is shutting down.
[2012.05.12 18:03:04][0] [server1-9] [SERVER-0566] The JVM process will be terminated.
```

3.1.4. Managed Serverの一時停止

MSの終了は、すべてのアプリケーション・サービスとMS JVMを終了するプロセスです。

一方で、サービス中のすべてのアプリケーションの一時停止機能も提供しています。この場合も同様に、グレースフル・タイムアウトを適用することで、現在実行中の要求の処理の完了を待つことができます。コンソール・ツールの**suspend-server**コマンドまたはWebAdminを使って操作できます。

WebAdminの使用

以下では、WebAdminを使ってサーバーを一時停止する手順について説明します。

1. WebAdminの左のメニューで**[Monitoring] > [Servers]**を選択すると、ドメインのサーバー・リストが照会されます。リストで一時停止するサーバーの名前の横にあるチェックボックスにチェックを入れた後、**[suspend]**ボタンをクリックします。

【図 3.2】ドメインのサーバー・リストの照会

The screenshot shows the WebAdmin interface for 'jeus_domain'. The left sidebar contains a navigation menu with 'Monitoring' selected. The main area is titled 'Servers' and displays a table of servers. The table has columns: Server, Status, Node Name, PID, and Latest Start Time / Shutdown Time. The table lists two servers: 'adminServer(*)' (RUNNING) and 'server1' (SHUTDOWN). Below the table are buttons for 'Start', 'Stop', 'Resume', 'Suspend', 'Dump', and 'Garbage Collection'.

| Server | Status | Node Name | PID | Latest Start Time / Shutdown Time |
|----------------|-------------------|-----------------|-----|-----------------------------------|
| adminServer(*) | RUNNING(00:14:35) | michael-desktop | 417 | 2016-10-06 (Thu) PM 09:53:33 KST |
| server1 | SHUTDOWN | michael-desktop | N/A | N/A |

2. 操作が完了すると、以下のように一時停止の実行結果が確認できます。

[図 3.3]ドメインのサーバーの一時停止

Servers

ドメインのサーバ情報を照会します。

Successfully suspended server(s).

| Server | Status | Node Name | PID | Latest Start Time / Shutdown Time |
|----------------|---------------------|-----------------|------|-----------------------------------|
| adminServer(*) | RUNNING(00:16:42) | michael-desktop | 417 | 2016-10-06 (Thu) PM 09:53:33 KST |
| server1 | SUSPENDED(00:00:42) | michael-desktop | 1026 | 2016-10-06 (Thu) PM 10:09:32 KST |

Start Stop Resume Suspend Dump Garbage Collection

コンソール・ツールの使用

以下は、コンソール・ツールでサーバーを一時停止する例です。

```
[DAS]domain1.adminServer>suspend-server -servers server1
Successfully suspended server(s).
```

server-infoコマンドを使って、サーバーがSUSPENDED状態になっていることが確認できます。

```
[DAS]domain1.adminServer>server-info

Information of Domain (domain1)
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Server | Status | Node | PID | Clu | Latest Start | Need | Listen | Runni |
|         |        | Name |     | ster | Time /      | to   | Ports  | ng    |
|         |        |      |     |      | Shutdown Time | Restart |        | Engin |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| admin | RUNNING | nod | 5716 | N/A | 2016-08-24 | true | base-0.0. | jms, |
| Server | (03:05:3 | e1  |     |     | (水) 午後   |      | 0.0:9736  | ejb, |
| (*)   | 6)      |     |     |     | 12:57:36 KST |      | http-serv | web  |
|         |         |     |     |     |              |      | er-0.0.0.0 |      |
|         |         |     |     |     |              |      | :8088     |      |
|         |         |     |     |     |              |      | jms-0.0.0 |      |
|         |         |     |     |     |              |      | .0:9741   |      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| serve | SUSPEND | nod | 139 | clu | 2016-08-24 | false | BASE-0.0. | jms, |
```

| | | | | | | | | |
|---|----------|----|----|--------------|--|----------|------|--|
| r1 | ED(00:00 | e1 | 84 | ster1 (水)午後 | | 0.0:9836 | ejb, | |
| | :58) | | | 04:02:14 KST | | | web | |
| +-----+-----+-----+-----+-----+-----+-----+-----+-----+ | | | | | | | | |
| ===== | | | | | | | | |

正常に一時停止したサーバーでは以下のようなログを確認できます。

```
[2012.05.12 18:24:51][2] [server1-30] [SERVER-0248] The JEUS server is SUSPENDING.
...
[2012.05.12 18:24:51][2] [server1-30] [EJB-4953] [deployment_helloear#ejb] The EJB
module stopped.
[2012.05.12 18:24:51][2] [server1-30] [WEB-3485] ServletContext[name=deployment_
helloear#web, path=/hello, ctime=Sat May 12 18:24:15 KST 2012] stopped successfully.
[2012.05.12 18:24:51][2] [server1-30] [SERVER-0248] The JEUS server is SUSPENDED.
```

参考

DASによって制御されていない場合も、コンソール・ツールでサーバーに直接接続してMSを一時停止することができます。

3.1.5. Managed Serverの再開

一時停止されたサーバーのアプリケーションを再びサービスするように再開することができます。

WebAdminの使用

以下では、WebAdminを使ってサーバーを再開させる手順について説明します。

1. WebAdminの左のメニューで**[Monitoring] > [Servers]**を選択すると、ドメインのサーバー・リストが照会されます。リストでサービスを再開するサーバーの名前の横にあるチェックボックスにチェックを入れた後、**[resume]**ボタンをクリックします。

[図 3.4] ドメインのサーバー・リスト

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

Servers

HISTORY

ドメインのサーバ情報を照会します。

Successfully suspended server(s).

Servers

No Group node Group Cluster

| Server | Status | Node Name | PID | Latest Start Time / Shutdown Time |
|-----------------|---------------------|-----------------|---|-----------------------------------|
| Running Engines | Cluster | Need to Restart | Listen Ports | |
| adminServer(*) | RUNNING(00:16:42) | michael-desktop | 417 | 2016-10-06 (Thu) PM 09:53:33 KST |
| | N/A | true | BASE-0.0.0.0:9736 http-server-0.0.0.0:8808 jms-jmstest-0.0.0.0:9741 | |
| server1 | SUSPENDED(00:00:42) | michael-desktop | 1026 | 2016-10-06 (Thu) PM 10:09:32 KST |
| | N/A | false | BASE-0.0.0.0:9836 | |

Start Stop Resume Suspend Dump Garbage Collection

2. 操作が完了すると、以下のようにサーバーが正常に再開されたことが確認できます。

[図 3.5] ドメインのサーバーの再開

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

Servers

HISTORY

ドメインのサーバ情報を照会します。

Successfully resumed the servers.

Servers

No Group node Group Cluster

| Server | Status | Node Name | PID | Latest Start Time / Shutdown Time |
|-----------------|-------------------|-----------------|---|-----------------------------------|
| Running Engines | Cluster | Need to Restart | Listen Ports | |
| adminServer(*) | RUNNING(00:19:11) | michael-desktop | 417 | 2016-10-06 (Thu) PM 09:53:33 KST |
| | N/A | true | BASE-0.0.0.0:9736 http-server-0.0.0.0:8808 jms-jmstest-0.0.0.0:9741 | |
| server1 | RUNNING(00:03:11) | michael-desktop | 1026 | 2016-10-06 (Thu) PM 10:09:32 KST |
| | N/A | false | BASE-0.0.0.0:9836 | |

Start Stop Resume Suspend Dump Garbage Collection

コンソール・ツールの使用

DASによって制御されていない場合も、コンソール・ツールでサーバーに直接に接続してSUSPENDED状態のMSを再開することができます。

以下は、コンソール・ツールを使ってサーバーを再開させる例です。

```
[DAS]domain1.adminServer>resume-server -servers server1
Successfully resumed the servers.
```

server-infoコマンドでサーバーが再びRUNNING状態になっていることが確認できます。

```
[DAS]domain1.adminServer>server-info

Information of Domain (domain1)
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Server | Status | Node | PID | Clu | Latest | Need | Listen | Running |
|         |         | Name |     | ster | Start Time | to | Ports | Engines |
|         |         |     |     |     | / Shutdown | Restart |      |         |
|         |         |     |     |     | Time      |      |      |         |
+-----+-----+-----+-----+-----+-----+-----+-----+
| admin | RUNNING | nod | 5716 | N/A | 2016-08-24 | true | base-0.0. | jms, |
| Server | (04:26:5 | e1 |     |     | (水) 午後 |      | 0.0:9736 | ejb, web |
| (*)   | 3)      |     |     |     | 12:57:36 KST |      | http-serv |      |
|         |         |     |     |     |              |      | er-0.0.0.0 |      |
|         |         |     |     |     |              |      | :8088      |      |
|         |         |     |     |     |              |      | jms-0.0.0 |      |
|         |         |     |     |     |              |      | .0:9741    |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| serve | RUNNING | nod | 4620 | clu | 2016-08-24 | false | BASE-0.0. | jms, |
| r1    | (00:01:0 | e1 |     | ster1 | (水) 午後 |      | 0.0:9836 | ejb, web |
|         | 0)      |     |     |     | 05:23:29 KST |      |          |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
=====
```

正常にアプリケーション・サービスが再開されたサーバーでは以下のようなログを確認できます。

```
[2012.05.12 18:25:07][2] [server1-30] [SERVER-0248] The JEUS server is RESUMING.
[2012.05.12 18:25:07][2] [server1-30] [Deploy-0098] Starting the application
[deployment_helloear].
[2012.05.12 18:25:07][2] [server1-30] [EJB-4952] [deployment_helloear#ejb]
The EJB module started.
[2012.05.12 18:25:07][2] [server1-30] [WEB-3484] ServletContext[name=deployment_
helloear#web, path=/hello, ctime=Sat May 12 18:24:15 KST 2012] started
successfully.
[2012.05.12 18:25:07][2] [server1-30] [Deploy-0099] Successfully started the
application[deployment_helloear].
[2012.05.12 18:25:07][2] [server1-30] [SERVER-0248] The JEUS server is RUNNING.
```


3.2. サーバー・エンジンの設定

JEUSサーバーでは、アプリケーションをサービスするエンジンが内部的に存在します。Webアプリケーション・サービス、EJBアプリケーション・サービス、JMSサービスのためのエンジンがそれぞれ存在します。各エンジンの使用可否は、初期化時点の設定を通じて制御することができます。ただし、現在これらの設定はサーバーを再起動しなければ適用されません。

3.2.1. エンジン使用可否の設定

JEUSサーバーの各エンジンの使用可否を設定します。各エンジンを使用する設定を行った場合、各エンジンはサーバーの起動時または各エンジンでサービスされる最初のアプリケーションがデプロイされる際に必要なエンジンが初期化されます。各エンジンを使用する設定を行っていない場合、サーバーが起動後にもエンジンは初期化されないため、該当エンジンでサービスされるアプリケーションをデプロイすると失敗します。

注意

エンジンが使用可能な際に正常にデプロイされたアプリケーションを、その後エンジンを使用していない状態でサーバーを起動すると、該当アプリケーションはデプロイに失敗します。そして、サーバーの状態もサービス可能状態(RUNNING)ではなく待機状態(STANDBY)に切り替わるため、管理者がこれを確認してエンジンおよびアプリケーションの状態を変更する必要があります。

WebAdminとコンソール・ツールも各エンジン別の使用可否を設定することができます。

WebAdminの使用

WebAdminの左のメニューの[Servers]でサーバーを選択すると、該当サーバーの設定を確認することができます。この画面の中間に各エンジンの使用可否を設定する部分があります。

【図 3.6】サーバー内部のエンジン設定画面

The screenshot displays the 'Server' configuration interface in WebAdmin. On the left is a sidebar menu with options like Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. The 'Servers' menu item is selected. The main area is titled 'Server' and contains a sub-header 'ドメイン内で使用するJEUSサーバの詳細設定を定義します。' (Define detailed settings for the JEUS server used within the domain). Below this is a tabbed interface with 'Basic', 'Resource', and 'Engine' tabs. The 'Basic' tab is active, showing a list of settings:

- Auto Generated:** A checkbox with a description: 'サーバが動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスタが削除されると、該当するサーバも自動的に削除されます。'
- Name:** A text field containing 'server1' with a note 'サーバ名です。'
- Log Home:** A text field with a description: 'JEUSサーバで生成するログのデフォルトパスを指定します。両パスが設定されていても、ローガのファイルハンドラに設定されているパスが絶対パスの場合にはそのパスが優先されます。デフォルトログのみならず、ローテーションされたバックアップログが生成されるパスにも同様に適用されます。'
- Node Name:** A dropdown menu showing 'michael-desktop' with a note: 'サーバが属するノード名を指定します。ノードにはマシン情報、ホスト情報などが含まれます。'
- Group:** A dropdown menu with a note: 'サーバを管理するためのグループを設定します。WebAdminでグループ別にサーバを管理することができます。'
- Action On Resource Leak:** A dropdown menu showing 'Warning' with a description: '【デフォルト: Warning】コンポーネント(主に、Stateless Component・Servlet/JSP、Stateless Session Bean、MDB)で利用したリソース(JCA、JDBCコネクションなど)をロギングするか、リターンするかを設定します。デフォルトはロギング(警告)です。データソース別に設定する場合、Action On Connection Leakを設定します。'
- Data Source Remote Lookup:** A checkbox with a description: '【デフォルト: false】リモートJVMでのデータソースルックアップを可能にします。データソースルックアップにより、リモートJVMでコネクションプールを構成して使用していた既存のスタンドアロンクライアントをサポートできます。'
- Engine Init On Startup:** A checkbox with a description: '【デフォルト: true】サーバに設定されているWeb、EJB、JMSエンジンなどの初期化時点を設定します。'
- Use Web Engine:** A checkbox with a description: '【デフォルト: true】サーバにWebエンジンを使用するか否かを設定します。'
- Use Ejb Engine:** A checkbox with a description: '【デフォルト: true】サーバにEJBエンジンを使用するか否かを設定します。'
- Use Jms Engine:** A checkbox with a description: '【デフォルト: true】サーバにJMSエンジンを使用するか否かを設定します。'

[LOCK & EDIT]ボタンをクリックし、エンジン情報の設定を変更して[確認]ボタンをクリックします。[Activate Changes]ボタンをクリックして設定情報を適用するには、サーバーを再起動する必要があります。

[図 3.7] サーバー内部のエンジン設定結果画面



コンソール・ツールの使用

コンソール・ツールでサーバー内の各エンジンの使用可否を設定する方法は以下のとおりです。

```
[DAS]domain1.adminServer>disable-engines adminServer -all
All engine(s) was(were) successfully disabled.
The configuration was changed.
=====
+-----+
|                                     Result                                     |
+-----+
```

```

| Successfully changed only the XML. |
| Restart the server to apply the changes. |
+-----+
=====
...
[DAS]domain1.adminServer>disable-engines adminServer -web -ejb
Web EJB engine(s) was(were) successfully disabled.
The configuration was changed.
=====
+-----+
|                                     Result                                     |
+-----+
| Successfully changed only the XML. |
| Restart the server to apply the changes. |
+-----+
=====
...

[DAS]domain1.adminServer>disable-engines adminServer -all
All engine(s) was(were) successfully enabled.
The configuration was changed.
=====
+-----+
|                                     Result                                     |
+-----+
| Successfully applied part of the changes. |
| Restart the server to apply the remaining changes. |
+-----+
=====
...
[DAS]domain1.adminServer>disable-engines adminServer -web -ejb
Web EJB engine(s) was(were) successfully enabled.
The configuration was changed.
=====
+-----+
|                                     Result                                     |
+-----+
| Successfully applied part of the changes. |
| Restart the server to apply the remaining changes. |
+-----+
=====

```

参考

JEUSコンソール・ツールのサーバー内のエンジン設定コマンドについての詳細は『*JEUS リファレンスガイド*』の「4.2.3.9. disable-engines」、『*JEUS リファレンスガイド*』の「4.2.3.15. enable-engines」を参照してください。

3.2.2. エンジンの初期化時点の設定

JEUSサーバーの各エンジンの使用可否を設定します。各エンジンを使用する設定を行った場合、サーバーの起動時にエンジンが初期化されるか、または各エンジンでサービスされる最初のアプリケーションがデプロイされる際にエンジンが初期化されます。各エンジンを使用する設定を行っていない場合、サーバーが起動後にもエンジンは初期化されないため、該当エンジンでサービスされるアプリケーションをデプロイすると失敗します。

WebAdminとコンソール・ツールも各エンジン別の使用可否を設定することができます。

WebAdminの使用

WebAdminの左のメニューの**[Servers]**でサーバーを選択すると、該当サーバーの設定を確認することができます。この画面の中間に各エンジンのサーバーを起動する際の初期化可否を設定する部分があります。

[LOCK & EDIT]ボタンをクリックし、エンジン情報の設定を変更して**[確認]**ボタンをクリックします。**[Activate Changes]**ボタンをクリックして設定情報を適用するには、サーバーを再起動する必要があります。エンジンの使用可否の設定確認画面と同じ画面であるため、上の図を参照してください。

コンソール・ツールの使用

コンソール・ツールでサーバー内の各エンジンの初期化時点を設定する方法は以下のとおりです。

```
[DAS]domain1.adminServer>disable-engine-init-on-boot adminServer,server1
EngineInitOnBoot was successfully enabled.
The configuration was changed.
=====
+-----+
|                                     Result                                     |
+-----+
| Successfully changed only the XML.                                         |
| Restart the server to apply the changes.                                   |
+-----+
=====

[DAS]domain1.adminServer>enable-engine-init-on-boot adminServer
EngineInitOnBoot was successfully enabled.
The configuration was changed.
```

```
=====
+-----+
|                                     Result                                     |
+-----+
| Successfully applied part of the changes.                                |
| Restart the server to apply the remaining changes.                        |
+-----+
=====
```

参考

JEUSコンソール・ツールのサーバー内のエンジン設定コマンドについての詳細は『*JEUS リファレンスガイド*』の「4.2.3.8. disable-engine-init-on-boot」、『*JEUS リファレンスガイド*』の「4.2.3.14. enable-engine-init-on-boot」を参照してください。

3.3. スレッドのモニタリングと制御

JEUSではサブレット・スレッドやEJBリモート要求スレッド、システム・スレッド・プールのモニタリング機能とスレッドのスタックを確認する機能、また特定のスレッドにインターラプト信号を送信する機能を提供します。

3.3.1. スレッドのモニタリング

JEUSではスレッドのモニタリング機能を提供します。モニタリングの対象スレッドは、サブレット・スレッド、EJBリモート要求スレッド、およびシステム・スレッド・プールです。

サブレット・スレッドとEJBリモート要求スレッドのモニタリング機能では、スレッドID、スレッド名、スレッド状態、処理時間などの情報を確認できます。スレッド・プールのモニタリング機能では、システム・スレッド・プールのコア・サイズ(core size)、最大サイズ(max size)、キープアライブ時間(keep alive time)、およびシステム・スレッド・プールに存在するスレッド情報を確認できます。

ただし、コンソール・ツールではサブレット・スレッドとEJBリモート要求スレッドのみモニタリングできます。

スレッド情報の確認

WebAdminとコンソール・ツールを使ってスレッド情報を確認できます。

• WebAdminの使用

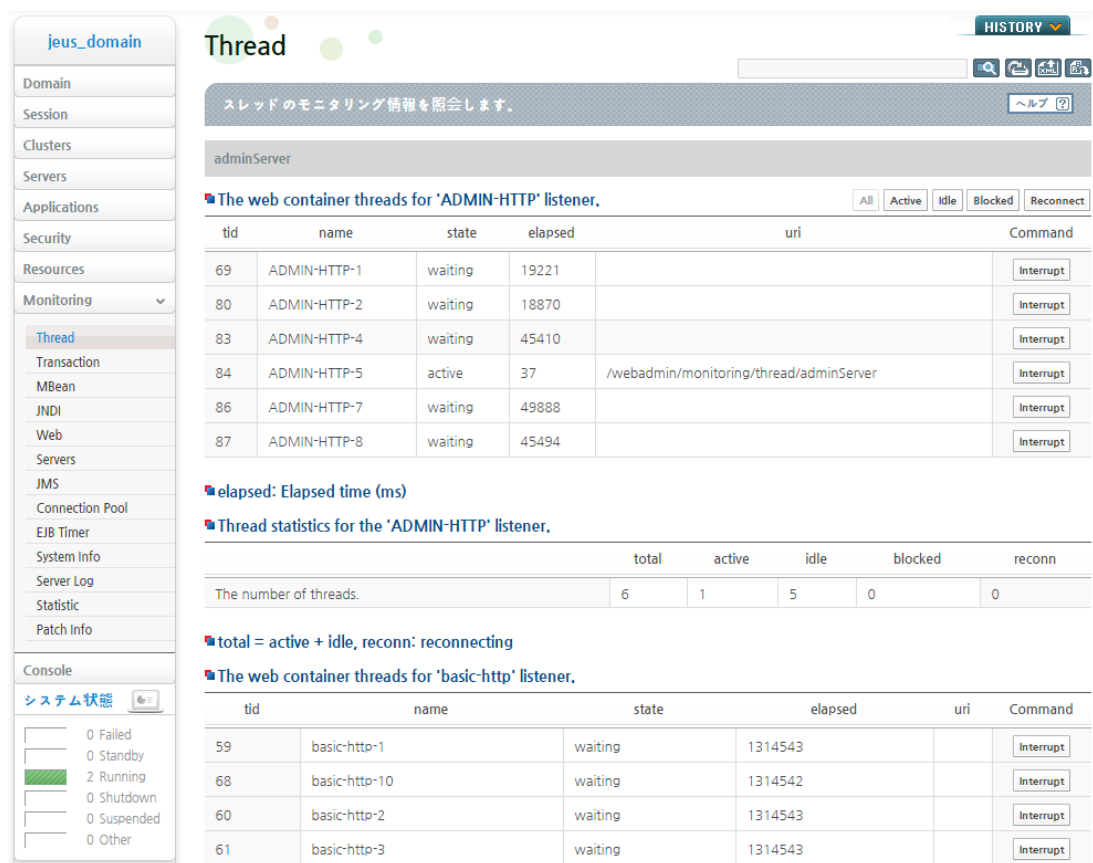
WebAdminの左のメニューで**[Monitoring] > [Threads]**を選択すると、スレッドのモニタリング画面が表示されます。

[図 3.8] スレッド・モニタリング画面



スレッド情報を照会するサーバーの名前を選択すると、以下のようにスレッド情報を確認することができます。

[図 3.9] WebAdminによるスレッド情報の確認



● コンソール・ツールの使用

コンソール・ツールでスレッド情報を確認する方法は以下のとおりです。

```
[DAS]domain1.adminServer>thread-info -server adminServer

Thread information for the server [adminServer]
There are no EJB RMI threads for the server [adminServer].
=====
The web container threads for 'ADMIN-HTTP' listener.

+-----+-----+-----+-----+-----+
| tid |          name          | state | elapsed | uri |
+-----+-----+-----+-----+-----+
|  48 | ADMIN-HTTP-2          | waiting | 21050 |    |
+-----+-----+-----+-----+-----+

elapsed: Elapsed time (ms)
=====

Thread statistics for the 'ADMIN-HTTP' listener.

+-----+-----+-----+-----+-----+
|          | total | active | idle | blocked | reconn |
+-----+-----+-----+-----+-----+
| The number of threads. |      1 |      0 |      1 |      0 |      0 |
+-----+-----+-----+-----+-----+

total = active + idle, reconn: reconnecting
=====

The web container threads for 'http1' listener.

+-----+-----+-----+-----+-----+[DAS]domain1.adminServer>disable-engines
adminServer -all
All engine(s) was(were) successfully disabled.
The configuration was changed.
=====

+-----+-----+-----+-----+-----+
|          | Result |
+-----+-----+-----+-----+-----+
| Successfully changed only the XML. |
| Restart the server to apply the changes. |
+-----+-----+-----+-----+-----+

[DAS]domain1.adminServer>thread-info -server adminServer

Thread information for the server [adminServer]
There are no EJB RMI threads for the server [adminServer].
```



```

=====
The web container threads for 'ADMIN-HTTP' listener.

+-----+-----+-----+-----+-----+
| tid |          name          | state | elapsed | uri |
+-----+-----+-----+-----+-----+
| 1053 | ADMIN-HTTP-14          | waiting | 123437 |    |
| 1077 | ADMIN-HTTP-16          | waiting | 107897 |    |
| 1078 | ADMIN-HTTP-17          | waiting | 123464 |    |
+-----+-----+-----+-----+-----+

elapsed: Elapsed time (ms)
=====

=====
Thread statistics for the 'ADMIN-HTTP' listener.

+-----+-----+-----+-----+-----+-----+
|                                     | total | active | idle | blocked | reconn |
+-----+-----+-----+-----+-----+-----+
| The number of threads.             |      3 |      0 |    3 |        0 |      0 |
+-----+-----+-----+-----+-----+-----+

total = active + idle, reconn: reconnecting
=====

=====
The web container threads for 'http1' listener.

+-----+-----+-----+-----+-----+
| tid |          name          | state | elapsed | uri |
+-----+-----+-----+-----+-----+
| 49  | http1-1                | waiting | 16083595 |    |
| 58  | http1-10               | waiting | 16083594 |    |
| 50  | http1-2                | waiting | 16083595 |    |
| 51  | http1-3                | waiting | 16083595 |    |
| 52  | http1-4                | waiting | 16083595 |    |
| 53  | http1-5                | waiting | 16083594 |    |
| 54  | http1-6                | waiting | 16083594 |    |
| 55  | http1-7                | waiting | 16083594 |    |
| 56  | http1-8                | waiting | 16083594 |    |
| 57  | http1-9                | waiting | 16083594 |    |
+-----+-----+-----+-----+-----+

elapsed: Elapsed time (ms)
=====

=====
Thread statistics for the 'http1' listener.

+-----+-----+-----+-----+-----+-----+
|                                     | total | active | idle | blocked | reconn |
+-----+-----+-----+-----+-----+-----+

```

```

| The number of threads.          |    10 |    0 |    10 |    0 |    0 |
+-----+-----+-----+-----+-----+-----+

total = active + idle, reconn: reconnecting
=====

=====
The threads for the 'threadpool.System' thread pool.

+-----+-----+-----+-----+-----+-----+
| tid |          name          | thread state | active thread |
+-----+-----+-----+-----+-----+
| 683 | threadpool.System-4 [adminServer-683] | RUNNABLE     | true          |
| 71  | threadpool.System-2 [adminServer-71]  | TIMED_WAITING | false         |
| 124 | threadpool.System-3 [adminServer-124] | TIMED_WAITING | false         |
| 68  | threadpool.System-1 [adminServer-68]  | TIMED_WAITING | false         |
+-----+-----+-----+-----+-----+
=====

=====
The statistics for the 'threadpool.System' thread pool.

+-----+-----+-----+-----+-----+-----+
| pool name | minimum | maximum | current | work | remaining work |
|           | pool size | pool size | pool size | queue size | queue size      |
+-----+-----+-----+-----+-----+-----+
| threadpoo | 0        | 100     | 4       | 4096 | 4096           |
|l.System   |          |         |         |      |                |
+-----+-----+-----+-----+-----+
=====

```

特定のスレッドのスタック・トレースの照会

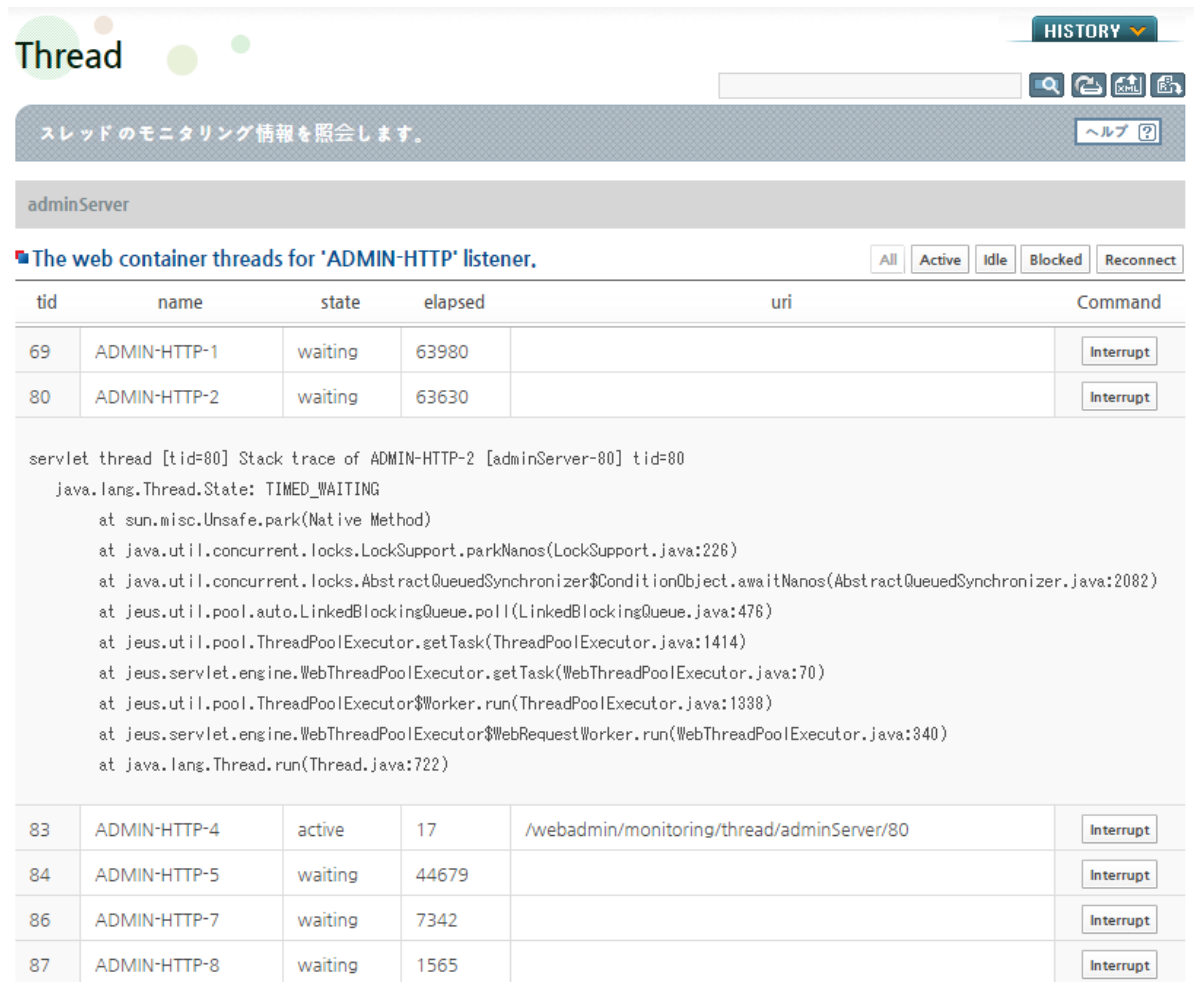
WebAdminとコンソール・ツールを使って特定のスレッドのスタック・トレースを照会できます。

● WebAdminの使用

WebAdminのスレッド・モニタリング画面で特定のtidをクリックすると、当該スレッドのスタック・トレースを照会できます。

以下は、WebAdminで特定のスレッドのスタック・トレースを照会する画面です。

[図 3.10] WebAdminによる特定のスレッドのスタック・トレースの照会



Thread

HISTORY

スレッドのモニタリング情報を照会します。

ヘルプ

adminServer

The web container threads for 'ADMIN-HTTP' listener.

All Active Idle Blocked Reconnect

| tid | name | state | elapsed | uri | Command |
|-----|--------------|---------|---------|-----|-----------|
| 69 | ADMIN-HTTP-1 | waiting | 63980 | | Interrupt |
| 80 | ADMIN-HTTP-2 | waiting | 63630 | | Interrupt |

servlet thread [tid=80] Stack trace of ADMIN-HTTP-2 [adminServer-80] tid=80

```

java.lang.Thread.State: TIMED_WAITING
    at sun.misc.Unsafe.park(Native Method)
    at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:226)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2082)
    at jeus.util.pool.auto.LinkedBlockingQueue.poll(LinkedBlockingQueue.java:476)
    at jeus.util.pool.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1414)
    at jeus.servlet.engine.WebThreadPoolExecutor.getTask(WebThreadPoolExecutor.java:70)
    at jeus.util.pool.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:1338)
    at jeus.servlet.engine.WebThreadPoolExecutor$WebRequestWorker.run(WebThreadPoolExecutor.java:340)
    at java.lang.Thread.run(Thread.java:722)
    
```

| | | | | | |
|----|--------------|---------|-------|--|-----------|
| 83 | ADMIN-HTTP-4 | active | 17 | /webadmin/monitoring/thread/adminServer/80 | Interrupt |
| 84 | ADMIN-HTTP-5 | waiting | 44679 | | Interrupt |
| 86 | ADMIN-HTTP-7 | waiting | 7342 | | Interrupt |
| 87 | ADMIN-HTTP-8 | waiting | 1565 | | Interrupt |

● コンソール・ツールの使用

コンソール・ツールで特定のスレッドのスタック・トレースを照会する方法は以下のとおりです。主に、スレッド情報を照会するコマンド(コンソール・ツールのtiコマンド)で確認したときにブロックされたスレッドがある場合に、そのスレッドのスタックを確認するために使用します。

```

[DAS]domain1.adminServer>print-stack-trace -server server1 45
servlet thread [tid=45] Stack trace of ADMIN-HTTP-1 [server1-45] tid=45
java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:156)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject
.await(AbstractQueuedSynchronizer.java:1987)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:399)
    at jeus.util.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1291)
    at jeus.servlet.engine.WebThreadPoolExecutor.getTask(WebThreadPoolExecutor.java:68)
    at jeus.util.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:1215)
    at jeus.servlet.engine.WebThreadPoolExecutor$WebRequestWorker.
run(WebThreadPoolExecutor.java:332)
    at java.lang.Thread.run(Thread.java:662)
    
```

参考

1. JEUSコンソール・ツールのtiコマンドとstraceコマンドについての詳細は『JEUS リファレンスガイド』の「4.2.2. ローカル・コマンド」を参照してください。
 2. サーブレット・スレッドとEJBリモート要求スレッドの情報とスタック・トレースは、WebAdminの[Monitoring] > [Threads]メニューで確認できます。WebAdminでは、システム・スレッド・プールの情報とシステム・スレッド・プールのスレッド情報も確認できます。より詳しい内容は『JEUS WebAdminガイド』とオンライン・ヘルプを参照してください。
-

3.3.2. スレッドの制御

JEUSでは特定のスレッドにインターラプト信号を送信して、実行中の作業を中断するように例外を発生させる機能があります。

注

この機能は現在試験的に提供されています。インターラプトされたスレッドの実行を直ちに中断するのではなく、インターラプト・ステータスをチェックして例外を発生させることでそれ以上実行を進めないように誘導しています。したがって、発生する例外に対してはユーザー・アプリケーションで処理しなければなりません。例外を正しく処理しないことで発生する問題については、ユーザーの責任となります。

スレッドのインターラプト

スレッドをインターラプトするということは、動作しているスレッドにインターラプト信号を送って現在実行中の動作の中断を誘導するために例外を発生させ、以降の作業を進めないようにヒントを与えることです。主に要求がブロックされているか、予想時間より長く遅延している場合に使用できます。

スレッドがインターラプトされるといって、実際にスレッドの動作が強制的に停止されるわけではありません。

JEUSでは、サーブレット・スレッドとEJBリモート要求スレッドにインターラプト信号を送信できる機能をサポートします。また、システム・スレッド・プールのスレッドがJNDIリモート・コネクションやEJBオペレーション、JDBCオペレーション、またはI/O操作を実行中の場合もインターラプト信号を送信できます。

● サーブレット・スレッド

サーブレット・スレッドは、次の場合にインターラプトされると、例外が発生します。発生した例外の処理はユーザー・アプリケーションで行う必要があります。

- JNDIリモート・コネクションを確立するとき
- JNDIリモート・オペレーションを実行するとき

– EJBを呼び出すとき

– JDBCコネクションを取得するとき、またはコネクション・オブジェクトのメソッドを呼び出すとき

コンソール・ツールまたはWebAdminを使って管理者が直接インターラプトをかけることができます。また、サーブレット・スレッドの実行時間をチェックして自動でインターラプトさせる機能を提供します。この機能の設定方法は、『*JEUS Webエンジンガイド*』の「2.3.7. 自動スレッド・プール管理の設定(スレッド状態の通知)」を参照してください。

● EJBリモート要求スレッド

EJBオペレーションでは、次の場合にインターラプト・ステータスをチェックして、`javax.ejb.EJBException`を発生させます。

– EJB 2.xで、`create`などのEJBHomeメソッドを呼び出すとき(EJB 3.0では、ユーザーがEJBルックアップをすると、EJBコンテナが内部で`create`処理を行う)

– EJBビジネス・メソッドを呼び出すとき

上記の場合に発生する例外については、ユーザー・アプリケーションで適切に処理する必要があります。もしEJBビジネス・メソッドがすでに呼び出された状態でインターラプト信号を受信すると、例外が発生しないことがあります。しかし、該当メソッドで他のEJBを呼び出したり、JDBCオペレーションやJNDIオペレーションを使用する場合は、インターラプトされることがあります。

● JDBC

JDBCオペレーションでは、次のオペレーションを実行するときにインターラプト信号を受信すると、`java.sql.SQLException`が発生します。発生した例外については、ユーザー・アプリケーションで適切に処理する必要があります。

– `DataSource#getConnection()`を使ってコネクション・プールからコネクションを取得するとき

– コネクション・プールから取得した`java.sql.Connection`に対してオペレーションを実行するとき

– JDBCコネクション・プールの設定に`use-sql-trace`または`stmt-caching-size`オプションを設定して、`executeQuery`などのオペレーションを実行するとき(オプションについての詳細は、「[6.4.2. コネクション・プールの設定](#)」を参照してください。)

● JNDI

JNDIオペレーションを実行中もインターラプトの効果がある場合があります。次のJNDIオペレーションを実行するときにインターラプト信号を受信すると、`javax.naming.NamingException`が発生します。発生した例外については、ユーザー・アプリケーションで適切に処理する必要があります。

– JNDIリモート・コネクションを確立するとき(`javax.naming.InitialContext`を作成する時点)

- JNDIオペレーションを行うためにリモート・メッセージを送るとき

Lookup、Bind、Rebind、RenameなどのJNDIオペレーションを行う場合です。ただし、ローカル(同じJVM)に存在するレジストリーにアクセスするか、キャッシュにアクセスする場合は、インターラプトの効果がありません。

● I/O操作

I/O操作を行うときは、状況によってインターラプトの効果がある場合があります。

- `java.nio.channels.SocketChannel`

`java.nio.channels.SocketChannel`を使用する場合は、インターラプトされます。この場合の処理は3段階に分けられます。

最初のbegin段階では、スレッドにインターラプト・ステータスが設定されているかチェックして、設定されている場合はチャンネルを終了します。接続段階では、`connect()`を実行しません。end段階では、`java.nio.channels.ClosedByInterruptedException`が発生します。(以降、インターラプト・ステータスはクリアされます。)

- `java.nio.channels.SocketChannel#read()/java.nio.channels.SocketChannel#write()`

`java.nio.channels.SocketChannel#read()`または`java.nio.channels.SocketChannel#write()`する場合も3段階に分けられます。

最初のbegin段階では、インターラプト・ステータスが設定されているかチェックして、設定されている場合はチャンネルを終了します。IOを処理する段階では、インターラプト・ステータスが設定されている場合、読み書きを進めません。最後のend段階では、`java.nio.channels.ClosedByInterruptedException`が発生します。(以降、インターラプト・ステータスはクリアされます。)

`java.nio.channels.ClosedByInterruptedException`が発生した後、以前生成したチャンネルを使って読み書きを行おうとすると、I/O操作が実行されず、`java.nio.channels.ClosedChannelException`が発生します。

- `java.net.Socket`の使用

`java.net.Socket`を使用する場合は、オペレーティング・システムやJVMによって異なります。HP-UXとSolarisでは、JVMのコンパイル・オプションによってソケットI/O操作をインターラプトできます。

ソケットのストリームによって読み書きを行う場合、インターラプト・ステータスがチェックされるため、インターラプトされることができ、`java.net.SocketException`が発生します(以降、インターラプト・ステータスはクリアされます)。しかし、このとき、実際にソケットが切断されるのではなく、インターラプトされた時点の次のオペレーションだけが中断されます。(以降、読み書きを実行すると、正常に動作します。)

一方、Windows、IBM AIX、Linuxなどでは、ソケットI/O操作を行うスレッドをインターラプトしても効果がありません。

- **Object#wait() / Thread#sleep() / Thread#join()**

スレッドがObject#wait()、Thread#sleep()、またはThread#join()を実行する場合、当該スレッドはインターラプトされます。この場合、スレッドはjava.lang.InterruptedExceptionが発生します。(以降、インターラプト・ステータスはクリアされます。)

参考

詳細については該当クラスのJavadoc API文書を参照してください。

インターラプト信号の送信

JEUSでは、コンソール・ツールとWebAdminを使ってスレッドにインターラプト信号を送信できるコマンドを提供します。

- **WebAdminの使用**

スレッド・モニタリング画面で[**interrupt**]ボタンをクリックすると、該当スレッドにインターラプト信号を送信できます。

- **コンソール・ツールの使用**

以下は、コンソール・ツールを使ってスレッド・インターラプト信号を送信する例です。JEUSコンソール・ツールの**interrupt-thread**コマンドについての詳細は、『*JEUS リファレンスガイド*』の「4.2.5.1. interrupt-thread」を参照してください。

```
[DAS]domain1.adminServer>interrupt-thread -server server1 45  
Sent an interrupt hint signal to the thread [tid=45] on the server server1.
```

3.4. メモリーのモニタリングと制御

JEUSでは、メモリーをモニタリングして、メモリーが特定の使用量を超えた場合にサーバーを制御できる機能を提供します。

3.4.1. メモリーのモニタリング

WebAdminとコンソール・ツールを使ってメモリー情報を確認できます。

WebAdminの使用

以下では、WebAdminを使ってメモリー情報を確認する方法について説明します。

参考

WebAdminを使ってメモリー情報を確認するには、**[Monitoring]**の**[Web]**メニューを使用する方法と、**[System Information]**メニューを使用する方法があります。**[System Information]**メニューは、メモリー情報だけでなく、CPU、プロセス、JVM情報も一緒に確認することができます。**[Web]**メニューでは簡単なメモリー情報を確認できます。本節では、**[System Information]**メニューを使用する方法を説明します。

1. WebAdminの左のメニューで**[Monitoring]** > **[System Information]**を選択すると、メモリーを含むシステムの全般的なモニタリング画面が表示されます。

[図 3.11] System Information画面

The screenshot shows the 'System Info' page for 'adminServer'. The left sidebar has a menu with 'Monitoring' expanded, showing 'Thread', 'Transaction', 'MBean', 'JNDI', 'Web', 'Servers', 'JMS', 'Connection Pool', 'EJB Timer', 'System Info' (selected), 'Server Log', 'Statistic', and 'Patch Info'. The main content area has a 'System Info' header with a 'HISTORY' dropdown. Below the header is a message 'システム情報を照会します。' and a link 'System Information of Server [adminServer]'. The 'adminServer' section shows 'CPU Information [adminServer]' and 'Process Information [adminServer]'. The CPU table lists: Number of Processors (4), CPU User Percent (5.35 %), CPU System Percent (8.02 %), CPU Wait Percent (0.00 %), CPU Nice Percent (0.00 %), and CPU Idle Percent (86.63 %). The Process table lists: Process Id (417), Thread Count (90), and Process Priority (20).

| Key | Value |
|----------------------|---------|
| Number of Processors | 4 |
| CPU User Percent | 5.35 % |
| CPU System Percent | 8.02 % |
| CPU Wait Percent | 0.00 % |
| CPU Nice Percent | 0.00 % |
| CPU Idle Percent | 86.63 % |

| Key | Value |
|------------------|-------|
| Process Id | 417 |
| Thread Count | 90 |
| Process Priority | 20 |

2. 照会するサーバーの名前を選択すると、以下のようにメモリーを含むシステム情報を確認できます。

[図 3.12] WebAdminによるメモリー情報の確認

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

システム状態

0 Failed

0 Standby

2 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル

Domain

Server もっと見る

Process Information [adminServer]

| Key | Value |
|------------------|-------------------|
| Process Id | 417 |
| Thread Count | 90 |
| Process Priority | 20 |
| Nice | 0 |
| Memory Resident | 423,034,880 bytes |
| Memory Share | 22118400 |
| Page Faults | 202807 |
| CPU Percent | 0.0 % |
| User Name | michael |

JEUS Memory Information [adminServer]

| Key | Value |
|--------------------------------|-------------------|
| Max Heap Memory | 477,233,152 bytes |
| Total Heap Memory | 141,099,008 bytes |
| Current Used Heap Memory | 112,739,664 bytes |
| Current Used Heap Memory Ratio | 23.6 % |

JEUS JVM Information [adminServer]

| Key | Value |
|--------------|------------------|
| JEUS Version | JEUS 8.0 (Fix#0) |

コンソール・ツールの使用

コンソール・ツールで**memory-info**コマンドを使ってメモリー情報を確認できます。memory-infoコマンドの詳細については、『*JEUS リファレンスガイド*』の「4.2.3. サーバー管理関連コマンド」を参照してください。

```
[DAS]domain1.adminServer>memory-info -servers adminServer
Domain [domain1] Memory Information
Memory Information
=====
+-----+-----+-----+
| Server | Total Amount of Memory | The Current Amount of Memory |
+-----+-----+-----+
| adminServer | 177274880 | 101567168 |
+-----+-----+-----+
=====
```

3.4.2. メモリー使用量による制御

JEUSでは頻繁に使用される機能ではないため、現在システム・プロパティによってのみメモリー関連動作を制御できます。

参考

1. メモリー・オーバーフローが発生した場合、1回だけヒープ・ダンプを取ります。(SERVER_HOME/logsディレクトリーに生成)
 2. IBM JDKの場合は、Javaを実行した場所にヒープ・ダンプが生成されます。異なる場所にパスを設定するには、-Xdump:heap:fileオプションを使用します。IBMのヒープ・ダンプに関連するより多くのオプションについてはIBMのホームページを参照してください。
 3. ヒープ・ダンプを記録する前にスレッド・ダンプがログに記録されます。
-

関連プロパティは以下のとおりです。

| プロパティ | 説明 |
|---|--|
| jeus.server.memorymonitor.enabled | MemoryMonitorServiceを使用するかどうかを設定します。 この設定が有効になっていない場合は、他のプロパティの属性は意味がありません (デフォルト値: false) |
| jeus.server.memorymonitor.ratio | メモリー・オーバーフローの基準値を設定します。 仮に0.8を設定したとすると、MAXメモリーを基準に約80%のメモリーを使用すると、オーバーフローとみなします(デフォルト値: 0.8) |
| jeus.server.memorymonitor.interval | メモリー使用量を測定する周期を設定します (デフォルト値: 2000、単位: ms) |
| jeus.server.memorymonitor.duration | メモリー・オーバーフローと判断する時間を設定します。(デフォルト値: 60000、単位: ms) たとえば、jeus.server.memorymonitor.ratioが0.8で、jeus.server.memorymonitor.durationが1分の場合、MemoryMonitorServiceではメモリーがMAXメモリーの80%以上使用される時間が1分以上持続する場合にメモリー・オーバーフローとみなします |
| jeus.server.enable.restart.in.memory.shortage | メモリー・オーバーフローと判定された場合、自動でサーバーを再起動するかどうかを設定します(デフォルト値: true) |

第4章 JNDIネーミング・サーバー

本章では、JEUS JNDIの基本概念と用語、環境設定、アプリケーションの開発方法について説明します。

4.1. 概要

Java Naming and Directory Interface™ (JNDI)は、Javaアプリケーションがネットワーク上でオブジェクトを取得できるようにする標準APIです。アプリケーションはオブジェクトの論理名を利用することで、オブジェクトを取得して使用することができます。ユーザーの観点から見ると、以前のエンタープライズ環境に比べて容易にオブジェクトを取得できる環境を提供します。

JEUS JNDIはJNDI 1.2 APIと互換性があり、Sun Microsystems社が提案した標準JNDI APIをサポートします。また、エンタープライズ環境に適合したJNDI Service Provider Interface(以下、SPI)も提供します。すなわち、JNDI SPIを実装した製品は、JEUS JNDIツリーのオブジェクトを使用することができます。

JEUS JNDIサービスはJEUSのシステム全体で使用されるため、EJB、Servlet/JSP、JMS、JDBCなどを使用するたびに言及されます。

4.2. 基本概念と構造

JEUS JNDIは、オブジェクトをバインドしてルックアップする固有のアーキテクチャーを持ちます。まず、JEUS JNDIの構造と基本概念について説明します。

4.2.1. 基本概念

Managed Server(MS)を起動すると、JNDIサービスが自動的に開始されます。JEUS JNDIサービスは、JNDIネーミング・サーバーによって提供されますが、JNDIネーミング・サーバーが実行されるときにJEUS ネーミング・サービス・サーバー(以下、JNSServer)が内部的に実行されます。このJNSServerと通信するためのクライアントの役割を担うのがJEUSネーミング・サービス・クライアント(以下、JNSClient)です。

JNSClientはJNSServerに接続されていて、JNSClientで先にオブジェクトのバインドとルックアップが行われ、次にJNSServerサーバーで行われます。1つのJNSServerは1つ以上のJNSClientに接続されます。さらに、JNSServerは他のJNSServerとも接続されているため(特にクラスタリング環境)、ツリーのような構造になっています。このようなネーミング・リポジトリ構造を**JNDIツリー**といいます。

JNDIツリーはオブジェクトをバインドまたはルックアップするときに使用されます。すべてのオブジェクトはサーバーを通じてJNDIツリーにバインドまたはルックアップされます。アプリケーションがJNSClientにオブジェクトのバインドを要求すると、JNDIツリーに要求が転送され、オブジェクトがバインドされます。このようにバインドされたオブジェクトはJNSClientを通じてアプリケーションでルックアップできます。

JNDIツリーのオブジェクトは、**InitialContext**を利用してのみアクセスできます。そのため、アプリケーションでJNDIを使用するためには、必ずInitialContextオブジェクトを作成する必要があります。このオブジェクトはJNDIツリーにアクセスしてオブジェクトを処理できるようにします。オブジェクトをバインド、ルックアップする機能だけでなく、オブジェクトのリストを取得または削除する機能も提供します。

バインドされているオブジェクトはWebAdminとコンソール・ツールを使用して確認することができます。

4.2.2. バインドされたオブジェクトの確認

WebAdminとコンソール・ツールを使ってバインドされているオブジェクトを確認できます。

参考

本節では、WebAdminを使って照会する方法のみ説明します。コンソール・ツールを使って照会する方法については、『*JEUS リファレンスガイド*』の「4.2.3.19. jndi-info」を参照してください。

WebAdminの使用

以下は、WebAdminでJNDIリストを照会する画面です。

1. WebAdminの左のメニューで**[Monitoring] > [JNDIs]**を選択すると、JNDIリスト照会画面が表示されます。画面のサーバー項目で照会するサーバーを選択すると、選択したサーバーのJNDIリストが照会されます。

[図 4.1] JNDIリストの照会



The screenshot displays the JNDI monitoring interface. At the top, there's a 'JNDI' header with a 'HISTORY' dropdown. Below it, a message states 'JNDIのモニタリング情報を照会します。' (Retrieve JNDI monitoring information). A search bar contains 'The JNDI list on the adminServer'. The main section is titled 'List of the context /' and shows a table of JNDI objects for 'adminServer'.

| Name | Value | Local Binding |
|---------------------|--|---------------|
| ConnectionFactory | jeus.jms.client.facility.factory.JeusConnectionFactory | false |
| ExamplesQueue | jeus.jms.common.destination.JeusQueue | false |
| ExamplesTopic | jeus.jms.common.destination.JeusTopic | false |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
| XAConnectionFactory | jeus.jms.client.facility.factory.JeusXAConnectionFactory | false |

Below the table, there's a section for 'server1'.

2. JNDIリストで照会するサーバーを選択してコンテキストをクリックすると、内部にバインドされたオブジェクト・リストが表示されます。リストで特定のオブジェクトをクリックすると詳細情報を照会できます。

[図 4.2] JNDIの詳細照会

JNDI

HISTORY

JNDIのモニタリング情報を照会します。

ヘルプ ?

The JNDI list on the server1

adminServer

server1

List of the context /JEUSMQ_DLQ

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |

Reference class name : jeus.jms.common.destination.JeusQueue
Reference Factory class name : jeus.jms.common.destination.DestinationOF
[0] StringRefAddr : JEUSMQ_DLQ, RefAddr type : name
[1] SerializableRefAddr : false, RefAddr type : needSort
end of RefAddr

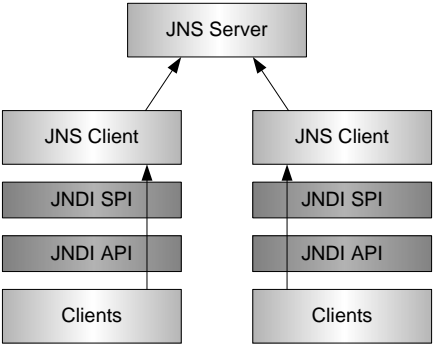
4.2.3. JNDIネーミング・サーバーのアーキテクチャー

本節では、JNDIツリー構造が実際にどのように動作するかについて説明します。

JNDIツリーは、JNSServerとJNSClientで構成されています。JNSServerはMSのJVMに存在し、JNSClientはMSまたはクライアントのJVMに存在します。JNSServerはJEUS JNDIアーキテクチャーのメイン要素としてJNDIツリーを作成および管理します。また、その下位にあるJNSClientを管理します。

下図は、JNSServerとJNSClientの関係を示しています。

[図 4.3] JNSServerとJNSClientの関係



JNDIツリー全体にアクセスするためには、JNSClientからJNSServerに要求を送信する必要があります。

JNSServerは他のMSのJNSServerに接続してクラスタリングを構成します。JNSClientはMS内でJNSServerと相互作用し、クライアントのアクセス要求を処理します。クライアントは直接JNSServerにアクセスするのではなく、JNSClientを利用してオブジェクトをバインドし、ルックアップします。

JNSServer

JNSServerはJNDIツリーを管理し、JNSClientがJNDIツリーにアクセスできるようにする独立したネーミング・サーバーです。複数のJNSServerに接続してJNDIツリーを拡張することもできます。JNSServerは他のMSのJNSServerに直接接続することができるためです。JEUSではMSが起動すると、JNSServerは自動的にJNSClientの接続を待ちます。

JNSClient

JNSClientの基本的な機能はJNSServerに接続してアプリケーションの要求を転送し、JNSServerの結果を再転送することです。それぞれのJVMでは、1つのサーバーに1つのJNSClientシングルトン・インスタンスが存在します。したがって、ルックアップを処理時に1つのJNSClientだけ使用するため、エンタープライズ環境においてEJBやサーブレットを使用する場合に有効です。

以下は、JNSClientの主な機能です。

- JNDIツリーへの接続

JNSClientはJNSServerに接続してMSが管理するJNDIツリーにアクセスする方法を提供します。バインドおよびルックアップされたオブジェクトは全体のJNDIツリーで共有されるか、クライアントの設定によって特定のクライアントのみアクセスすることもできます。

- ルックアップされたオブジェクトのキャッシュ

JNSClientは頻繁に使われるオブジェクトをキャッシュして、クライアントがより速やかに使用できるようにします。JNSClientはJNSServerと通信してオブジェクトをキャッシュします。

- JNSServerとの接続管理

JNSClientはクライアントからの要求を受信してJNSServerに転送し、その結果を受信して返します。クライアントが存在するJVMにJNSClientが存在するため、特に入出力(I/O)を行わずに効率的に通信することができます。

JNSClientは、その場所(JNSServerと同じMSにあるか、リモートにあるか)によって2種類に分けられます。

| JNSClient | 説明 |
|-----------|--|
| サーバー側 | MSの各エンジンにあるEJB Bean、サーブレット、JSPなどが使用します。 サーバー側のJNSClientを使用するには、java.naming.factory.initialプロパティ値をjeus.jndi.JNSContextFactoryに設定します。この値は基本的にMSの起動時に設定されるため特に考慮する必要はありません |

| JNSClient | 説明 |
|-----------|--|
| クライアント側 | <p>MSで動作しないアプレットやアプリケーション・クライアントなどが使用します。</p> <p>クライアント側のJNSClientを使用するには、<code>java.naming.factory.initial</code>プロパティ値を<code>jeus.jndi.JNSContextFactory</code>に設定します。</p> <p>クライアント側のJNSClientは一定時間通信がないと一旦コネクションを切って、必要な時に再びコネクションを作成し、リソースを効率的に管理します。クライアント・コンテナで動作するアプリケーションの場合は、クライアント・コンテナが実行されているときはこの値をシステム・プロパティとして設定するので、アプリケーションで別途設定する必要はありません</p> |

4.2.4. JNDIクラスタリング

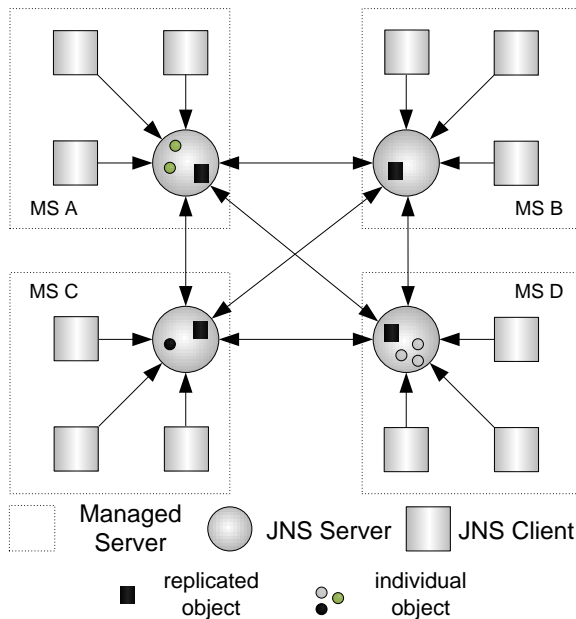
JNDIツリーは、JNSServerとJNSClient間の接続によって構成されています。この構造は、1つのコンポーネント(JNSClient)でオブジェクトの情報に変化がある場合、他のコンポーネント(JNSClient)にその内容を伝えます。また、複数のMSをサポート(クラスタリング)できます。

つまり、JNDIネーミング・クラスタリングは、独立したJNDIを1つの拡張されたJNDIツリーで構成することであるといえます。この場合も、それぞれのJNDIツリーでオブジェクトの情報に変化があったら、他のJNDIにその内容を伝えます。したがって、JNDIネーミング・クラスタリング環境では、1つのネーミング・サーバーでバインドしたオブジェクトを複数のネーミング・サーバーでルックアップすることができます。

たとえば、下図のようにA、B、C、Dという4つのMSがクラスタリングを構成する環境において、MS Aにobj1というオブジェクトを「objName1」という名前でバインドすると、他の3つのMS B、C、Dにもこのオブジェクトが転送されます。したがって、最初にクライアントが直接バインドしたMS A以外の3つのMSでも「objName1」でルックアップすると、オブジェクトobj1が取得できます。

しかし、バインドしたすべてのオブジェクトがクラスタリングに属するMSに複製されるのではなく、クラスタリング対象としてバインドしたオブジェクトのみ複製されます。たとえば、データソースのように各MSに特化されたオブジェクトは各MSのJNSServerにのみバインドされます。

【図 4.4】 クラスタリング環境のJEUS JNDIアーキテクチャー



各MSはそれぞれのJNSServerを管理する責任があります。各JNSServerはJEUSシステムの起動時に開始され、他のMSのJNSServerと接続します。各JEUSエンジンが初期コンテキストを取得すると同時に、JNSClientがJNSServerに接続されます。クライアントがJNDIツリーのオブジェクトをルックアップするときはJNSClientに要求し、該当MSのJNSServerに要求が転送されます。

クラスタリング環境のリモート・ルックアップ

別途の設定がない場合、JNDIルックアップは自身が含まれたJEUS JNDIクラスタリングの領域内で行われます。しかし、アプリケーションがクラスターにない他のMSのJNDIサーバーにある内容をルックアップするときは、プロバイダーURL(『*JEUS リファレンスガイド*』の「1.4. JNDIのシステム・プロパティ」を参照)を指定してコンテキストを作成するか、ルックアップ時にJEUSで作成したjh(JEUS Host)プロトコルを使用した名前でルックアップする必要があります。関連内容については、「[4.5. JNDIのプログラミング](#)」を参照してください。

JNSServerの複製

JEUSの各JNSServerは他のサーバーと接続され相互作用することを待ちます。既存のクラスターに新しいMSが参加すると、新規MSのJNSServerは起動時にすでに存在する他のJNSServerに通知を出します。そして、各JNSServerは自身のデータを新規JNSServerに転送します。こうして新たに参加したJNSServerでも以前バインドされたオブジェクトをルックアップできるようになります。

このような拡張性のため、異常終了して再起動されたJNSServerは他のJNSServerからJNDIツリー情報を取得し、正常に動作することができます。

4.3. JNDIネーミング・サーバーの設定

JNDIネーミング・サーバーはJNSServerとJNSClientで構成され、それぞれは異なる設定を有します。

JNSServerでは、JNSClientのコネクションを受けるための設定とJNSServerと接続するための設定が必要です。JNSClientの場合は、JNSServerとの接続やJNDIツリーの反映のための設定が必要です。

本節では、JNSServerとJNSClientの設定について説明します。

4.3.1. JNSServerの設定

WebAdminの左のメニューで[Servers]を選択すると、サーバー・リスト照会画面が表示されます。サーバー・リストでサーバーを選択すると、サーバー設定画面に移動します。設定画面で[Naming Server]メニューを選択すると、JNSServerを設定できる画面に移動します。

以下は、Naming Server設定画面です。

【図 4.5】 ネーミング・サーバーの設定

The screenshot displays the 'Naming Server' configuration interface. On the left is a sidebar with a menu for 'domain1' containing options like Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. Below the menu is a 'システム状態' (System Status) section showing various status indicators and a '運用マニュアル' (Operation Manual) link. The main area is titled 'Naming Server' and includes a 'HISTORY' button. Below the title is a description: 'JNDIサービスを行うためのJEUSネーミングサーバの情報を定義します。' (Define information for the JEUS Naming Server for JNDI service). The configuration is organized into tabs: Basic, Resource, and Engine. The 'Basic' tab is active, showing sub-tabs: Basic Info, Res Ref, Naming Server (selected), System Thread Pool, System Logging, User Logging, and Tm Config. A note indicates that settings can be changed by clicking 'Lock & Edit' in the sidebar. The configuration is divided into two main sections: 'Pooling' and 'Stuck Thread Handling'. The 'Pooling' section includes a 'Shared' radio button (selected) and a 'Dedicated' radio button. The 'Reserved Thread Num' is set to 0. The 'Min' and 'Max' thread counts are set to 0 and 10 respectively. The 'Keep Alive Time' is set to 60000 ms. The 'Queue Size' is set to 4096. The 'Stuck Thread Handling' section includes a 'Max Stuck Thread Time' set to 3600000 ms, an 'Action On Stuck Thread' set to None, and a 'Stuck Thread Check Period' set to 300000 ms.

共用スレッド・プールの設定

WebAdminまたはコンソール・ツールを使ってサーバー全体で共有するスレッド・プールを設定できます。スレッド・プールについての基本的な説明は、「[2.3.3. スレッド・プールの設定](#)」を参照してください。

• WebAdminの使用

以下は、WebAdminの共用スレッド・プールの設定画面です。「**Pooling**」チェックボックスにチェックを入れ、「**Shared**」項目を選択します。共用スレッド・プールの項目を設定して**[確認]**ボタンをクリックします。

[図 4.6] 共用スレッド・プールの設定

☒ **Pooling**

JNDIサービスのため、JEUSネーミングサーバへの要求を処理するためのスレッドプールを設定します。

☒ **Shared**

サーバ全体で共有するスレッドプールを使用します。

| | |
|---------------------|---------------------------------|
| Reserved Thread Num | <input type="text" value="10"/> |
|---------------------|---------------------------------|

[デフォルト: 0] サーバ全体で共有するスレッドプールを使用する場合、別のサービスがスレッドをすべて占有すると、スレッドの割り当てを受けられない場合があります。したがって、このサービスのためのスレッドを事前に割り当てたい場合に設定します。設定値は、他のサービスとの合計がシステムスレッドプールの最大値を超えないようにします。

• コンソール・ツールの使用

コンソール・ツールを使っても共用スレッド・プールを設定できます。

```
[DAS]domain1.adminServer>modify-system-thread-pool adminServer -service namingserver -r 10
Successfully performed the MODIFY operation for The namingserver thread pool of the server (adminServer)., but all changes were non-dynamic. They will be applied after restarting.
Check the results using "show-system-thread-pool adminServer -service namingserver" or modify-system-thread-pool adminServer -service namingserver"

[DAS]domain1.adminServer>modify-system-thread-pool adminServer -service namingserver
show the current configuration.
The namingserver thread pool of the server (adminServer)
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Reserved Threads for the Service namingserver                                | 10      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
=====
```

サービス専用スレッド・プールの設定

WebAdminまたはコンソール・ツールを使ってサービス別の専用スレッド・プールを設定できます。

● WebAdminの使用

以下は、WebAdminのサービス専用スレッド・プールの設定画面です。「**Pooling**」チェックボックスにチェックを入れ、「**Dedicated**」項目を選択します。サービス専用スレッド・プールの項目を設定して**[確認]**ボタンをクリックします。

[図 4.7] サービス専用スレッド・プールの設定

● **Dedicated**
サーバ別の専用スレッドプールを使用します。

| | |
|-----------------|---|
| Min | 5 [デフォルト: 0] スレッドプールで管理するスレッドの最小数です。 |
| Max | 10 [デフォルト: 10] スレッドプールで管理するスレッドの最大数です。 |
| Keep Alive Time | <input type="text"/> ms [デフォルト: 60000] 最小値(Min)以下のスレッドに対し、設定された時間内に使用されない場合は自動的にスレッドプールから削除されます。0の場合は削除しません。 |
| Queue Size | <input type="text"/> [デフォルト: 4096] スレッドプールで処理するタスクを蓄積するキューのサイズを指定します。 |

▼ **Stuck Thread Handling**
スレッドが特定のタスクによって一定時間以上継続して占有されている場合、当該スレッドに対して適切なアクションを取るための設定です。

| | |
|---------------------------|--|
| Max Stuck Thread Time | <input type="text"/> ms [デフォルト: 3600000] スレッドをスタックスレッドと判断する基準になる値を設定します。設定された時間以上継続して占有されている状態であれば、当該スレッドをスタックスレッドと見なします。 |
| Action On Stuck Thread | <input type="text"/> [デフォルト: None] スタックスレッドと判断された場合、そのスレッドに対して適切なアクションを取るための設定です。 |
| Stuck Thread Check Period | <input type="text"/> ms [デフォルト: 300000] スタックスレッドの状態をチェックする周期を設定します。 |

－ スタック・スレッド処理の設定

スレッドが特定の作業のために一定時間以上継続して占有された状態である場合、そのスレッドに対して特定のアクションを取るための設定です。共用スレッド・プールを使用します。

以下は、**Stuck Thread Handling**設定項目についての説明です。

| 項目 | 説明 |
|------------------------|---|
| Max Stuck Thread Time | スレッドをスタック・スレッドと判断する基準になる値です。 設定した時間以上スレッドが継続して占有された状態にある場合、そのスレッドをスタック・スレッドとみなします(デフォルト値: 3600000(1時間)、単位: ms) |
| Action On Stuck Thread | スタック・スレッドと判断した場合、そのスレッドに対して特定のアクションを取るための設定です。 以下の値のいずれかを設定します － None : 何のアクションも取りません |

| 項目 | 説明 |
|---------------------------|---|
| | <ul style="list-style-type: none"> Interrupt : インターラプト信号を送ります IgnoreAndReplace : スタック・スレッドを無視して新しいスレッドに切り替えます |
| Stuck Thread Check Period | <p>スタック・スレッドの状態をチェックする周期を設定します。</p> <p>設定した周期でスレッドの状態をチェックし、スタック・スレッドとして判断するかどうかを決定します(デフォルト値: 300000(5分)、単位: ms)</p> |

● コンソール・ツールの使用

コンソール・ツールを使ってもサービス専用スレッド・プールを設定できます。

```
[DAS]domain1.adminServer>modify-service-thread-pool server1 -service namingserver -min 0
-max 20
Successfully performed the MODIFY operation for The namingserver thread pool of
the server (server1)., but all changes were non-dynamic. They will be applied af
ter restarting.
Check the results using "show-service-thread-pool server1 -service namingserver
or modify-service-thread-pool server1 -service namingserver"

[DAS]domain1.adminServer>modify-service-thread-pool server1 -service namingserver
Shows the current configuration.
The namingserver thread pool of the server (server1).
=====
+-----+-----+
| Min                | 0      |
| Max                | 20     |
| Keep-Alive Time    | 60000  |
| Queue Size         | 4096   |
| Max Stuck Thread Time | 3600000 |
| Action On Stuck Thread | NONE   |
| Stuck Thread Check Period | 300000 |
+-----+-----+
=====
```

参考

サービスのスレッド・プールの設定の変更は動的に反映できるため、サーバーを再起動する必要がありません。ただし、共用スレッド・プールから専用スレッド・プールを使用するように変更する場合は、サーバーを再起動する必要があります。

4.3.2. JNSClientの設定

JNSClientはその場所によって設定方法が異なります。

- サーバー側のJNSClientの設定

サーバー側のJNSClientはMSと一緒に実行され、JNSSeverの設定に従います。

- クライアント側のJNSClientの設定

クライアント側のJNSClientは、異なるJVMのJNSServerにアクセスします。そのため、JNSServerに接続してJNDIツリーの内容を反映するスレッドが存在します。JEUS JNDIはスレッド・プールでスレッドを管理します。スレッド・プールはJEUSプロパティを使用して設定し、デフォルト値を使用しても構いません。

クライアント側のJNSClientのプロパティを設定するには、JVMのシステム・プロパティを使用するか、初期コンテキストのハッシュ・テーブルを使用します。

以下は、クライアント側のJNSClientのプロパティの種類です。

- JNSContext.RESOLUTION
- JNSContext.CONNECT_TIMEOUT
- JNSContext.CONNECTION_DURATION

参考

EJBやサーブレット/JSPのようなサーバー側のオブジェクトでJNDIを使用する場合は、MSによってサーバー側のJNSClientを使用してバインドまたはルックアップされるため、上記のようなプロパティの設定は不要です。プロパティの詳細については、『*JEUS リファレンスガイド*』の「1.4. JNDIのシステム・プロパティ」を参照してください。

4.4. クラスター環境でのJNDI

MS間でクラスタリング環境が構成されている場合は、JNDIをクラスタリングするための別途の作業は不要です。MSがクラスタリングされると、各MSのJNSServerも自動でクラスタリング環境を構成します。MSのクラスタリングについては、『*JEUS ドメインガイド*』の「第5章 JEUSクラスタリング」を参照してください。

JNSClientは基本的にローカルのJNSServerに接続します。JNSClientはMSのJNSServerアドレスをContext.PROVIDER_URLで提供されます。JNSClientは提供されたContext.PROVIDER_URLを基にロード・バランシングとフェイルオーバーを実行するため、クラスタリング環境ではクラスタリングに参加しているすべてのMSのアドレスを以下のように記述する必要があります。

```
Hashtable ht = new Hashtable();  
ht.put(Context.PROVIDER_URL, "host1:9736,host2:9736"); //クラスターにhost1、host2が  
参加
```

もし、Context.PROVIDER_URLに記述されたMSのうち特定のMSがFAILED状態になると、JNSClientはFAILED状態になったMSでJNSServerのJNDIオペレーションを行う際にそれを検知して他のJNSServerにフェイルオーバーします。JNSClientはFAILED状態と判断されたMSを周期的にチェックしてRUNNING状態になったか確認します。この設定は-Djeus.jndi.cluster.rechecktoオプションで設定できます。(デフォルト値: 5、単位: 分)

JNSClientがDASの管理を受けるMSで動作している場合は、以下のようにDASで設定したクラスター名を記述して使用できます。

```
Hashtable ht = new Hashtable();  
ht.put(Context.PROVIDER_URL, "jeus://cluster1"); //cluster1はDASに設定したクラスター  
名
```

クラスター名を記述して使用する場合は、JNSClientの「jeus.jndi.cluster.recheckto」オプションに関係なく、DASから常にMSサーバーの最新の状態情報を受信してクラスタリングできます。

つまり、クラスターに属するMSがFAILED状態になったり、新しいMSが追加された場合に、JNSClientはJNDIオペレーションを実行する前に最新のサーバー状態をアップデートして動作します。

そのため、JNSClientがDASの管理を受けるMSで動作する場合は、クラスター名を使用することを推奨します。

4.4.1. ルックアップ

クラスター内のクライアントはJNDIツリーにバインドされているすべてのオブジェクトをルックアップできます。クライアントがJNSClientにオブジェクトをバインドすると、直ちにJNSServerによりクラスタリングされたすべてのMSでそのオブジェクトを共有することになります。また、データの削除や変更が発生すると、直ちに各MSのJNSClientに反映されます。

JEUS JNDIではオブジェクトの特性によって、一部のオブジェクトはクラスター全体で共有され(ルックアップのためのエクスポート名などのオブジェクト)、また一部のオブジェクトは自身のMSでのみ有効です(データソースのようなMSに依存するオブジェクト)。

4.5. JNDIのプログラミング

本節では、JEUS JNDIを使用するプログラミング方法について説明します。

Javaクライアントは初期コンテキストを使用してJNDIツリーにアクセスします。初期コンテキストで使用されるプロパティには、JNDI標準プロパティとJEUS用のプロパティがあります。

まずJNDI環境を設定してから、初期コンテキストを使用してオブジェクトをルックアップした後、オブジェクトのリファレンスを取得します。最後に、使用が終わった初期コンテキストを閉じます。

Javaクライアントは以下の手順でJEUS JNDIサービスを使用します。

1. JEUSの環境設定を行います。
2. 初期コンテキストのプロパティを設定します。
3. コンテキストを使用した名前付きオブジェクトをルックアップします。
4. 名前付きオブジェクトを使用してオブジェクトのリファレンスを取得します。
5. コンテキストを閉じます。

各項目の詳しい説明については、該当する節を参照してください。

4.5.1. JEUSの環境設定

JNDIサービスで必要とするクラスを使用できるように、JEUSクライアント・モジュールのパス(JEUS_CLIENT)をクラス・パスに設定します。

```
-classpath ${JEUS_CLIENT};
```

4.5.2. 初期コンテキストのプロパティ設定

JavaクライアントがJEUS JNDIサービスを使用する前に、まず初期コンテキストの環境プロパティを設定します。

設定方法は次の2つがあります。

- JVMの-Dオプションを使用する方法
- ハッシュ・テーブルを作成して初期コンテキストのコンストラクター(constructor)に渡す方法

上記の2つの方法のうち、後者より前者が優先されます。

JEUS JNDIの初期コンテキストを作成するために設定が必要なプロパティは下記のとおりです。

- Context.INITIAL_CONTEXT_FACTORY (required)
- Context.URL_PKG_PREFIXES
- Context.PROVIDER_URL

- Context.SECURITY_PRINCIPAL
- Context.SECURITY_CREDENTIALS

参考

プロパティに関する内容は、『*JEUS リファレンスガイド*』の「1.4. JNDIのシステム・プロパティ」を参照してください。

これらのプロパティはハッシュ・テーブルに入れて初期コンテキストを作成するときに使用します。サーバー側のオブジェクト(EJBやサーブレット/JSP)でのみ初期コンテキストを使用する場合は、別途の設定を行わずにデフォルトの初期コンテキストを使用します。

クライアント・プログラムでは以下のように設定します。

```
Context ctx = null;
Hashtable ht = new Hashtable();
ht.put(Context.INITIAL_CONTEXT_FACTORY, "jeus.jndi.JNSContextFactory");
ht.put(Context.URL_PKG_PREFIXES, "jeus.jndi.jns.url");
ht.put(Context.PROVIDER_URL, "<hostname>");
ht.put(Context.SECURITY_PRINCIPAL, "<username>");
ht.put(Context.SECURITY_CREDENTIALS, "<password>");

try {
    ctx = new InitialContext(ht);
    // use the context
} catch (NamingException ne) {
    // fail to get an InitialContext
} finally {
    try{
        ctx.close();
    } catch (Exception e) {
        // an error occurred
    }
}
```

クラスタリング環境でJNDIを使用する場合は、JNDIツリーを構成し、JNSClientの内部動作を有効に管理するために、jeus.jndi.JNSContextの環境プロパティを追加使用することができます。

- ハッシュ・テーブルを使用したコンテキストの作成

初期コンテキストの環境プロパティを設定するとき、ハッシュ・テーブル(java.util.Hashtable)のキーにプロパティ名を入力し、値にはプロパティのデータを入力した後、初期コンテキスト・コンストラクターのパラメータとして設定します。

- セキュリティー・コンテキストの作成

JEUSセキュリティ・ドメインのユーザーを実行スレッドに適用するために、セキュリティ関連の環境プロパティを使用できます。初期コンテキストを作成するとき、下記のプロパティでセキュリティ・コンテキストを構成します。

- ユーザー名プロパティ : `java.naming.security.principal`
- パスワード・プロパティ : `java.naming.security.credentials`

認証に成功すると、認証されたユーザーの情報が実行スレッドに設定され、JEUSセキュリティ・マネージャーが管理するリソースにアクセスできます。認証に失敗した場合は、ゲスト(guest)ユーザーとして認識されます。

セキュリティ・コンテキストが設定されていても、初期コンテキストが作成される前にJEUSセキュリティAPIを使用してログインした場合には、セキュリティ・コンテキストは無視されます。すなわち、すでにログインされたサブジェクトを使用してJNDI通信を行います。

参考

1. ユーザー情報の設定に関する詳しい内容とJEUSセキュリティ・サービスに関する内容は、『JEUSセキュリティガイド』を参照してください。
 2. 初期コンテキストの設定に関連するプロパティと環境プロパティの詳細については、『JEUS リファレンスガイド』の「1.4. JNDIのシステム・プロパティ」を参照してください。
-

4.5.3. コンテキストを使用した名前付きオブジェクトのルックアップ

JNDIツリーにバインドされたオブジェクトはJNDIコンテキストの`Lookup()`メソッドを使って検索します。以下は、ルックアップするオブジェクトのモジュール名が`countermod`で、EJBの名前が`Count`である場合の例です。

```
try {
    Context ctx = new InitialContext();
    Count count = (Count)ctx.lookup("java:global/countermod/Count");
//    Count count =
    (Count)ctx.lookup("java:global/countermod/Count!my.ejb3.tx.Count");
//    Count count = (Count)ctx.lookup("Count");
    // successfully got the object
} catch (NameNotFoundException ex) {
    // no such binding exists
} catch (NamingException e) {
    // an error occurred
}
```

ビジネス・インターフェースが複数ある場合は、"!インターフェース名"を後ろにさらに付けてルックアップします。

```
Count count = (Count)ctx.lookup("java:global/countermod/Count!my.ejb3.tx.Count");
```

エクスポート名を与えた場合は、エクスポート名でルックアップできます。

```
Count count = (Count)ctx.lookup("Count");
```

以下は、EJB 2.xオブジェクトをルックアップする例です。

```
try {
    Context ctx = new InitialContext();
    CountHome countHome = (CountHome)ctx.lookup("Count");
    // successfully got the object
} catch (NameNotFoundException ex) {
    // no such binding exists
} catch (NamingException e) {
    // an error occurred
}
```

4.5.4. 名前付きオブジェクトの使用

EJB 3.xでは、ルックアップしたオブジェクトを以下のように使用します。

```
count.service();
```

EJB 2.xでは、ルックアップしたEJBホーム・オブジェクトのcreate()メソッドを使ってEJBリモート・オブジェクトのリファレンスを取得します。以下のようにメソッドを実行して、使用するオブジェクトのホーム・リファレンスを取得し、そのリファレンスのメソッドを実行します。

```
Count count = countHome.create();
count.service();
```

4.5.5. コンテキストのクローズ

コンテキストを使用した後は、以下のようにclose()メソッドを実行してコンテキストを閉じます。

```
try {
    cx.close();
} catch (Exception e) {
    // an error occurred
}
```

4.5.6. クラスターリング・コンテキストの作成

JEUSクラスターリング環境で使用可能なコンテキストを作成するには、以下のようにContext.PROVIDER_URLに複数のホストを設定します。

```
Hashtable ht = new Hashtable();
ht.put(Context.PROVIDER_URL, "host1:9736,host2:9736");
```

ドメインに設定したクラスター名を知っている場合は、以下のようにクラスター名を設定して使用することもできます。

```
Context.PROVIDER_URL 'jeus://' + <クラスター名>
```

ただし、クラスターに属するMSでのみ使用することができます。独立したクライアント(スタンドアロン・クライアント)では、クラスター情報を知らないため使用できません。

```
Hashtable ht = new Hashtable();
ht.put(Context.PROVIDER_URL, "jeus://cluster1");
```

クラスターリングされたコンテキストでルックアップする場合、どのMSからオブジェクトを取得するかに関するポリシーを決めることができます。

jeus.jndi.clusterlink.selection-policyプロパティを提供しており、次の3つの値を設定できます。

| 区分 | 説明 |
|---------------------|--|
| locallinkPreference | ローカルMSにあるオブジェクトを使用します |
| roundrobin | 最初の要求では任意に選択したMSのオブジェクトを使用し、その後の要求からは1ずつ増やしながらサーバーを選択します |
| random | クラスターリングされたMSのうち、任意に1つを選択して使用します |

以下は設定例です。

```
Hashtable ht = new Hashtable();
ht.put(Context.PROVIDER_URL, "host1:9736,host2:9736");
ht.put("jeus.jndi.clusterlink.selection-policy", "random");
```

4.5.7. リモート・ルックアップの実行

アプリケーションがクラスターに存在しない他のMSのJNDIサーバーの内容をルックアップする場合は、PROVIDER URLを指定してコンテキストを作成するか、ルックアップするときにJEUSで生成したjh(JEUS Host)プロトコルを使用した名前でルックアップする必要があります。

以下は、JEUSのクラスター環境でリモート・ルックアップを実行できる特別なLookup構文です。この構文は、自身が属するJEUS JNDIクラスタリング領域を超えて、リモートのJEUS JNDIクラスタリングにあるオブジェクトをルックアップするときに使用します。

```
jh:<remote JEUS host name>:<remote JEUS base port>/<export name>  
("jh" = JEUS host)
```

以下は、ルックアップ構文の使用方法です。JNDIコンテキストの文字列内に下記の構文を入力します。

```
try {  
    Context ctx = new InitialContext();  
    CountHome countHome = (CountHome)ctx.lookup("jh:dev:9736/Count");  
    // successfully got the object  
} catch (NameNotFoundException ex) {  
    // no such binding exists  
} catch (NamingException e) {  
    // an error occurred  
}
```

第5章 外部リソース

本章では、JEUSと連動して1つのシステムを構築できる多様な外部リソースを紹介し、その設定方法について説明します。外部リソースの設定と使用方法に関する詳しい内容については、各外部リソースのマニュアルを参照してください。

5.1. リソースの種類

外部リソースはアプリケーションがJEUSを介してアクセスできる、JEUSの外部に存在するリソースです。代表的な外部リソースとしてはデータベースがあります。JEUSに関連設定を追加することで、外部リソースを連携することができます。

参考

外部リソースがJCA標準互換のリソース・アダプターを提供している場合は、リソース・アダプターをデプロイして使用することをお勧めします。

以下は、JEUSに設定可能なリソースです。

● データソース

データソースはJDBC互換データベースです。

データソースはクライアントから直接アクセスすることができますが、その場合は特にJEUSでの設定は不要です。しかし、データソースを設定すると、JNDIを利用してJDBCコネクション・プールを使用できるため、アプリケーションがデータベースにより容易にアクセスすることができます。データソースの設定については「[第6章 DBコネクション・プールとJDBC](#)」を参照してください。

● メール・リソース

メール・リソースは、SMTPなどのメール・プロトコルを利用してクライアント・アプリケーションからメールを送信するときに使用します。JEUSでは、JNDIエクスポート名にメール・ホストの情報をバインドして、クライアントが間接的にアクセスしてホストを使用するようにします。

JNDIをルックアップすると、`javax.mail.Session`タイプのメールソースを取得します。

● URLリソース

URLリソースは、アプリケーションがJNDIを利用して外部URLオブジェクトにアクセスできるようにします。URLが変更される場合は、URLの設定を変更することでアプリケーションのソースを修正せずにそのまま使用できるようにします。

JNDIをルックアップすると、java.net.URLタイプのURLオブジェクトを取得します。

- **メッセージ・ブリッジ**

メッセージ・ブリッジは様々なJMSベンダーのデスティネーション間の接続のために使用されます。

JMS 1.1仕様を満たすMQであれば何でも設定可能です。詳細は『*JEUS MQガイド*』の「6.1. JEUS MQのメッセージ・ブリッジ」を参照してください。

- **カスタム・リソース**

カスタム・リソースはJava Bean形式のリソースをJNDIリポジトリにバインドできます。ルックアップ時にJNDI ObjectFactoryを通じて登録したサービスを使用できるようにする一般的なリソースです。

- **外部ソース - IBM MQ、Sonic MQなど**

JEUSに接続できる外部ソースです。通常、JEUSに設定可能な外部ソースには、IBM MQ、Sonic MQなどのJMS(Javaメッセージ・サービス)製品とTmax TPモニターなどがあります。

これらのソースは、JEUSに設定しなくても、Java APIを利用して直接アクセスすることができます。しかし、トランザクション・マネージャーでこれらのソースを管理するためには設定が必要です(「[第7章 トランザクション・マネージャー](#)」を参照)。

- **外部リソース**

JEUS上で動作するリソースです。主にJEUSと連動するWebTやjTmax、またはInfiniteCacheで使われます。JEUSでこのような外部リソースを使用するには、jeus.external.ResourceBootstrapperを実装したクラスを登録する必要があります。

- **並行処理ユーティリティ・リソース**

Concurrency Utilities for Java EEと関連するリソースを定義します。これにより、アプリケーション・サーバー上で管理可能なタスクを定義し、タスクが実行されるときにコンテキストを維持しつつ動作することができます。詳細については、『*JEUS 並行処理ユーティリティガイド*』を参照してください。

5.2. リソースの設定

本節では、各リソースをWebAdminを使って設定する方法について説明します。

リソースはドメイン範囲で設定します。サーバーが起動時にこの設定情報を読み込んでリソースを登録します。

WebAdminの左のメニューで[Resources]を選択すると、それぞれのリソースを照会し、サーバーに登録、変更、削除することができます。

【図 5.1】 WebAdminによるリソースの設定

The screenshot shows the WebAdmin interface for 'jeus_domain'. The left sidebar has a menu with 'Resources' selected. The main area displays the following information:

- Servers**: Failed 0 | Standby 0 | Running 3 | Shutdown 0 | Suspended 0 | Other 0
- DataSource**: Database 2 | Cluster-ds 1
- Applications**: Application 4
- Clusters**: Clusters 0
- Resource Manager**: Message Bridge, Mail Source, External Source, DataSource

| Server | Status | Engine |
|----------------|-------------------|--------|
| adminServer(*) | RUNNING(01:56:29) | |
| server1 | RUNNING(00:32:59) | |
| server2 | RUNNING(02:03:23) | |

システム状態

- 0 Failed
- 0 Standby
- 3 Running
- 0 Shutdown
- 0 Suspended
- 0 Other

5.2.1. データソースの設定

データソースの設定はデータベースに関する設定です。この内容については、「[第6章 DBコネクション・プールとJDBC](#)」を参照してください。

5.2.2. メールソースの設定

クライアント・アプリケーションがJEUSにメールを送信するときに使用するSMTPホストを設定できます。WebAdminを使ってSMTPサーバーごとにメール・ホストの情報を設定します。メール・ホスト情報はJavaMail仕様を参照してください。

WebAdminの左のメニューから[Resources] > [Mail Source]を選択すると、ドメインに登録されたメールソースを照会できます。

[図 5.2] WebAdminによるメールソースの設定(1)

[Add]ボタンをクリックすると、ドメインにメールソースを登録できます。

[図 5.3] WebAdminによるメールソースの設定(2)

参考

「Main Property」に設定する属性の名前はJavaMailの仕様に従います。JavaMailの仕様を参照して指定してください。

5.2.3. URLソースの設定

以下は、JNDI名がPRIMARY_URL、SECONDARY_URLであり、それぞれにバインドされるURLが下記のようなURLソースの例です。

http://www.foo.com

http://www.bar.com

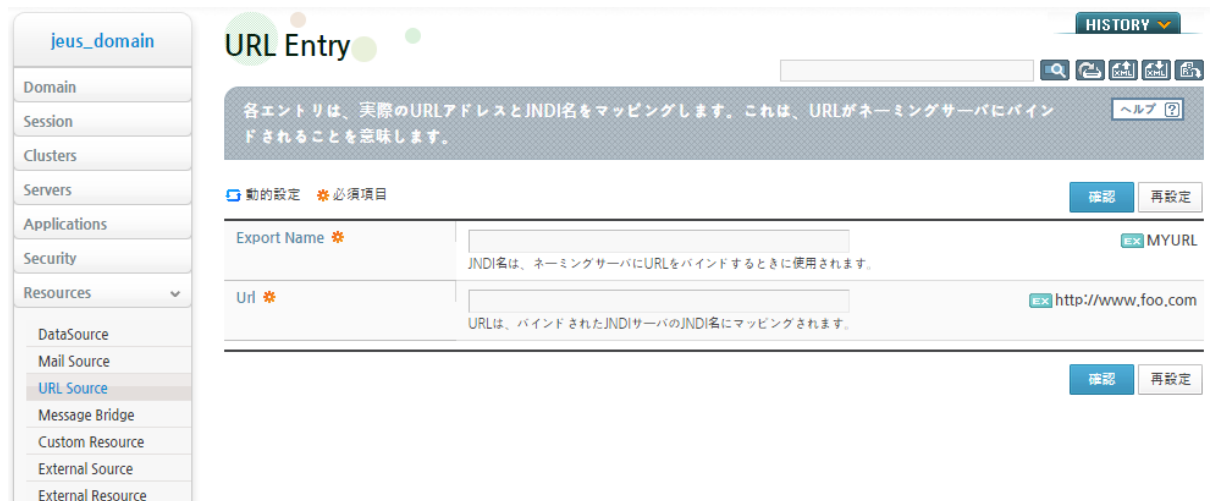
WebAdminの左のメニューから[Resources] > [URL Source]を選択すると、URLソースを照会、登録、変更および削除できる画面が表示されます。

[図 5.4] WebAdminによるURLソースの設定(1)



[Add]ボタンをクリックすると、ドメインにURLソースを登録できます。

[図 5.5] WebAdminによるURLソースの設定(2)



5.2.4. カスタム・リソースの設定

カスタム・リソースは、Java Bean形式のリソースをJNDI ObjectFactoryを使ってルックアップして使用できるようにする一般的なリソースです。本節では、カスタム・リソースの実装方法と登録方法について説明します。

以下は、JavaBean形式のリソース・クラスとリソース・インスタンスを作成するObjectFactoryクラスの例です。このクラスはSERVER_HOME/lib/application、またはDOMAIN_HOME/lib/applicationに存在する必要があります。

[例 5.1] カスタム・リソースの例題クラス

```
package dog;

import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.Name;
import javax.naming.spi.ObjectFactory;

public class DogFactory implements ObjectFactory {
    public DogFactory() {}

    public Object getObjectInstance(Object obj, Name name, Context nameCtx,
        Hashtable<?, ?> environment)
        throws Exception {
        Dog dog = Dog.getInstance();
        System.out.println("Creating a dog whose name is " + dog.getName() + ",
and age is " + dog.getAge());
        return dog;
    }
}
```

[例 5.2] カスタム・リソースを作成するファクトリー・クラスの例

```
package dog;

public class Dog implements java.io.Serializable {
    public static final String DOG_NAME = "wangwang";
    public static final int DOG_AGE = 1;

    private static Dog instance = new Dog();

    private int age = DOG_AGE;
    private String name = DOG_NAME;

    public Dog() {}

    public static Dog getInstance() {
        return instance;
    }

    public int getAge() {
        return age;
    }
}
```

```
}

public void setAge(int age) {
    this.age = age;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public boolean equals(Object anObject) {
    if (this == anObject) {
        return true;
    }
    if (anObject instanceof Dog) {
        Dog anotherDog = (Dog) anObject;
        return (this.age == anotherDog.age && this.name.equals(anotherDog.name));
    }
    return false;
}
}
```

参考

カスタム・リソースを動的に追加するためには、該当クラスがあらかじめサーバーのクラス・ローダーにクラス・パスとして設定されている必要があります。サーバーのクラス・ローダーでこのクラスをロードできない場合は、動的な追加コマンドはペンディング処理されます。そのような場合は、カスタム・リソース・クラスをSERVER_HOME/lib/application、またはDOMAIN_HOME/lib/applicationに追加してサーバーを再起動する必要があります。

カスタム・リソースはグレースフルに削除されません。つまり、実行中の要求があっても、それを完了しないため、ユーザー・アプリケーションでエラーが発生する可能性があることに注意してください。

WebAdminの使用

以下では、WebAdminを使って動的にカスタム・リソースを登録する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [Custom Resource]を選択します。

[図 5.6] WebAdminによるカスタム・リソースの設定(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

DataSource

Mail Source

URL Source

Message Bridge

Custom Resource

External Source

External Resource

Custom Resource

HISTORY

一般的に使用されるリソースを登録します。

Custom Resource

Export Name

Resource Class Name

Add

該当する内容が存在しません。

2. [Add]ボタンをクリックして、ドメインに登録するカスタム・リソースの情報を入力し、[確認]ボタンをクリックします。

[図 5.7] WebAdminによるカスタム・リソースの設定(2)

Custom Resource

HISTORY

一般的に使用されるリソースを登録します。

動的設定 必須項目

確認 再設定

Export Name *

custom/dog

カスタムリソースがネーミングサーバに登録される名前を指定します。

Resource Class Name *

dvt.customresource.Dog

JavaBean形式で実装されたカスタムリソースのクラス名を指定します。

Factory Class Name *

dvt.customresource.DogFactory

カスタムリソースインスタンスを作成するオブジェクトファクトリのクラス名を指定します。

Custom Resource Property

name=value

カスタムリソースのプロパティを設定します。

確認 再設定

3. 設定内容が保存され、結果メッセージが画面上部に表示されます。また、カスタム・リソース・テーブルに追加したリソースの情報が表示されます。

【図 5.8】 WebAdminによるカスタム・リソースの設定(3)

| Export Name | Resource Class Name |
|-------------|------------------------|
| custom/dog | dvt.customresource.Dog |

4. 追加したカスタム・リソースをサーバーで使用できるように、サーバー設定にカスタム・リソースを追加します。

WebAdminの左のメニューで[**Servers**] > [**Server1**] > [**Basic Info**] > **Custom Resource Refs**リストを選択し、追加したカスタム・リソースを選択して[**確認**]ボタンをクリックします。

【図 5.9】 WebAdminによるカスタム・リソースの設定(4)

| Name |
|--|
| <input checked="" type="checkbox"/> custom/dog |

5. [**Activate Changes**]ボタンをクリックして、カスタム・リソースをドメインに反映します。画面上部にカスタム・リソースがドメインに反映されたという結果メッセージが表示されます。

該当カスタム・リソースは追加したサーバーで即時使用可能です。

コンソール・ツールの使用

コンソール・ツールを使ってサーバーに登録されたカスタム・リソースを照会することができます。また、新しいカスタム・リソースを動的に追加、削除することもできます。

【例 5.3】 コンソール・ツールによるカスタム・リソースの照会・追加・削除

```
[DAS]domain1.adminServer>add-custom-resource custom/dog -resource dog.Dog -factory
dog.DogFactory
Successfully performed the ADD operation for A custom resource.
Check the results using "list-custom-resources or add-custom-resource"

[DAS]domain1.adminServer>list-custom-resources
```

```
List of Custom Resources
=====
+-----+-----+-----+-----+
| Export Name | Resource Class | Factory Class | Properties |
+-----+-----+-----+-----+
| custom/dog | dog.Dog | dog.DogFactory | [test=1, |
| | | | test1=2] |
+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>add-custom-resource-to-servers custom/dog -servers server1
Successfully performed the ADD operation for A custom resource.
Check the results using "list-custom-resources"

[DAS]domain1.adminServer>remove-custom-resource custom/dog
Successfully performed the REMOVE operation for A custom resource.
Check the results using "list-custom-resources or remove-custom-resource"

[DAS]domain1.adminServer>list-custom-resources
List of Custom Resources
=====
+-----+-----+-----+-----+
| Export Name | Resource Class | Factory Class | Properties |
+-----+-----+-----+-----+
(No data available)
=====
```

5.2.5. 外部リソースの設定

外部ソースは、大きくJMSソースとコネクタに分けられます。WebAdminの左のメニューから[Resources]>[External Source]を選択すると表示される外部リソース・リスト画面で関連設定を行うことができます。

[図 5.10] WebAdminによる外部リソースの設定



JMSソース

WebAdminを使ってドメインに設定されているJMSソースを照会し、JMSソースを追加、変更、削除することができます。JMSソースを追加するには、外部リソース・リスト画面で**Jms Source**画面の上部にある**[Add]**ボタンをクリックします。

[図 5.11] WebAdminによるJMSソースの追加

JMS Source

HISTORY

JEUSTランザクションマネージャ、IBM MQ、Sonic MQなど、メッセージングリソース製品同士の互換のためには、以下の項目を設定する必要があります。設定に関する詳細情報は、各製品のマニュアルを参照してください。

ヘルプ

動的設定 必須項目

確認 再設定

| | |
|--------------------|---|
| Vendor | <div></div> <div>JMSリソースドライバのベンダ名を指定します。</div> |
| Factory Class Name | <div></div> <div>JMSリソースドライバのファクトリクラス名を指定します。</div> |
| Resource Type | <div></div> <div>JMSリソースタイプを設定します。</div> |
| Export Name | <div></div> <div>JNDIに登録され、サービスされる名前です。</div> |
| Queue | <div></div> <div>リソースタイプがQの場合のみ使用されます。詳細内容は、IBM MQまたはSonic MQのマニュアルを参照してください。</div> |
| QueueManager | <div></div> <div>Tタイプ以外のIBM MQを使用する場合のみ使用されます。詳細内容は、IBM MQのマニュアルを参照してください。</div> |
| Topic | <div></div> <div>リソースタイプがTの場合のみ使用されます。詳細内容は、IBM MQまたはSonic MQのマニュアルを参照してください。</div> |
| Property | <div><div>name:type=value</div><div>JMSリソースの設定に必要なプロパティを記述します。</div></div> |

確認 再設定

以下は、JMSソースの設定項目についての説明です。

| 項目 | 説明 |
|--------------------|---|
| Vendor | JMSベンダーを設定します。次の値のうち1つを設定します – ibmmq : IBM社の製品を示します – sonicmq : Sonic MQを示します – others : その他の製品を示します |
| Factory Class Name | JMSリソースのファクトリー・クラスの名前を指定します |
| Resource Type | JMSのタイプを決定します。 |

| 項目 | 説明 |
|---------------|--|
| | 以下の8つの値のうち1つを設定します – QCF – TCF – Q – T – XAQCF – XATCF – LOCALXAQCF – LOCALXATCF |
| Export Name | JNDIにバインドされる名前を指定します。この名前を使ってJMSのコネクション・ファクトリー、デスティネーションなどを利用できます |
| Queue | 「Resource Type」が「Q」の場合にのみ使用します |
| Queue Manager | 「Resource Type」が「Q」の場合にのみ使用します |
| Topic | 「Resource Type」が「T」の場合にのみ使用します |
| Property | JMSリソースに必要な属性を記述します。この設定は、name、type、valueで構成されます |

参考

各タグについての詳細はIBM MQまたはSonic MQのマニュアルを参照してください。

コネクタの追加

WebAdminを使ってドメインに設定されているコネクタを照会し、コネクタを追加、変更、削除することができます。コネクタについての詳細は『JEUS JCAガイド』を参照してください。

5.2.6. 外部リソースの設定

JEUSではTmaxやInfinite Cacheのような他製品との連携するために外部リソースを設定することができます。Tmaxとの連携のためには、JEUSからTmaxに接続してTmaxのトランザクション・サービスを使用するアウトバウンド(Outbound)のWebTと、TmaxからJEUSに送られたサービス要求を受けるインバウンド(Inbound)のJTmaxを設定することができます。詳しい内容は関連するTmaxマニュアルの『JTmaxServerユーザーガイド』を参照してください。

本節では外部リソースの実装方法と登録方法について説明します。

以下は、jeus.external.ResourceBootstrapperインターフェースです。このインターフェースを実装したクラスは、SERVER_HOME/lib/applicationまたはDOMAIN_HOME/lib/applicationに存在する必要があります。

[例 5.4] jeus.external.ResourceBootstrapper

```
package jeus.external;

import javax.naming.Context;
import java.util.Map;

/**
 * 外部リソースをJEUSでできるようにするブートストラッパーです。
 */
public interface ResourceBootstrapper {

    /**
     *
     * @param propertyMap resourceの設定を持つMap
     */
    void setProperties(Map propertyMap) throws InvalidPropertyException;

    /**
     * リソースをバインドします。
     * @param context
     */
    void initResources(Context context);

    /**
     * 使用できるプロパティの情報を返します。
     * @return
     */
    ResourcePropertyInfo[] getPropertyInfo();

    /**
     *
     * @param propertyMap 変更されるプロパティがあるMap
     */
    void modifyProperties(Map propertyMap) throws InvalidPropertyException;

    /**
     * リソースを再度バインドする際に呼び出されます。
     * @param context
     */
    void reconfigResources(Context context);

    /**
     * リソースを削除する際に使用します。
     * @param context
     */
    void destroyResources(Context context);
}
```

参考

外部リソースを動的に追加するためには、該当クラスがあらかじめサーバーのクラス・ローダーにクラス・パスとして設定されている必要があります。サーバーのクラス・ローダーでこのクラスをロードできない場合は、動的な追加コマンドはペンディング処理されます。そのような場合は、外部リソース・クラスをSERVER_HOME/lib/application、またはDOMAIN_HOME/lib/applicationに追加してサーバーを再起動する必要があります。

外部リソースはグレースフルに削除されません。つまり、実行中の要求があっても、それを完了しないため、ユーザー・アプリケーションでエラーが発生する可能性があることに注意してください。

WebAdminの使用

以下では、WebAdminを使って動的に外部リソースを登録する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [External Resource]を選択します。[Add]ボタンをクリックして、JTmaxとWebT、Inifinite Cacheのためのリソース・ブートストラッパーをドメインに登録します。

[図 5.12] WebAdminによる外部リソースの設定(1)



2. 登録する外部リソースの設定情報を入力して[確認]ボタンをクリックします。

[図 5.13] WebAdminによる外部リソースの設定(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

DataSource

Mail Source

URL Source

Message Bridge

Custom Resource

External Source

External Resource

Concurrency Utilities

Resource

External Resource

HISTORY

ドメイン内で使用される外部リソースの設定を行います。 JEUSと連携されるWebT、jTmax、またはInfiniteCacheの設定を行います。

ヘルプ ?

動的設定 必須項目

確認 再設定

Name * test/ext
外部リソースの名前を指定します。一意の名前である必要があります。

Class Name * dvt.externalresource.TestResourceBootstrapper
WebT、jTmax、またはInfiniteCacheのResourceBootstrapper実装クラス名を指定します。

Property
外部リソースで使用するプロパティを設定します。

name=value

確認 再設定

3. [確認]ボタンをクリックして一時保存をすると、結果メッセージが画面上部に表示されます。また、外部リソース・テーブルに追加するリソースの情報が表示されます。

[図 5.14] WebAdminによる外部リソースの設定(3)

External Resource

HISTORY

ドメイン内で使用される外部リソースの設定を行います。 JEUSと連携されるWebT、jTmax、またはInfiniteCacheの設定を行います。

ヘルプ ?

追加されました。

External Resource

| Name | Class Name | |
|----------|---|--------|
| test/ext | dvt.externalresource.TestResourceBootstrapper | Delete |

Add

4. 追加した外部リソースをサーバーで使用できるように、サーバー設定に外部リソースを追加します。

WebAdminの左のメニューで[Servers] > [Server1] > [Basic Info] > External Resource Refsリストを選択し、追加した外部リソースを選択して[確認]ボタンをクリックします。

[図 5.15] WebAdminによる外部リソースの設定(4)

External Resource Refs

サーバまたはクラスタで有効なリソースを指定します。

| Name | Class Name |
|----------|---|
| test/ext | dvt.externalresource.TestResourceBootstrapper |

サーバまたはクラスタで有効なリソースのIDを設定します。

5. **[Activate Changes]**ボタンをクリックして、外部リソースをドメインに反映します。画面上部に外部リソースがドメインに反映されたという結果メッセージが表示されます。

該当外部リソースは追加したサーバーで即時使用可能です。

コンソール・ツールの使用

コンソール・ツールを使ってサーバーに登録された外部リソースを照会することができます。また、新しい外部リソースを動的に追加、削除することもできます。

[例 5.5] コンソール・ツールによる外部リソースの照会・追加・削除

```
[DAS]domain1.adminServer>add-external-resource test/ext -resource
test.ext.TestResourceBootstrapper
Successfully performed the ADD operation for A external resource.
Check the results using "list-external-resources or add-external-resource"
```

```
[DAS]domain1.adminServer>list-external-resources
List of External Resources
=====
+-----+-----+-----+-----+
| Export Name | Resource Class | Properties |
+-----+-----+-----+-----+
| test/ext | test.ext.TestResourceBootstrapper | [] |
+-----+-----+-----+-----+
=====
```

```
[DAS]domain1.adminServer>add-external-resource-to-servers test/ext -servers server1
Successfully performed the ADD operation for A external resource.
Check the results using "list-external-resources"
```

```
[DAS]domain1.adminServer>remove-external-resource test/ext
Successfully performed the REMOVE operation for A external resource.
Check the results using "list-external-resources or remove-external-resource"
```

```
[DAS]domain1.adminServer>list-external-resources
List of External Resources
=====
+-----+-----+-----+-----+
| Export Name | Resource Class | Properties |
+-----+-----+-----+-----+
(No data available)
=====
```

第6章 DBコネクション・プールとJDBC

本章では、JEUSが提供するJDBCコネクション・プールの基本メカニズムとJDBCコネクション・プールの使用方法、JEUSドメイン構造におけるデータソースの管理方法について説明します。

6.1. 概要

Webアプリケーションは情報を格納する必要がある場合に主にデータベースを利用します。また、JEUSのようなWebアプリケーション・サーバー(web application server、以下WAS)は、アプリケーションにコネクション・プーリングなどのデータベースに依存するサービスを提供するためにデータベースとの通信が必要です。このような必要をより体系的かつ効率的に満たすために、DBクライアントとDB間のインターフェースを定義したものがJava Database Connectivity(JDBC)標準です。

JDBC標準はアプリケーションのDBコネクションの使用方法やSQLの実行方法について記述したもので、関連APIを提供します。JDBC標準についての詳細は、Sun JDBCページ(<http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html>)を参照してください。

JEUSではJDBC 4.0をベースに、アプリケーションにコネクション・プーリングとその他の付加サービスを提供します。

6.2. データソースとJDBCコネクション・プーリング

データソースはアプリケーションにJDBCコネクションを提供するファクトリーで、抽象化されたオブジェクトです。内部でJDBCコネクション・プールを構成し、コネクション・プーリング・サービスを提供します。

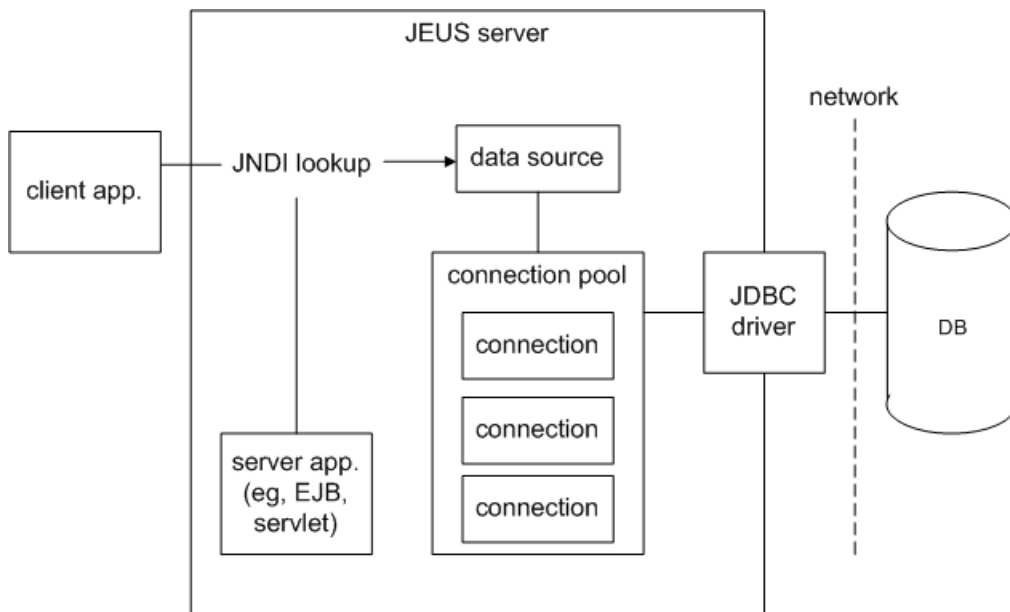
JDBCコネクション・プーリングはJDBCコネクションの管理サービスです。データベースから一定数のコネクションを取得して保管し、アプリケーションが必要とするときに提供します。また、使用されたコネクションを回収して再活用します。

アプリケーションはデータベースに直接コネクションを要求するのではなく、JNDIルックアップによりデータベースに関連付けられたデータソースを取得し、そのデータソースにコネクションを要求します。コネクションの要求を受けたデータソースは、自身のコネクション・プールからコネクションを取り出してアプリケーションに渡してコネクション要求を処理します。

JEUSはアプリケーションで使用が終わったコネクションを回収し、以降再活用できるように再びコネクション・プールに入れて管理します。

以下は、JEUSサーバーにおけるJDBCコネクション・プーリングのメカニズムを簡単に表した図です。

[図 6.1] JEUSのコネクション・プーリング



参考

JEUSは起動の際にコネクション・プールを作成および初期化するのではなく、アプリケーションからデータソースへのコネクション要求が最初に発生したときにコネクション・プールを作成および初期化します。

6.2.1. JDBCドライバー

JDBCドライバーはデータベースとの通信に必要なAPI実装です。JEUSはJDBC認証を受けたドライバーと連携してコネクション・プーリングやその他のサービスを提供できます。認証を受けたドライバーのタイプと種類は、SunのJDBCドライバー関連ページ(<http://devapp.sun.com/product/jdbc/drivers>)で確認できます。

JEUSのJDBC環境構成は、どのベンダーのデータベースを使用するかによって異なります。それは、ドライバーごとに要求する属性が異なるからであり、調整が必要な詳細属性は各ベンダーのJDBCドライバーのマニュアルを確認する必要があります。

6.2.2. JDBCコネクション・プール

JDBCコネクション・プールはDBコネクションをより効率的に使用・管理するためのフレームワークであり、コネクション・プーリング・サービスを提供する主体です。

コネクション・プールを使用するメリットは次の2つに要約できます。

- **高性能**

システムの性能を向上できます。DBコネクションの作成と削除は本来システムに負担を与えるので、コネクション・プールにコネクションをあらかじめ作成して置いてそれを再活用すると、コネクション・プーリングを使用せずにコネクションを頻繁に作成・削除することで発生するオーバーヘッドをかなり削減できます。

- **コネクション管理**

同時コネクションの数を制御できます。同時コネクションの数を設定値以下に抑えることで、データベースに無理な負荷をかけることを避けられます。

6.2.3. データソース

データソースはアプリケーションとコネクション・プール間のインターフェースです。アプリケーションはデータソース・オブジェクトをDBコネクションのファクトリーとみなします。

データソースによるコネクションの使用方法は、`java.sql.DriverManager`によるコネクションの使用方法より多くのメリットを提供します。

本節で説明するデータソースの特性と機能をよく考慮し、適切なデータソースを使用することで、サービスの性能向上を図ることができます。

データソースのタイプ

データソースは次の3つのタイプに分類されます。

- **基本データソース**

`javax.sql.DataSource`タイプです。このタイプはコネクション・プーリングに使用できない基本データソース・タイプです。

- **コネクション・プール・データソース**

`javax.sql.ConnectionPoolDataSource`タイプです。JEUSではこのタイプに対しコネクション・プールを提供します。

主に以下のような状況で使用できます。

- XAを使用する必要がなく、簡単にデータベースにアクセスするアプリケーションを作成する場合
- Auto Commitをfalseに設定して直接ローカル・トランザクションをコントロールする場合

```
import java.sql.Connection;

Connection conn = datasource.getConnection();
conn.setAutoCommit(false);
```

```
...DBアクセス・コード... // JDBCドライバ内部でローカル・トランザクションを実行  
conn.commit();
```

● XAデータソース

javax.sql.XADataSourceタイプです。JEUSではこのタイプに対しコネクション・プールを提供します。このタイプのデータソースはコネクション・プーリングと共に、グローバル・トランザクション(以下、XA)連携をサポートします。

主に以下のような状況で使用できます。

- Servlet/EJBのようなJava EEコンポーネント・ロジックが2つ以上のデータベースにアクセスする場合
- Java EEコンポーネントが1つのデータベースにアクセスする場合でも、関連ロジックが一連の動作で行われる必要がある場合

コネクション・プール・データソースに対するXAエミュレーション

XA処理性能を高めるために、ローカル・トランザクションの最適化機能を使用できます。同機能は、コネクション・プール・データソースから取得したコネクションをXAに参加するようにエミュレーションします。

主に以下のような状況で使用できます。

- データベースがXAをサポートしない場合や、JDBCドライバがjavax.sql.XADataSourceの実装を提供しない場合
- アプリケーションでXAが必要であるけれど、性能問題のためXAデータソースを使用したくない場合(つまり、ローカル・トランザクションの最適化が必要な場合)

現在は、XAをサポートしないデータベースやJDBCドライバはあまりないので、主にローカル・トランザクションの最適化に使用することになります。しかしこの機能には、1つのXAには最大1つのローカルXAデータソースだけ参加できるという制約があり、そのことを必ず考慮する必要があります。そのため、ローカルXAデータソースはなるべく下記のような状況で使います。

- Servlet/EJBのようなJava EEコンポーネント・ロジックが1つのデータベースだけ使用するため、XAデータソースを使用する必要がない場合
- Java EEコンポーネントで複数のデータベースを使用しているも、そのうち1つをローカル・トランザクションで処理し、XA性能を高めたい場合

複数のデータベースを使用する状況で注意すべきことは、XAエミュレーション機能を使用するコネクション・プール・データソースは実際に2 Phase Commit(以下、2PC)をサポートするのではないので、トランザクションの回復が正常に行われないこともあります。

参考

コネクション・プール・データソースのXAエミュレーションはAuto Commitを無効にして使用するローカル・トランザクションとして扱われ、データベースの観点から見れば、JEUSTランザクション・マネージャーが管理するXAトランザクションとは異なるトランザクションです。その代わり、JEUSがそのローカル・トランザクションがXAトランザクションに参加できるようにエミュレーションするので、アプリケーションの観点からは1つのトランザクションとしてみなします。

6.2.4. クラスター・データソース

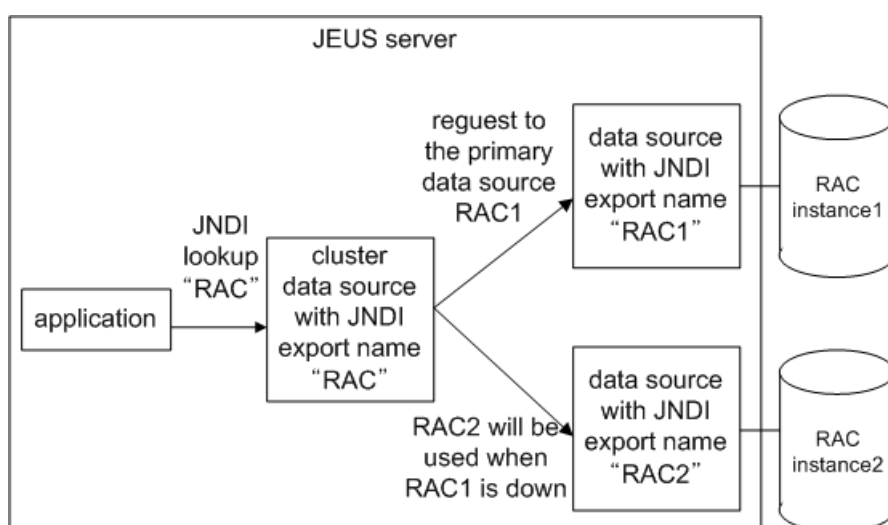
本節では、クラスター・データソースのフェイルオーバー、XAアフィニティ、ONSの使用について説明します。

6.2.4.1. フェイルオーバー

RACインスタンス間のフェイルオーバーやフェイルバックをサポートするために、JEUSではクラスター・データソースを提供します。RAC(Real Application Cluster)はOracleが提供するDBクラスタリング機能です。RACの詳細についてはOracleの文書を参照してください。

クラスター・データソースは1つのJNDI名をもつデータソース・インスタンスです。このインスタンスは複数の個別データソースを論理的にまとめて管理します。クラスター・データソースに属する個別データソース(以下、コンポーネント・データソース)もそれぞれのJNDI名を持つ独立したデータソースであり、必ずRACインスタンスのデータソースとして設定される必要があります。クラスター・データソースはアプリケーションから入ってきたコネクション要求をコンポーネント・データソースの中の1つに委任する方式で動作します。コネクション要求はそれを処理するのに問題がないと確認されたコンポーネント・データソースに委任されるため、クラスター・データソースは透過的にアプリケーションにフェイルオーバーすることができます。

[図 6.2] RACに対するクラスター・データソースのフェイルオーバー



Oracle JDBCドライバー・レベルで提供するCTF(Connect Time Failover)より、JEUSのクラスター・データソースを使用することを推奨します。Oracle CTFはコネクション別にフェイルオーバーを実行するため、デー

タソース全体に問題が生じた場合、コネクション・プールに存在するすべてのコネクションを復旧するのに長い時間がかかることがあります。しかし、JEUSクラスター・データソースはコンポーネント・データソースに問題が発生したことを検知して、データソース単位でフェイルオーバーを実行するのでより効率的です。さらに、JEUSクラスター・データソースは自動フェイルバック機能も提供します。

JEUSクラスター・データソースを使用する方式とOracle JDBCドライバーにRAC属性を設定する方式は、その動作が異なります。

前者はJEUSがフェイルオーバーを処理するので、各RACインスタンスのコンポーネント・データソースのfailの有無を監視するためにcheck-queryとcheck-query-periodを設定する必要があります。一方、後者はドライバーがフェイルオーバーを担当するため、check-queryが必須ではありません。クラスター・データソースの設定についての詳細は、「[6.5. クラスター・データソースの設定](#)」を参照してください。

6.2.4.2. XAアフィニティ

グローバル・トランザクションがRACで行われる場合、物理的に分散された異なるRACインスタンスで処理されるよりは、できるだけ1つのRACインスタンスで処理されるのが性能の面で有利です。

グローバル・トランザクション処理においてクラスター・データソースは、最初のコネクション要求が処理されたコンポーネントや関連のRACインスタンスを記憶しておきます。以降のコネクション要求は無条件で当該コンポーネント・データソースに委任することで、XAアフィニティ、つまり1つのRACインスタンスを対象にしてグローバル・トランザクションが行われます。

複数のクラスター・データソースが同じRACを参照する場合でも、各々のクラスター・データソースへのコネクション要求は、それが異なるクラスター・データソースへのコネクション要求であっても、同じRACインスタンスで処理されるようにします。また、複数のRACが関連付けられている場合においても、各RACインスタンスへのXAアフィニティを保持します。

クラスター・データソースaとクラスター・データソースbがそれぞれ異なるRAC AとRAC Bを参照している場合、クラスター・データソースaへのコネクション要求はRAC Aで、クラスター・データソースbへのコネクション要求はRAC BでXAアフィニティを保証します。異なるRACであるRAC AとRAC Bの間でXAアフィニティを保持することは根本的に不可能なので、その点に注意する必要があります。

XAアフィニティが設定されている場合、既存のフェイルオーバー、フェイルバック、ロードバランシングの方式は無視され、XAに関連付けられたRACインスタンスと同じデータソースを優先的に選択します。XAで選択するデータソースが存在しない場合は、既存の方式に従います。

6.2.4.3. ONSを用いたクラスター・データソース

ONS(Oracle Notification Service)は、RACノード同士で状態情報を共有できるようにするOracleの通知サービスです。RACにONSが設定されていると、基本的に各RACノードの状態をすべてのRACノードが共有することになります。一方、RACサーバー・コンポーネントでなくても、ONSクライアントとしてONSに参加すると、ONSからRACノードの状態情報を一部取得することができます。JEUSでは、ONSを用いた高機能のクラスター・データソースを提供しています。ONSを用いたクラスター・データソースは、内部で各コンポーネント・データソースと関連したRACインスタンスをマッピングしています。このような構造により、ONSから取

得したRACインスタンスの情報を、関連付けられたコンポーネント・データソースの管理に利用することができます。

以下は、JEUSがONSクライアントとしてONSから取得できる状態情報です。

1. RACインスタンスの起動と終了の通知 : RACインスタンスが起動しているか、あるいは終了しているかを通知します。
2. RACインスタンスごとの使用可能な量の割合 : ランタイム・ロードバランシング・アドバイザリーとも呼ばれるこの情報は、RAC全体の使用可能な量を100とし、RACインスタンスごとの使用可能な量を割合で通知します。

ONSを用いたクラスター・データソース機能

以下は、ONSを用いたクラスター・データソースで提供する機能についての説明です。

- より効果的なコンポーネント・データソースの状態管理

以前はDBの状態を把握するためにポーリング方式を使用していました。この方式は要求の処理中に実行されるため、性能が低下したり、ネットワークが切断された場合にはDBの状態が把握できなかったりする問題がありました。RACインスタンスの起動と終了の通知(Up & Down Notification)を利用すると、これらの問題を回避することができ、かつ、コンポーネント・データソースの失敗や復旧の可否をより効果的に検知することができます。

- ランタイム・ロードバランシング・アドバイザリーを利用した効果的なロードバランシング

以前はDBの使用可能な量を判断できなかったため、単純なラウンドロビン方式のロードバランシングしか提供できませんでした。今は、ランタイム・ロードバランシング・アドバイザリーを利用してリアルタイムでRACインスタンスごとの相対的な使用可能な量を把握できるようになったため、それを基にした効率的なロードバランシングが可能となりました。たとえば、コンポーネント・データソースAとコンポーネント・データソースBがそれぞれRACインスタンスA、RACインスタンスBと関連付けられ、これらの使用可能な量がランタイム・ロードバランシング・アドバイザリーにより、各々60%と40%と通知された場合、以降のコネクション要求は3:2の割合でコンポーネント・データソースAとコンポーネント・データソースBに委任されます。

ONSの設定

JEUSをONSクライアントとして使用するためには、ONSライブラリーをインストールする必要があります。

Oracleのサポート・ページからons.jarをダウンロードして、\$JEUS_HOME/lib/datasourcesディレクトリーに格納します。次に、クラスター・データソースにONS関連の設定を行うと、ONSを用いたクラスター・データソースを使用することができます。

最初にONS上の各RACノードがONS通信に使用するIPとポート番号を設定します。ここまで設定すると、クラスター・データソースはONSと結合されます。これにより、フェイルオーバー、フェイルバック、ロードバラン

シングの方式において、クラスター・データソースはONSを利用してコンポーネント・データソースの失敗や復旧の可否を効率よく検知することができます。特に、ロードバランシング方式の場合は、ランタイム・ロードバランシング・アドバイザーを利用すると、効率的なロードバランシングが可能になります。

6.3. データソースとコネクション・プールの管理

JEUSがドメイン構造に拡張されたことによって、データソースとコネクション・プールの管理構造にも一部変化がありました。しかし、拡張されたドメイン構造を理解すれば、データソースとコネクション・プールの変更された管理構造を学ぶのもそれほど難しくはありません。

JEUSでサービスを実行する最小単位は**サーバー**であり、場合によって複数のサーバーがクラスタリングされてサービスを実行することもできます。さらに、これらのサーバーとクラスターは1つの管理単位でまとめることができますが、そのような管理単位が**ドメイン**です。つまり、ドメインは関連付けて管理する必要があるサーバーとクラスターの集合を意味します。ドメインには、ドメインに属するサーバーとクラスターを管理し、関連する諸般サービスを提供する特別な役割を果たす唯一のサーバーがありますが、これが**DAS**です。ドメインに属するDAS以外のすべてのサーバーは**MS**といいます。DASはDASとして機能する同時にMSとして機能することもできますが、サービスはMSを通じて提供し、DASは本来のドメイン管理の役割だけを行う方式使用することを推奨します。JEUSのドメイン構造についての詳細は、『JEUS ドメインガイド』を参照してください。

本節では、ドメイン構造でデータソースとコネクション・プールを管理する方法について説明します。

データソースは基本的にドメイン範囲で使用されるリソースです。ドメインに属するサーバーとクラスター(より正確に言うと、クラスターに属するサーバー)はデータソースの設定を参照して独自のコネクション・プールを作成し、コネクション・プール・サービスを提供します。クラスターはサーバーのグループを抽象化したもので、サービスを提供する実質的な主体ではないので、クラスターで参照するデータソースは実際にはそれに属するサーバーで有効です。そのため、クラスターに属するサーバーはクラスターで参照するデータソースを参照できます。また、クラスターに属するサーバーは自動的にデータソースを参照するように設定することができます。この場合、サーバーはクラスターで参照するデータソースと自身が参照するデータソースすべてを有効なデータソースとして使用できるようになります。

サーバーやクラスターによるデータソースの参照は、参照するデータソースのIDを明示的に登録することで可能になります。つまり、サーバーまたはクラスターが特定のデータソースIDを指定すると、当該サーバーと当該クラスターに属するサーバーは指定したデータソースの設定を読み込んでコネクション・プールの作成に必要な情報を構成し、それをデータソースのJNDI名にマッピングして自身のJNDIリポジトリにバインドします。このプロセスが終わると、サーバーにデプロイされたアプリケーションはバインドしたJNDI名でデータソースをルックアップしてコネクション・プールを作成して使用できるようになります。

参考

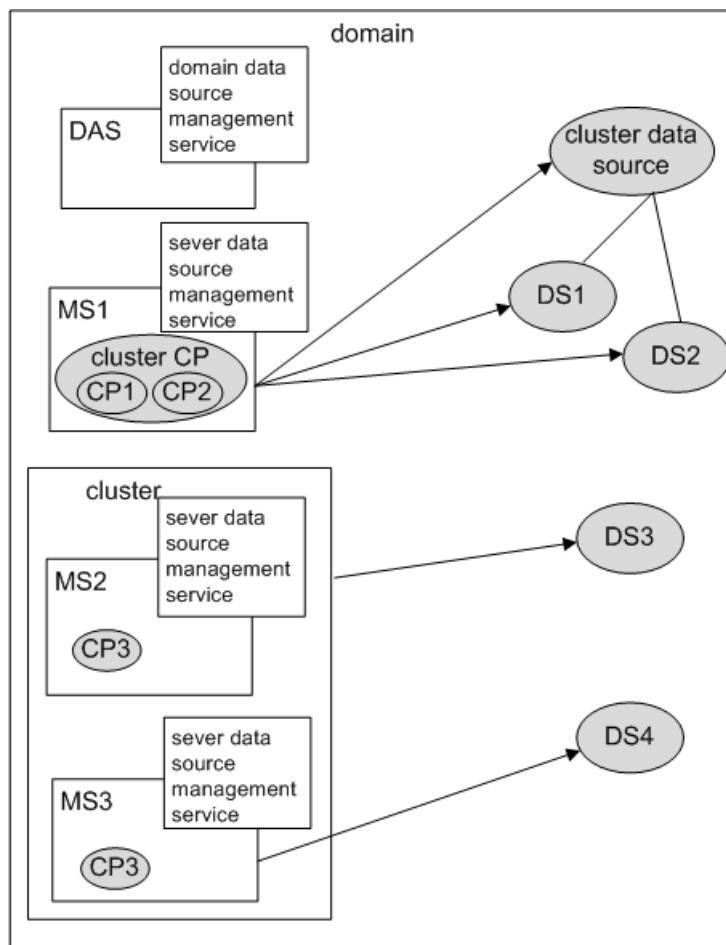
データソースのJNDIバインドはサーバー単位で行われるため、相互異なるデータソースであっても、同じサーバーにJNDIバインドされないことを保証できれば、同じJNDI名を持つことができます。これは、すな

わち同じJNDI名を持つ異なるデータソースを任意のサーバーで同時に参照するように設定することは許容されないことを意味します。

このように異なるデータソースが同じJNDI名を持つことが可能なため、ドメインではデータソースIDでデータソースを識別します。データソースIDはドメインで唯一な値を設定して、データソースの識別子として機能できる必要があります。

下の図は、DASと3つのMSで構成されたドメイン構成の例です。

【図 6.3】 JEUSDメイン構造におけるデータソースとコネクション・プールの管理



MS1はクラスター・データソースを登録しています。クラスター・データソースのコンポーネント・データソースはDS1とDS2です。MS1がクラスター・データソースを正常に使用するためには、コンポーネント・データソースのDS1とDS2もMS1に登録されている必要があります。このように設定することで、MS1はDS1とDS2のコネクション・プールをそれぞれ作成してクラスターにし、クラスター・コネクション・プール・サービスを提供できます。

一方、MS2とMS3はクラスタリングされています。このクラスターはデータソースDS3を登録しているので、DS3はクラスターに属するMS2とMS3の両方で有効です。これは、MS2とMS3がそれぞれDS3のコネクション・プールを作成してコネクション・プール・サービスを提供できることを意味します。

また、MS3は独自でDS4を登録しています。したがって、MS3はクラスターで登録したDS3と共に、自身が登録したDS4に対してもコネクション・プール・サービスを提供することができます。

MSはサーバー・レベルでデータソースとコネクション・プール管理サービスを行い、DASはドメイン・レベルでデータソースとコネクション・プール管理サービスを行います。DASサービスはそれぞれのMSサービスと連携して、ドメインに存在するすべてのデータソースとコネクション・プールを総括的に管理し、コンソール・ツールまたはWebAdminからのデータソースとコネクション・プール関連の要求を処理します。

6.4. データソース設定

データソースとJDBCコネクション・プールを実際にJEUSで使用するためには、最初に\$JEUS_HOME/lib/datasource/ディレクトリー内にJDBCドライバ・ライブラリーが存在するか確認し、データソースの構成に必要な情報を設定します。データソース設定は、JDBCドライバの設定などに必要な基本設定とコネクション・プール設定を含みます。

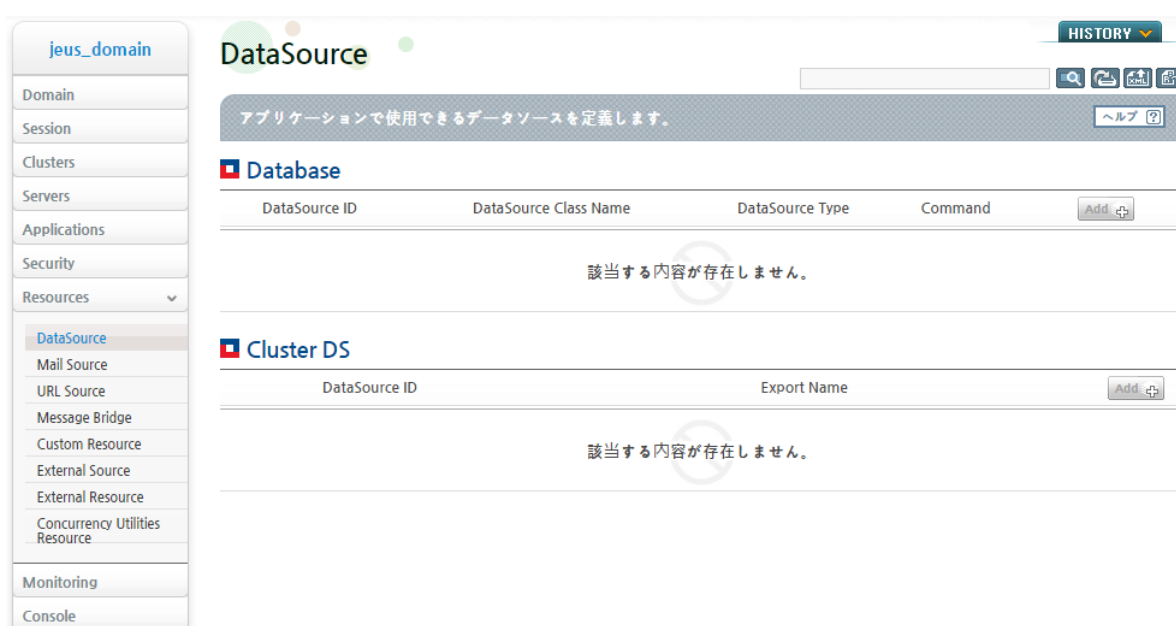
コンソール・ツールまたはWebAdminを使って設定を行えます。本節では、WebAdminを使用した設定方法についてのみ説明します。コンソール・ツールを使用した設定方法については、『JEUS リファレンスガイド』の「4.2.11.1. add-data-source」を参照してください。

6.4.1. 基本設定

以下では、WebAdminを使ってデータソースを設定する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [Data Source]を選択して[LOCK & EDIT]ボタンをクリックした後、表示されるDatabaseリストで[Add]ボタンをクリックすると、データソース基本設定画面に移動します。

【図 6.4】データソースの基本設定画面(1)



2. 基本設定はデータベースのアクセス情報など、主にJDBCドライバーの構成に関連する設定を含みます。その他の付加設定は画面下部の**詳細設定**で確認できます。

[図 6.5] データソースの基本設定画面(2)

jeus8

Domain

Session

Clusters

Servers

Applications

Security

Resources

DataSource

Mail Source

URL Source

Message Bridge

Custom Resource

External Source

External Resource

Concurrency Utilities Resource

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

1 Shutdown

0 Suspended

0 Other

Runtime Info

Activate Changes

Undo All Changes

変更された設定を保存、またはキャンセルする機能です。

運用マニュアル

Domain

Server

もっと見る

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

Data Source Id

データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。

Export Name

データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。

Vendor

JDBCドライバベンダの名前を指定します。

Data Source Class Name

JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。

Data Source Type

データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。

Server Name

DBが実行されるホスト名、またはIPを設定します。

Port Number

DBリスナーのポート番号を設定します。

Database Name

DBの名前を指定します。Oracleの場合、DBのSIDを設定します。

User

DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。

Password

DBユーザのパスワードを設定します。暗号化して保存するときは、{algorithm}ciphertext,の形式で記述します。

Support Xa Emulation

[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEU56までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。

詳細設定

すべてを開く

Description

Login Timeout

Isolation Level

Auto Commit

Stmt Query Timeout

Pool Destroy Timeout

Property

name:type=value

Action On Connection Leak

確認

再設定

注

サービス名、ネットワーク・プロトコル、ドライバー・タイプなど特定のJDBCドライバーに依存的なプロパティは、**詳細設定**の「**Property**」項目で設定します。設定方法については下の表の説明を参照してください。

以下は、各設定項目についての説明です。

| 項目 | 説明 |
|------------------------|---|
| Data Source Id | データソースのIDです。1つのドメインでデータソースIDはデータソースの唯一な識別子として動作するように設定する必要があります |
| Export Name | データソースのJNDI名です。異なる2つのデータソースが別々のサーバーにJNDIバインドされることを保証できれば、該当データソースは同じJNDI名を持つことができます。これは任意のサーバーで同じJNDI名を持つ異なるデータソースを許容しないことを意味します。この項目を設定しない場合、データソースIDをJNDI名として使用します |
| Data Source Class Name | JDBCドライバー・データソース・クラスの名前です。パッケージ名を含む完全な形式で入力します |
| Data Source Type | データソースのタイプです <ul style="list-style-type: none">DATA_SOURCE : コネクション・プーリング・サービスを提供しませんCONNECTION_POOL_DATA_SOURCE : コネクション・プーリング・サービスを提供しますXA_DATA_SOURCE : コネクション・プーリング・サービスと共に、XA連携をサポートします |
| Vendor | JDBCドライバー・ベンダーの名前です |
| Server Name | データベースが実行されるホスト名またはIPです |
| Port Number | DBリスナーのポート番号です |
| Database Name | データベースの名前です。Oracleの場合は、データベースのSIDです |
| User | データベース・ユーザーのIDです。トランザクション処理などのためには、十分な権限を持っている必要があります |
| Password | データベース・ユーザーのパスワードです。パスワードを暗号化して保存する場合は、「{algorithm}ciphertext」の形式で入力します |
| Support Xa Emulation | コネクション・プール・データソース・タイプのデータソースにのみ有効な設定です。この設定を適用する場合、コネクション・プール・データソースのコネクションがグローバル・トランザクション(XA)に参加するようにエミュレーションします。関連内容は 6.2.3節「コネクション・プール・データソースに対するXAエミュレーション」 を参照してください |
| Description | データソースに関する説明です |

| 項目 | 説明 |
|---------------------------|---|
| Login Timeout | データベースとコネクションを確立するとき、ログインするまで待つ最大の時間です(単位: 秒) |
| Isolation Level | java.sql.Connectionで定義するトランザクションのIsolation設定です |
| Auto Commit | <p>データソースの基本自動コミットの設定で、TRUE、FALSE、DRIVERのいずれかを設定します。</p> <p>DRIVERを設定すると、JEUSはAuto Commit設定に関与せず、JDBCドライバーのAuto Commit設定に従います。XAエミュレーション機能を使用するコネクション・プール・データソースまたはXAデータソースの場合は、トランザクションが連携されない場合にのみこの設定値が有効です</p> |
| Stmt Query Timeout | <p>データソースから取得したコネクションによって生成されたjava.sql.Statementオブジェクトに指定するクエリー・タイムアウトです(単位 : ms)。</p> <p>JEUSは、JDBC APIによって定義され、JDBCドライバー・ベンダーによって実装されたクエリー・タイムアウト設定メソッド(java.sql.Statement#setQueryTimeout)を呼び出します。JDBCドライバー・ベンダー別にクエリー・タイムアウトの適用に多少差があります</p> |
| Pool Destroy Timeout | <p>コネクション・プールの破棄の完了を待つ時間です(単位 : ms)。</p> <p>データソースを定義したアプリケーションをアンデプロイする場合や、サーバーを終了する場合にコネクション・プールを破棄しますが、その過程でデータベースとネットワーク通信を行うときにブロックされる可能性があり、コネクション・プールの破棄を待ち続ける問題が発生することがあります。このような問題を避けるために使用する設定です。設定した時間までコネクション・プールの破棄の完了を待つからアプリケーションのアンデプロイまたはサーバーの終了を続けます</p> |
| Property | JDBCドライバーごとにばらばらの属性の設定をすべて受け入れるために提供される統一された設定方法です。1つの属性に対し、「名前:タイプ=値」の形式で入力し、複数の属性を設定する場合はコンマ(,)で区切ります |
| Action On Connection Leak | <p>アプリケーション(主にステートレス・コンポーネント - Servlet/JSP、ステートレス・セッションBean、MDB)が使用した後に正しく返していないJDBCコネクションを検知したときに、JEUSが取るアクションを定義します。設定しない場合、サーバーの設定に従います</p> <ul style="list-style-type: none"> – NO_ACTION : 何のアクションも取りません。WARNINGに設定すると、返されていないJDBCコネクション情報をログで残します – AUTO_CLOSE : 返されていないJDBCコネクション情報をログに残し、当該JDBCコネクションを回収します |

6.4.2. コネクション・プールの設定

以下では、WebAdminを使ってコネクション・プールを設定する方法について説明します。

基本設定を完了して[確認]ボタンをクリックすると、**Database**リストに基本設定が完了したデータソースが追加されたことを確認できます。

[図 6.6] コネクション・プール設定画面(1)

jeus8

Domain
Session
Clusters
Servers
Applications
Security
Resources

DataSource
Mail Source
URL Source
Message Bridge
Custom Resource
External Source
External Resource
Concurrency Utilities
Resource

Monitoring
Console

DataSource

HISTORY

アプリケーションで使えるデータソースを定義します。

追加されました。

Database

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|-----------------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | <div>Delete Duplicate</div> |

Cluster DS

| DataSource ID | Export Name | |
|----------------|-------------|--|
| 該当する内容が存在しません。 | | |

追加されたデータソースのデータソースIDをクリックすると、再び基本設定画面に移動します。画面で **[Connection Pool]** タブをクリックすると、コネクション・プール設定画面に移動します。

[図 6.7] コネクション・プール設定画面(2)

jeus8

Domain

Session

Clusters

Servers

Applications

Security

Resources

DataSource

Mail Source

URL Source

Message Bridge

Custom Resource

External Source

External Resource

Concurrency Utilities Resource

Monitoring

Console

システム状態

0 Failed

0 Standby

1 Running

1 Shutdown

0 Suspended

0 Other

Runtime Info

Activate Changes

Undo All Changes

変更された設定を保存、またはキャンセルする機能です。

運用マニュアル

Domain

Server もっと見る

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定 必須項目

確認

再設定

Data Source Id

ds1

データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。

Export Name

データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。

Vendor

oracle

入力

JDBCドライバベンダの名前を指定します。

Data Source Class Name

oracle.jdbc.pool,OracleConnectionPoolDataSource

oracle.jdbc.pool,OracleConnectionPoolDataSource

JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。

Data Source Type

ConnectionPoolDataSource

データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。

Server Name

192.168.1.165

DBが実行されるホスト名、またはIPを設定します。

Port Number

1521

DBリスナーのポート番号を設定します。

Database Name

ora10g

DBの名前を指定します。Oracleの場合、DBのSIDを設定します。

User

jeustest1

DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。

Password

.....

入力

(DES)FQrLbQ/D8O1IDV571L28rw==

DBユーザのパスワードを指定します。暗号化して保存するときは、{algorithm}ciphertextの形式で記述します。

Support Xa Emulation

☐

[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEU56までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。

詳細設定

すべてを閉く

Description

Login Timeout

s

Isolation Level

Auto Commit

Stmt Query Timeout

ms

Pool Destroy Timeout

ms

Property

driverType:java.lang.String=thin

name:type=value

Action On Connection Leak

確認

再設定

その他の付加設定は画面下部の**詳細設定**で確認できます。

[図 6.8] コネクション・プール設定画面(3)

Connection Pool

HISTORY

データソース別にコネクションプール情報を設定します。

ヘルプ

BasicConnection Pool

動的設定必須項目

確認再設定

■ Pooling

JDBCコネクションプールのサイズおよび調整に関する設定です。

| | |
|--------|---|
| Min | <input type="text"/> [デフォルト:2] プーリングされるオブジェクトの最小値を設定します。 |
| Max | <input type="text"/> [デフォルト:30] プーリングされるオブジェクトの最大値を設定します。 |
| Step | <input type="text"/> [デフォルト:1] プールにコネクションが不足する場合、現在作成されたコネクションが最大値以下の場合は新規作成します。このとき新規作成するコネクション数を設定します。 |
| Period | <input type="text"/> ms [デフォルト:3600000] 指定した周期になったらコネクションプールのサイズを最小値に合わせて調整します。コネクションプールのサイズが最小値を超過する場合は使用しないコネクションをクローズし、コネクションプールのサイズが最小値に満たない場合にはコネクションを新たに確立します。ミリ秒単位で設定します。 |

■ Wait Free Connection

コネクションプール内のすべてのコネクションが占有されているとき、コネクション要求をハンドリングするメソッドを定義します。

| | |
|-------------|--|
| Enable Wait | <input type="checkbox"/> [デフォルト:false] コネクションプールに使用可能なコネクションが存在せず、それ以上コネクションを増やすこともできないとき、コネクション要求を処理する方法を指定します。trueに設定すると、使用可能なコネクションを取得するための待機します。falseに設定すると、新しいコネクションを作成して提供しますが、そのコネクションが返されたときはプーリングされず捨てられます。これを使い捨てのコネクションともいいます。 |
| Wait Time | <input type="text"/> ms [デフォルト:10000] コネクションを取得するために待機する時間です。設定時間が過ぎると、システムはタイムアウト例外を返します。Enable Waitがtrueの場合のみ有効です。ミリ秒単位で設定します。 |

詳細設定

すべてを開く

■ Connection Pool

| | |
|-----------------------------|--------------------------|
| Delegation Datasource | <input type="text"/> |
| Max Use Count | <input type="text"/> |
| Delegation Db | <input type="text"/> |
| Db | <input type="text"/> |
| Db Timeout | <input type="text"/> ms |
| Stmt Caching Size | <input type="text"/> |
| Stmt Fetch Size | <input type="text"/> |
| Use Sql Trace | <input type="checkbox"/> |
| Keep Connection Handle Open | <input type="checkbox"/> |
| Init Sql | <input type="text"/> |

▼ Connection Trace

| | |
|----------------------|-------------------------------------|
| Enabled | <input type="checkbox"/> |
| Get Connection Trace | <input checked="" type="checkbox"/> |
| Auto Commit Trace | <input type="checkbox"/> |

確認再設定

140 JEUS サーバガイド

以下は、各設定項目についての説明です。

● Pooling

JDBCコネクション・プールのサイズとその調整に関する設定を定義します。

| 項目 | 説明 |
|--------|--|
| Min | コネクション・プールにプーリングされるコネクションの最小値を指定します |
| Max | コネクション・プールにプーリングされるコネクションの最大値を指定します |
| Step | コネクション・プールにコネクションが不足する場合、コネクション数が最大値以下であれば、データベースからコネクションを取得しますが、そのときに取得するコネクションの数を指定します |
| Period | コネクション・プールのサイズを最小値に合わせて調整する周期を設定します。 コネクション・プールのサイズが最小値を超える場合は、使用しないコネクションを閉じ、またコネクション・プールのサイズが最小値に及ばない場合は、データベースからコネクションを新たに取得します(単位 : ms) |

● Wait Free Connection

コネクション・プールにあるすべてのコネクションが占有されている場合に、コネクション要求を処理するメソッドを定義します。

| 項目 | 説明 |
|-------------|---|
| Enable Wait | コネクション・プールに使用できるコネクションがなく、コネクションを増やすこともできない場合に、コネクション要求を処理する方法を決定します – true : 使用可能なコネクションを取得するために待ちます – false : 新しいコネクションを作成して提供します。ただし、このコネクションは返された後にプーリングされずに破棄されます。このようなコネクションを使い捨て(disposable)コネクションとも言います |
| Wait Time | 「Enable Wait」がtrueの場合にのみ有効な設定です。コネクションを取得するために待つ限界時間を示します。この時間が過ぎてもコネクションを取得できない場合、JEUSはタイムアウト例外を発生させます(単位 : ms) |

● Connection Validation

アプリケーションがコネクションを要求したとき、コネクションをアプリケーションに渡す前に特定のクエリーを実行してコネクションの状態をチェック(validation)する機能です。JDBCコネクション内部エラーによる切断、ファイアウォールによるソケット切断などをチェックするときに有効です。

コネクションの状態に異常がある場合、データベースからコネクションを新たに取得してアプリケーションに渡します。RACのためのクラスター・データソースに属するデータソースなら、必ずこの設定を行う必要があります。

| 項目 | 説明 |
|-------------------------|--|
| Check Query | <p>コネクション状態のチェックに使用されるクエリーを設定します。</p> <p>通常、データベースとの接続の有効性のみを確認すればいいので、簡単なselectクエリーを使用することを推奨します。</p> <p>チェック・クエリーを実行する代わりに、JDK 6でjava.sql.Connectionに追加されたisValidメソッドを利用できます。クエリー文の代わりに「use isValid method」を記します。</p> <p>それ以外のConnection Validation関連の設定(Check Query Timeout、Check Query Periodなど)は、isValidメソッドを使用する際にも同様に適用されます</p> |
| Check Query Timeout | <p>コネクションをチェックするためにチェック・クエリーを実行したとき、データベースから応答がなく、ドライバが待ち続ける状況が発生することがあります。このようなことを避けるために、チェック・クエリーに対してクエリー・タイムアウトを適用します。(単位: ms)</p> <p>これはJDBC APIで定義したjava.sql.Statement#setQueryTimeoutメソッドを呼び出すことで可能です。</p> <p>1000msより小さい場合、0に設定されるので注意してください</p> |
| Non Validation Interval | <p>コネクションのチェックが頻繁に起こりすぎて、オーバーヘッドが発生する場合に設定します。(単位: ms)</p> <p>コネクション・チェックを実行する直前の時刻と最後にコネクション・チェックを実行した時刻との差が設定した時間間隔内である場合、コネクション・チェックを省略する設定です。</p> <p>たとえば、この設定値が5000msの場合、あるコネクションの最後のコネクション・チェック時間からまだ5秒が過ぎていなければ、そのコネクションに対するチェックが省略されたままアプリケーションに渡されます</p> |
| Check Query Period | <p>コネクション・プールのコネクションを設定した周期でチェックし、問題のあるコネクションを削除します。クラスター・データソースに属するデータソースは自身の状態チェックに使用するため、必ず設定する必要があります(単位 : ms)</p> |
| Check Query Class | <p>コネクション・チェック機能をカスタマイズしたい場合、そのために実装したクラスのパッケージ名を含む名前を記述します。</p> <p>当該クラスは必ずjeus.jdbc.connectionpool.JEUSConnectionCheckerインターフェースを実装する必要があります</p> |

| 項目 | 説明 |
|-------------------------------|---|
| Check Query Retrial Count | <p>コネクションのチェックは、基本的にDestroy Policy On Check QueryがFAILED_CONNECTION_ONLYに設定されている場合、一度実行されます。</p> <p>Destroy Policy On Check QueryがALL_CONNECTIONSに設定されている場合は、最初のコネクション・チェックでコネクションの異常が確認されると、また他のコネクションに対してもう一度コネクションのチェックが行われ、合計2回のコネクション・チェックが実行されることがあります。このようなコネクション・チェックの基本実行回数にこの設定値が加えられ、最終コネクション・チェック実行回数が決まります</p> |
| Destroy Policy On Check Query | <p>コネクションが有効でないと確認されたとき、コネクション・プールにある他のコネクションの処理ポリシーを設定します</p> <ul style="list-style-type: none"> – FAILED_CONNECTION_ONLY : 有効でないと確認されたコネクションのみ削除します – ALL_CONNTECTIONS : 有効でないと確認されたコネクションを削除し、コネクション・プールにある他のコネクションの有効性をもう一度確認します。そのコネクションも有効でないと確認されたら、コネクション・プールのすべてのコネクションを削除します |

Check Query Class設定で説明したjeus.jdbc.connectionpool.JEUSConnectionCheckerインターフェースの詳細は以下のとおりです。カスタマイズされたcheck-query機能を使用したい場合は、このインターフェースを実装したクラスをチェック・クエリー・クラスとして設定します。

[例 6.1] jeus.jdbc.connectionpool.JEUSConnectionChecker

```
public interface JEUSConnectionChecker {
    void checkConnection(Connection vcon) throws SQLException;
    void setQueryString(String query);
    void setQueryTimeout(int timeout);
}
```

| メソッド | 説明 |
|-------------------|--|
| checkConnection() | 実際にコネクション・チェックを実行時に呼び出されます。このメソッドにコネクションをチェック時に実行したい作業を実装します |
| setQueryString() | チェック・クエリーが設定された場合、設定値を引数として呼び出します。設定したクエリーがコネクションのチェックに使用されます |
| setQueryTimeout() | Check Query Timeoutが設定された場合、設定値を引数として呼び出します。設定したクエリー・タイムアウトがコネクションのチェックに適用されます |

● Connection Pool

その他のコネクション・プールの付加機能を設定します。

| 項目 | 説明 |
|--------------------------|---|
| Delegation Datasource | <p>トランザクションと連携されていない状態では、XAデータソースを介してコネクションを取得するよりは、コネクション・プール・データソースを利用してコネクションを取得する方が望ましいです。</p> <p>機能上の違いはありませんが、トランザクション連携機能が含まれているXAコネクションは、システムにより多くの負担を与えます。そのため、XAデータソースの場合は、この設定により、トランザクションと連携されていない状態でコネクションの要求を委任するコネクション・プール・データソースを指定します。</p> <p>一方、Oracle、DB2などで、XAコネクションをトランザクションなしで使用したり、またはトランザクションに連携して使用したりすると、XAが開始できない例外が発生することがありますが、これを避けるためにもこの設定を利用します</p> |
| Max Use Count | <p>コネクションの最大の使用回数です。指定した使用回数を超えると、新しいコネクションに切り替えます(デフォルト値: 0。コネクションは切り替えないことを意味する)</p> |
| Delegation Db | <p>データベースのセッションを強制的に切断できる権限(DBA権限)を持つデータソース(以下、DBA委譲データソース)のJNDI名を設定します。この設定を行ったデータソースから取得したコネクションを利用したクエリー実行が一定時間以上遅延されると、JEUSは委譲DBAデータソースによりそのコネクションと関連付けられたDBセッションを強制的に削除するようにデータベースにクエリーを送信します。</p> <p>以降、使用できなくなったコネクションのため発生した例外をアプリケーションが処理しコネクションを閉じると、JEUSはそのコネクションを削除し、データベースから新しいコネクションを取得してコネクション・プールに入れます。現在、Tibero、Oracle、Sybaseに対してこの機能をサポートします。</p> <p>この機能は、JDBC 2.0以下のJDBCドライバーでクエリー実行が長引く場合、実行を中断する方法として設けられたものです。しかし、JDBC 3.0以上のバージョンを実装したJDBCドライバーではjava.sql.Statement#setQueryTimeoutを実装するため、この機能により強制的にDBセッションを削除するより、Stmt Query Timeout設定を使用することを推奨します。</p> <p>特にXAデータソースの場合は、XAが正常に実行されている途中にDBセッションが削除されると、XA処理に問題が発生することがあるので、Stmt Query Timeout設定とサーバーのトランザクション・タイムアウト設定を適切に使用するようになります</p> |

| 項目 | 説明 |
|-----------------------------|--|
| Dbc Timeout | <p>ここで設定した時間の間、DBA委譲データソースはコネクションのクエリーの実行を待ちます。設定した時間が過ぎると、コネクションと関連付けられたDBセッションを強制的に削除するようにするクエリーをデータベースに送信します。(単位: ms)</p> <p>Delegation DBAが設定された場合にのみ有効です</p> |
| Stmt Caching Size | JDBCドライバーはアプリケーションがPreparedStatementを要求するたびに、パラメータとして渡されたSQL文をパースします。このパース作業が性能に影響を与えることがあるため、それを回避するためにJEUS内部でPreparedStatementをキャッシュする機能を提供します。この設定はキャッシュするPreparedStatementの数を設定します |
| Stmt Fetch Size | JDBCドライバー・ステートメントのフェッチ・サイズを設定します |
| Use Sql Trace | <p>コネクション別に使用しているSQLクエリーを表示する機能です。jeus.jdbc.sqlロガーのレベルをFINEに設定する場合、サーバー・ログによりSQLクエリーの履歴を確認できます。</p> <p>この機能を使用する場合、JDBCドライバーのステートメント実装をJEUSのステートメント実装で囲むため、JDBCドライバーのステートメント・オブジェクトをキャストして使用するアプリケーションはこの機能を使用できません</p> |
| Keep Connection Handle Open | <p>コネクションをプーリングする間、コネクション・ハンドル(または論理コネクション)を常にオープンして使用する場合に設定します。</p> <p>[参考]</p> <p>1. IBM DB2で提供するUniversal Driver(JCC) type 4のXAデータソースを使用する場合、この機能を使用することを推奨します。複数のスレッドが異なる物理コネクションのコネクション・ハンドルを開閉して使用するときにハングアップされる問題が発生するためです。</p> <p>スレッドは内部的に同じベクトル(java.util.Vector)を共有して使用していて、1つのスレッドがベクトルにロックを掛けた状態で無限ループに陥ってしまい、他のスレッドはそのロックを待つのが表面的な原因です。DB2の内部ロジックが分からないため、より根本的な原因は確認できませんが、テスト結果、コネクション・ハンドルを常に開いていれば、問題が発生しないことを確認しました。コネクション・ハンドルを常に開いておけば、コネクション・ハンドルを最初に作成するときと物理コネクションを閉じるとき以外は、内部共有ベクトルにアクセスすることがないためです。</p> <p>2. DB2ドライバーの3.53.95から上記のバグが解決されました。関連するIBMの公式バグ・レポート番号はAPAR IZ41181であり、DB2 9.5 Fixpak 4文書で確認できます。</p> |

| 項目 | 説明 |
|----------|--|
| | [注意] この機能を使用すると、コネクション・ハンドルが閉じられないため、コネクションを閉じるときにドライバーが行うクリア作業が行われません。たとえば、Oracle JDBC ドライバーの場合、Auto Commitをfalseにして使用して、コミットまたはロールバックをせずにコネクションを閉じると、無条件にコミットするようになっていますが、このような処理が行われません |
| Init Sql | コネクションを作成した後、一番最初に行うSQLクエリーを設定します |

● Connection Trace

コネクション関連の付加情報を提供するかどうかを決定します。

| 項目 | 説明 |
|----------------------|--|
| Enabled | コネクション関連の付加情報を提供するかどうかを決定します。falseの場合、「 Get Connection Trace 」と「 Auto Commit Trace 」設定は無効になります |
| Get Connection Trace | アプリケーションがjava.sql.DataSource#getConnectionを呼び出したときのスタック・トレースを確認できるようにします |
| Auto Commit Trace | java.sql.Connection#setAutoCommitを呼び出したときの関連ログとスタック・トレースをサーバー・ログに記録します。ただし、jeus.jdbc.connection-traceロガーのログ・レベルをFINEに設定する必要があります |

6.5. クラスター・データソースの設定

本章では、クラスター・データソースの設定について説明します。

コンソール・ツールまたはWebAdminを使って設定作業を行うことができます。ここでは、WebAdminを使用した設定方法を説明します。コンソール・ツールを使用した設定方法については、『*JEUS リファレンスガイド*』の「4.2.11.6. add-cluster-data-source」を参照してください。

6.5.1. クラスター・データソースの設定

WebAdminの左のメニューで[Resources] > [Data Source]を選択し、[LOCK & EDIT]ボタンをクリックしてロックを取得します。画面のCluster Dsリストで[Add]ボタンをクリックすると、クラスター・データソースの基本設定画面に移動します。

[図 6.9] クラスター・データソースの設定画面(1)

DataSource HISTORY

アプリケーションで利用できるデータソースを定義します。 ヘルプ

Database

| DataSource ID | DataSource Class Name | DataSource Type | Command | Add |
|---------------|---|-----------------------------|---------|------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolID ataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete Duplicate |

Cluster DS

| DataSource ID | Export Name | Add |
|----------------|-------------|-----|
| 該当する内容が存在しません。 | | |

その他の付加設定は、画面下部の**詳細設定**で確認できます。

[図 6.10] クラスター・データソースの設定画面(2)

Cluster DS

HISTORY

ヘルプ ?

アプリケーションサーバレベルでRACのフェイルオーバーおよびフェイルバック機能を提供するためにクラスターデータソースを使用します。これは、1つの独立的なJNDI名を持つデータソースインスタンスです。このインスタンスはアプリケーションの呼び出しを受け、クラスタリングされたデータソースの1つに渡す役割をします。フェイルオーバー、フェイルバック機能を使用する場合、プライマリデータソースがダウンすると、セカンダリデータソースを選択してアプリケーションの要求を処理します。

動的設定 * 必須項目

確認 再設定

| | | | |
|--------------------------|---|--|--|
| Data Source Id * | <div></div> <p>クラスターデータソースのIDを設定します。1つのドメインにおいて、各クラスターデータソースのIDは一意である必要があります。</p> | | |
| Export Name | <div></div> <p>クラスターデータソースのJNDI名を指定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、クラスターデータソースIDをJNDI名として使用します。</p> | | |
| Data Source Selector | <div></div> EX foo.bar.MyDataSourceSelector クラスターデータソースからコネクションを取得するとき、ユーザまたは開発者が特定のコンポーネントデータソースの指定に関するポリシーを直接定義することができます。jeus.jdbc.helper.DataSourceSelector抽象クラスを継承して実装し、その実装クラスのパッケージ名を含む名前を記述します。このオプションを設定すると、負荷分散設定は無意味になります。ポリシーを定義するときは、同期化を考慮する必要があります。 | | |
| Load Balance | <input type="checkbox"/> [デフォルト: false] 負荷分散を使用するか否かを指定します。trueに設定すると、Is Pre Conn設定とUse Failback設定は動作しません。 | | |
| Component Data Sources * | <div>ds1</div> <div></div> <p>クラスターデータソースに属するコンポーネントデータソースのデータソースIDを設定します。</p> | | |

詳細設定

すべてを開く

| | |
|--------------|-------------------------------------|
| Is Pre Conn | <input type="checkbox"/> |
| Use Failback | <input checked="" type="checkbox"/> |
| Xa Affinity | <input checked="" type="checkbox"/> |
| Ons Support | <input type="checkbox"/> |

確認 再設定

以下は、各設定項目についての説明です。

- 基本情報

| 項目 | 説明 |
|------------------------|---|
| Data Source Id | クラスター・データソースのIDです。1つのドメインでクラスター・データソースIDはクラスター・データソースの唯一な識別子として動作するように設定する必要があります |
| Export Name | クラスター・データソースのJNDI名です。 異なる2つのデータソースが別々のサーバーにJNDIバインドされることを保証できる場合、それらのデータソースは同じJNDI名を持つことができます。これは、任意のサーバーで同じJNDI名を持つ異なるデータソースを許容しないことを意味します。設定しない場合、クラスター・データソースIDをJNDI名として使用します |
| Data Source Selector | クラスター・データソースからコネクションを取得するときに、ユーザーや開発者が特定のコンポーネント・データソースの選択ポリシーを直接定義できます。jeus.jdbc.helper.DataSourceSelectorインターフェースを実装し、その実装クラスのパッケージ名を含む名前を記述します。これを設定すると、Load Balance設定は意味を持ちません。ポリシーを定義するときは、同期化を考慮する必要があります。これは、実装者の責任のもとで行われる必要があります |
| Load Balance | ロード・バランシングの設定を行います。この設定値がtrueの場合、「Is Pre Conn」、「Use Failback」設定は両方とも適用されません |
| Component Data Sources | クラスター・データソースが属するコンポーネント・データソースのデータソースIDを指定します。指定した順序で主データソースの役割を担います |

- 詳細設定

| 項目 | 説明 |
|--------------|--|
| Is Pre Conn | クラスター・データソースに属するコンポーネント・データソースのコネクション・プールをあらかじめ作成するかどうかを決定します。コンポーネント・データソースのコネクション・プールをあらかじめ作成しておくと、性能上メリットがありますが、リソースの節約という面では望ましくありません |
| Use Failback | 以前バージョンのJEUSではフェイルオーバーだけサポートしていたため、その互換性のために提供するオプションです。補助データソースにフェイルオーバーしてから、主データソースにフェイルバックするかどうかを設定します。基本的にフェイルバックを試みます。フェイルバックするためには、必ず主データソースに対して「Check Query」および「Check Query Period」を設定する必要があります |
| Xa Affinity | グローバル・トランザクションのアフィニティを設定します。この設定を有効にすると、グローバル・トランザクション処理が1つのRACインスタンスを対象に行われ、グローバル・トランザクション処理の性能が向上されます |

| 項目 | 説明 |
|-------------|--|
| Ons Support | <p>ONSを用いたクラスター・データソースを設定します。このクラスター・データソースは、ONSを利用してコンポーネント・データソースの失敗や復旧の可否を効率よく検知することができます。特に、ロードバランシング方式の場合は、ランタイム・ロードバランシング・アドバイザーを利用して、より効率的なロードバランシングを提供します。</p> <p>以下のような下位設定があります</p> <ul style="list-style-type: none"> – Ons Config <p>ONS上の各RACノードがONS通信に使用するIPとポート番号を設定します。クラスター・データソースは、設定されたIPとポート番号にソケット接続を確立し、ONSクライアントとして動作することになります。nodes=host1:6200,host2:6200のような形式で入力します</p> |

Data Source Selector設定で触れたjeus.jdbc.helper.DataSourceSelectorインターフェースの詳細は以下のとおりです。コンポーネント・データソースの選択ポリシーを直接定義するには、このインターフェースを実装したクラスをデータソース・セレクターとして設定します。

[例 6.2] jeus.jdbc.helper.DataSourceSelector

```
public interface DataSourceSelector {
    public void setComponentDataSourceList(List<String> componentDataSourceList);
    public String selectDataSource();
}
```

| メソッド | 説明 |
|------------------------------|--|
| setComponentDataSourceList() | クラスター・データソースに属するコンポーネント・データソースのIDリストを設定します |
| selectDataSource() | クラスター・データソースに属するコンポーネント・データソースの選択ポリシーを定義するために実装するメソッドです。リターン値は定義されたポリシーにより選択されたデータソースのIDになります。ほとんどの環境でクラスター・データソースからコネクションを取得する場合は多数の要求スレッドが同時にアクセスするはずなので、ポリシーを定義するときに同期化を考慮する必要があります。これは、実装者の責任のもとで行われる必要があります |

上記の詳細をもとに、以下のようなカスタマイズされたDataSourceSelectorの実装が可能です。このDataSourceSelector実装は、クラスター・データソースに参加する2つのデータソースに対して2:1の選択比率を定義しています。

[例 6.3] DataSourceSelectorの実装例

```
package foo.bar;

import jeus.jdbc.helper.DataSourceSelector

public class MyDataSourceSelector implements DataSourceSelector {
    List<String> componentDataSourceList = new ArrayList<String>();

    // 同期化の保証
    AtomicInteger dataSourceIndex = new AtomicInteger(0);

    public void setComponentDataSourceList(List<String> componentDataSourceList)
    {
        this.componentDataSourceList.addAll(componentDataSourceList);
    }

    public String selectDataSource() {
        int reminder = (dataSourceIndex.getAndIncrement() & 0x7fffffff) % 3;
        if(reminder < 2) {
            return componentDataSourceList.get(0);
        }
        else {
            return componentDataSourceList.get(1);
        }
    }
}
```

6.5.2. クラスター・データソースに属するコンポーネント・データソースの設定

クラスター・データソースは自身に属するコンポーネント・データソースにコネクション要求処理を委任する方式で動作します。Load Balance項目が設定されていない場合、クラスター・データソースはコネクションの要求処理を常に主コンポーネント・データソースに委任します。しかし、主コンポーネント・データソースに異常があると判断した場合は、新しいコンポーネント・データソースを主コンポーネント・データソースにして、コネクション要求処理に問題がないようにします。これがクラスター・データソースのフェイルオーバー機能です。

フェイルオーバーした後にフェイルバックするためには、最初的主コンポーネント・データソースが復旧されたことを確認する必要があります。そのコンポーネント・データソースに周期的にチェック・クエリーを送信することで確認できます。そのため、クラスター・データソースでフェイルバック機能を利用するためには、クラスター・データソースに属するすべてのコンポーネント・データソースに対して「**Check Query**」および「**Check Query Period**」を設定する必要があります。もしフェイルオーバーだけを使用したい場合は、「**Use Failback**」をfalseに設定します。手動でフェイルオーバー・コマンドを実行することも可能ですが、これについては、『*JEUS リファレンスガイド*』の「4.2.12.8. control-cluster-data-source」を参照してください。

参考

コンポーネント・データソースのDestroy Policy On Check Query項目はALL_CONNECTIONSに設定することを推奨します。そうしない場合、フェイルオーバーした後も異常が検知されたコンポーネント・データソースの接続・プールに使用できない接続がしばらくの間残っているためです。もちろん、これらの接続は周期的な接続のチェックによって整理されますが、フェイルオーバーが行われる時に削除されることが望ましいです。

6.6. データソース関連設定の動的変更

本節では、例を介してデータソース関連設定の動的な変更方法について説明します。理解を助けるために、JEUSのドメイン構造やドメインでのデータソース管理構造とデータソースの各設定項目について熟知する必要があります。JEUSのドメイン構造については、『*JEUSドメインガイド*』の「1.3. 構成要素」を参照してください。ドメインでのデータソース管理構造とデータソースの各設定項目の使用については、それぞれ「[6.3. データソースと接続・プールの管理](#)」と「[6.4. データソース設定](#)」を参照してください。

動的な変更はコンソール・ツールまたはWebAdminを使って実行することができ、本節では両方の方法を説明します。コンソール・ツールを使用した方法を説明するときは、便宜上縮約されたコマンドおよびオプションの名前を使用します。縮約コマンドおよびオプション名とコマンドの詳しい使用方法については、コンソール・ツールでコマンド名を引数としてhelpを実行するか、『*JEUS リファレンスガイド*』の「4.2.11. データソース関連コマンド」および『*JEUS リファレンスガイド*』の「4.2.12. 接続・プールのモニタリングおよび制御コマンド」を参照して確認できます。

説明を行うために必要な次の事前作業を行います。まず、domain1という名前のドメインとadminServerという名前のDASを作成します。DASを起動した後、それぞれserver1、server2、server3という名前を持つ3つのMSをdomain1に追加し、これらも起動します。そして、その中のserver2とserver3をcluster1という名前のクラスターで構成します。ドメイン、クラスター、サーバーの設定方法については『*JEUSドメインガイド*』の「第2章 ドメインの作成」、『*JEUSドメインガイド*』の「第5章 JEUSクラスタリング」、および「[第2章 JEUSの設定](#)」を参照してください。以降の説明では、上述した事前作業がすべて完了されていることを仮定します。

参考

それぞれの例をシナリオを構成して説明するので、本節の内容は、記述された順に従って読み進んでください。

6.6.1. データソースの追加

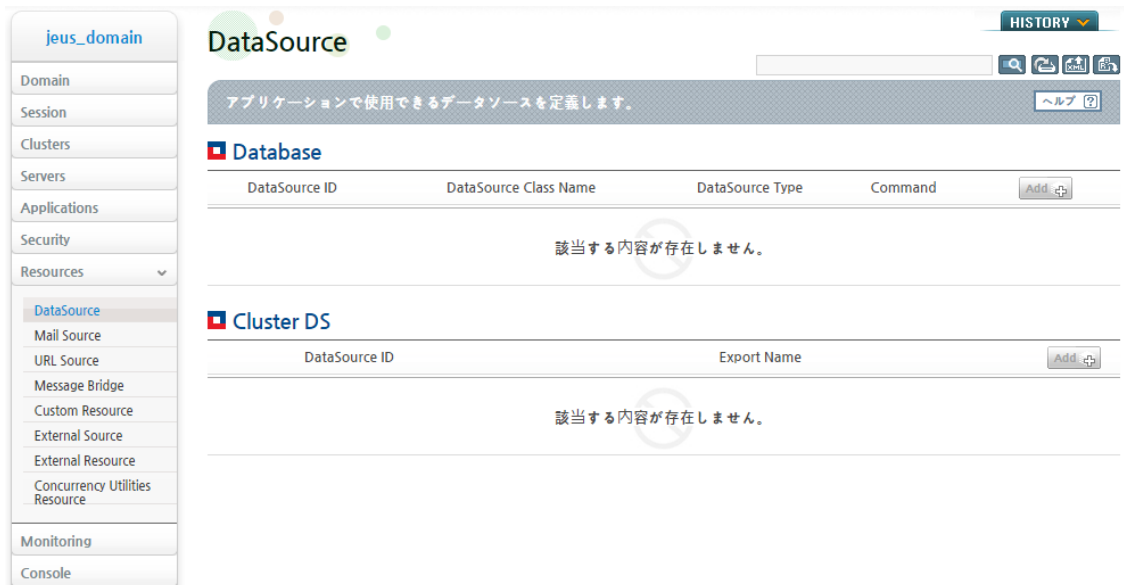
データソースの動的追加が可能です。データソースの追加は、ドメインにデータソースの設定を登録してドメインに属するサーバーとクラスターによって参照されることができる状態にする作業です。

WebAdminの使用

以下では、WebAdminを使ってデータソースを追加する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [Data Source]を選択して[LOCK & EDIT]ボタンをクリックした後、画面に表示されるDatabaseリストで[Add]ボタンをクリックすると、データソースをドメインに動的に追加できる画面に移動します。

[図 6.11] ドメインへのデータソースの追加(1)



2. 以下は、必要な設定を入力して、データソースIDがds1のデータソースをドメインに追加する例です。設定を入力した後、[確認]ボタンをクリックします。

[図 6.12] ドメインへのデータソースの追加(2)

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|--|
| Data Source Id * | ds1 データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。 |
| Export Name | データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。 |
| Vendor | oracle 入力 JDBCドライバベンダの名前を指定します。 |
| Data Source Class Name * | oracle.jdbc.pool,OracleConnectionPoolDataSource oracle.jdbc.pool,OracleConnectionPoolDataSource JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。 |
| Data Source Type * | ConnectionPoolDataSource データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプールリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプールリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプールリングサービスと共にXA運動がサポートされます。 |
| Server Name | 192.168.2.49 DBが実行されるホスト名、またはIPを設定します。 |
| Port Number | 1521 DBリスナーのポート番号を設定します。 |
| Database Name | ora10g DBの名前を指定します。Oracleの場合、DBのSIDを設定します。 |
| User | jeusdb2 DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。 |
| Password | 入力 {DES}FQrLbQ/D8O1dVS71L28rw== DBユーザのパスワードを指定します。暗号化して保存するときは、r{algorithm}ciphertextLの形式で記述します。 |
| Support Xa Emulation | <input type="checkbox"/> [デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。 |

詳細設定 すべてを開く

| | |
|---------------------------|--|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java.lang.String=thin</div> <div>name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

3. ds1をドメインに追加することを最終反映するために[**Activate Changes**]ボタンをクリックします。
4. 以下のように、ds1がドメインに追加されたことが確認できます。以降の説明のために、さらにデータソースIDがそれぞれds2、ds3のデータソースを上述した方法で追加します。以降の説明では、ds2、ds3がdomain1に追加されていることを仮定します。

[図 6.13] ドメインへのデータソースの追加(3)

jeus8
Domain
Session
Clusters
Servers
Applications
Security
Resources
DataSource
Mail Source
URL Source
Message Bridge
Custom Resource
External Source
External Resource
Concurrency Utilities
Resource
Monitoring
Console
システム状態
0 Failed
0 Standby
1 Running
1 Shutdown
0 Suspended
0 Other

DataSource

アプリケーションで使用するデータソースを定義します。

```

domain.xmlの設定を一部自動的に適用しました。
domain.xml : PARTIALLY_ACTIVATED
resources.dataSource.database : ACTIVATED
resources.dataSource.database.ds1 : ACTIVATED
resources.dataSource.database.ds1 : ACTIVATED - adminServer
previous value : null, edited value : jeus.xml.binding.jeusDO.DatabaseType@1307988, result value :
jeus.xml.binding.jeusDO.DatabaseType@1307988
resources.customResource : PENDING
resources.externalResource : PENDING
変更内容を適用するには、サーバを再起動してください。

```

Database

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete Duplicate |

Cluster DS

| DataSource ID | Export Name |
|----------------|-------------|
| 該当する内容が存在しません。 | |

コンソール・ツールの使用

コンソール・ツールで**add-data-source**を実行すると、データソースを動的にドメインに追加できます。add-data-sourceを実行してデータソースを追加するには、データソースの各詳細設定のオプション値を設定する必要があります。add-data-sourceのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.11.1. add-data-source」を参照してください。

以下は、add-data-sourceを実行して必要な設定を入力し、データソースIDがds1のデータソースをdomain1に追加する例です。

```
[DAS]domain1.adminServer>addds -id ds1 -dst ConnectionPoolDataSource -dscn
oracle.jdbc.pool.OracleConnectionPoolDataSource -sn 192.168.1.165 -pn 1521 -dn ora10g
-user jeustest1 -password jeustest1 -property driverType;java.lang.String;thin
Successfully performed the ADD operation for data source [ds1] to domain.
Check the results using "add-data-source"
```

上記のメッセージのように、ds1が追加されたことを、**add-data-source**を実行して確認できます。

```
[DAS]domain1.adminServer>addds
Shows the current configuration
Data sources in domain
=====
+-----+-----+-----+-----+
| ds1   | common data source |
+-----+-----+-----+-----+
=====
```

以降の説明のために、データソースIDがそれぞれds2、ds3のデータソースを上述した方法で追加します。以降の説明では、ds2、ds3がdomain1に追加されていることを仮定します。

6.6.2. サーバーへのデータソースの登録

ドメインにデータソースを追加しても、サーバーで実際にデータソースを参照して使用するためには、データソースをサーバーに登録する必要があります。データソースがサーバーに登録されると、データソース情報がサーバーのJNDIリポジトリにバインドされ、以降サーバーにデプロイされたアプリケーションはこれをルックアップできます。サーバーにデータソースを登録する作業は動的に処理できます。

WebAdminの使用

以下では、WebAdminを使ってサーバーにデータソースを登録する方法について説明します。

1. WebAdminの左のメニューで[**Servers**]を選択して[**LOCK & EDIT**]ボタンをクリックした後、表示される画面で任意のサーバーを選択すると、データソースを動的に登録できる画面に移動します。

[図 6.14] サーバーへのデータソースの登録(1)

Servers

ドメイン内でJEUSサーバを構成するとき、各サーバの設定を行います。

Servers

| Name | Status | Pid | Need To Restart | Command | Actions |
|-----------------|-------------------|------|-----------------|------------|------------------|
| adminServer (*) | RUNNING(00:02:36) | 8867 | false | Start Stop | Delete Duplicate |
| server1 | RUNNING(00:00:09) | 9098 | false | Start Stop | Delete Duplicate |
| server2 | RUNNING(00:00:14) | 9106 | false | Start Stop | Delete Duplicate |
| server3 | RUNNING(00:00:13) | 9117 | false | Start Stop | Delete Duplicate |

Server Templates

該当する内容が存在しません。

2. 以下は、server1にds1、ds2を登録する例です。「**Data Source**」項目で追加された「ds1」、「ds2」のチェックボックスにチェックを入れて[**確認**]ボタンをクリックします。

[図 6.15] サーバーへのデータソースの登録(2)

Data Sources

サーバまたはクラスタで有効なデータソースを指定します。

Data Source

- ☒ ds1
- ☒ ds2
- ☐ ds3

サーバ、またはクラスタで有効なデータソースのIDを指定します。

3. ds1、ds2をserver1に登録することを最終反映するために[**Activate Changes**]ボタンをクリックします。

4. 以下のように、ds1、ds2がserver1に登録されていることが確認できます。

[図 6.16] サーバーへのデータソースの登録(3)

Server

HISTORY

ヘルプ ?

ドメイン内で使用するJEUSサーバの詳細設定を定義します。

domain.xmlの設定を動的に適用しました。

domain.xml : ACTIVATED

servers.server.[? name == 'server1'].dataSources : ACTIVATED

servers.server.[? name == 'server1'].dataSources.ds1 : ACTIVATED

previous value : null, edited value : ds1, result value : ds1

servers.server.[? name == 'server1'].dataSources.ds2 : ACTIVATED

previous value : null, edited value : ds2, result value : ds2

Basic

Resource

Engine

Basic Info

Res Ref

Naming Server

System Thread Pool

System Logging

User Logging

Tm Config

動的設定 * 必須項目

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

| | |
|----------------|--|
| Auto Generated | <input type="checkbox"/> サーバが動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスタが削除されると、該当するサーバも自動的に削除されます。 |
| Name * | server1 サーバ名です。 |
| Log Home | <div></div> JEUSサーバで生成するログのデフォルトパスを指定します。同パスが設定されていても、ロガーのファイルハンドラに設定されているパスが絶対パスの場合にはそのパスが優先されます。デフォルトログのみならず、ローテーションされたバックアップログが生成されるパスにも同様に適用されます。 |
| Node Name | michael-desktop サーバが属するノード名を指定します。ノードにはマシン情報、ホスト情報などが含まれます。 |

5. JNDIモニタリング情報を確認すると、server1のJNDIリポジトリにds1、ds2がバインドされていることが確認できます。データソースのJNDI名を別途入力していないので、それぞれのデータソースIDがJNDI名としても使用されています。

【図 6.17】 サーバーにJNDIバインドされたデータソースの確認

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| ds2 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

コンソール・ツールの使用

コンソール・ツールで**add-data-sources-to-server**を実行すると、データソースを動的にサーバーに登録できます。add-data-sources-to-serverのより詳しい使用方法は、『*JEUS リファレンスガイド*』の「4.2.3.2. add-data-sources-to-server」を参照してください。

以下は、add-data-sources-to-serverを実行して、ds1、ds2をserver1に登録する例です。

```
[DAS]domain1.adminServer>adddstosvr -server server1 -ids ds1,ds2
Successfully performed the ADD operation for data sources to the server [server1].
Check the results using "add-data-sources-to-server -server server1"
```

上記のメッセージのように、ds1、ds2がserver1に登録されたことを、「**add-data-sources-to-server -server server1**」を実行して確認できます。

```
[DAS]domain1.adminServer>adddstosvr -server server1
Shows the current configuration.
Data sources registered in the server [server1].
=====
+-----+-----+
| data sources | ds1, ds2 |
+-----+-----+
=====
```

jndilistを実行すると、server1のJNDIリポジトリにds1とds2がバインドされていることを確認できます。ds1とds2のJNDI名を別途入力していないので、それぞれのデータソースIDがJNDI名としても使用されています。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
+-----+-----+-----+
|      Name      |      Value      | Local Binding |
+-----+-----+-----+
| ds1            | jeus.jdbc.info.JDBCBindInfo | true         |
| ds2            | jeus.jdbc.info.JDBCBindInfo | true         |
| JEUSMQ_DLQ     | jeus.jms.common.destination.JeusQueue | false        |
| mgmt           | jeus.jndi.JNSContext      | false        |
+-----+-----+-----+
=====
```

6.6.3. サーバーからのデータソースの削除

サーバーに登録したデータソースは動的に削除することができます。データソースがサーバーから削除されるときは、データソース情報がJNDIアンバインドされ、コネクション・プールが作成された場合、コネクション・プールが破棄されます。

WebAdminの使用

以下では、WebAdminを使ってサーバーからデータソースを削除する方法について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択して**[LOCK & EDIT]**ボタンをクリックした後、表示される画面で任意のサーバーを選択すると、サーバーからデータソースを動的に削除できる画面に移動します。

[図 6.18] サーバーからのデータソースの削除(1)

Servers

ドメイン内でJEUSサーバを構成するとき、各サーバの設定を行います。

| Name | Status | Pid | Need To Restart | Command | Actions |
|-----------------|-------------------|------|-----------------|------------|------------------|
| adminServer (*) | RUNNING(00:02:36) | 8867 | false | Start Stop | Delete Duplicate |
| server1 | RUNNING(00:00:09) | 9098 | false | Start Stop | Delete Duplicate |
| server2 | RUNNING(00:00:14) | 9106 | false | Start Stop | Delete Duplicate |
| server3 | RUNNING(00:00:13) | 9117 | false | Start Stop | Delete Duplicate |

Server Templates

該当する内容が存在しません。

2. 以下は、server1からds2を削除する例です。「**Data Source**」項目で「ds2」のチェックボックスのチェックを解除して[確認]ボタンをクリックします。

[図 6.19] サーバーからのデータソースの削除(2)

Data Sources

サーバまたはクラスタで有効なデータソースを指定します。

| Data Source | Status |
|-------------|-------------------------------------|
| ds1 | <input checked="" type="checkbox"/> |
| ds2 | <input type="checkbox"/> |
| ds3 | <input checked="" type="checkbox"/> |

サーバ、またはクラスタで有効なデータソースのIDを指定します。

Activate Changes

3. ds2をserver1から削除することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、ds2がserver1から削除されていることが確認できます。

【図 6.20】サーバーからのデータソースの削除(3)

ドメイン内で使用するJEUSサーバの詳細設定を定義します。

```
domain.xmlの設定を自動的に適用しました。
domain.xml : ACTIVATED
servers.server.{? name == 'server1' }.dataSources : ACTIVATED
servers.server.{? name == 'server1' }.dataSources.ds2 : ACTIVATED
previous value : ds2, edited value : null, result value : null
```

Basic Resource Engine

Basic Info Res Ref Naming Server System Thread Pool System Logging User Logging Tm Config

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

| | |
|----------------|--|
| Auto Generated | <input type="checkbox"/> サーバが自動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスタが削除されると、該当するサーバも自動的に削除されます。 |
| Name * | server1 サーバ名です。 |
| Log Home | <input type="text"/> JEUSサーバで生成するログのデフォルトパスを指定します。同パスが設定されていても、ローガーのファイルハンドラに設定されているパスが絶対パスの場合にはそのパスが優先されます。デフォルトログのみならず、ローテーションされたバックアップログが生成されるパスにも同様に適用されます。 |
| Node Name | michael-desktop サーバが属するノード名を指定します。ノードにはマシン情報、ホスト情報などが含まれます。 |
| Group | <input type="text"/> サーバを管理するためのグループを設定します。WebAdminでグループ別にサーバを管理することができます。 |

5. JNDIモニタリング情報を確認すると、server1のJNDIリポジトリからds2がアンバインドされていることが確認できます。

【図 6.21】サーバーからJNDIアンバインドされたデータソースの確認

jeus_domain

Domain Session Clusters Servers Applications Security Resources Monitoring Thread Transaction MBean JNDI Web Servers JMS Connection Pool

JNDI

JNDIのモニタリング情報を照会します。

The JNDI list on the server1

adminServer

server1

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMQ_DLO | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server2

server3

コンソール・ツールの使用

コンソール・ツールで**remove-data-sources-from-server**を実行すると、データソースをサーバーから動的に削除できます。remove-data-sources-from-serverのより詳しい使用方法是『*JEUS リファレンスガイド*』の「4.2.3.34. remove-data-sources-from-server」を参照してください。

以下は、remove-data-sources-from-serverを実行してds2をserver1から削除する例です。

```
[DAS]domain1.adminServer>rmdsfrosvr -server server1 -ids ds2
Successfully performed the REMOVE operation for data sources from the server [server1].
Check the results using "remove-data-sources-from-server -server server1"
```

remove-data-sources-from-server -server server1を実行すると、server1からds2が削除され、ds1だけが残っていることが確認できます。

```
[DAS]domain1.adminServer>rmdsfrosvr -server server1
Shows the current configuration.
Data sources registered in the server [server1].
=====
+-----+-----+-----+
| data sources | ds1 |
+-----+-----+-----+
=====
```

jndilistを実行すると、server1のJNDIリポジトリからds2が削除され、ds1だけがバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+
=====
```

6.6.4. クラスターへのデータソースの登録

サーバーにデータソースを登録したように、クラスターにもデータソースを登録することができます。

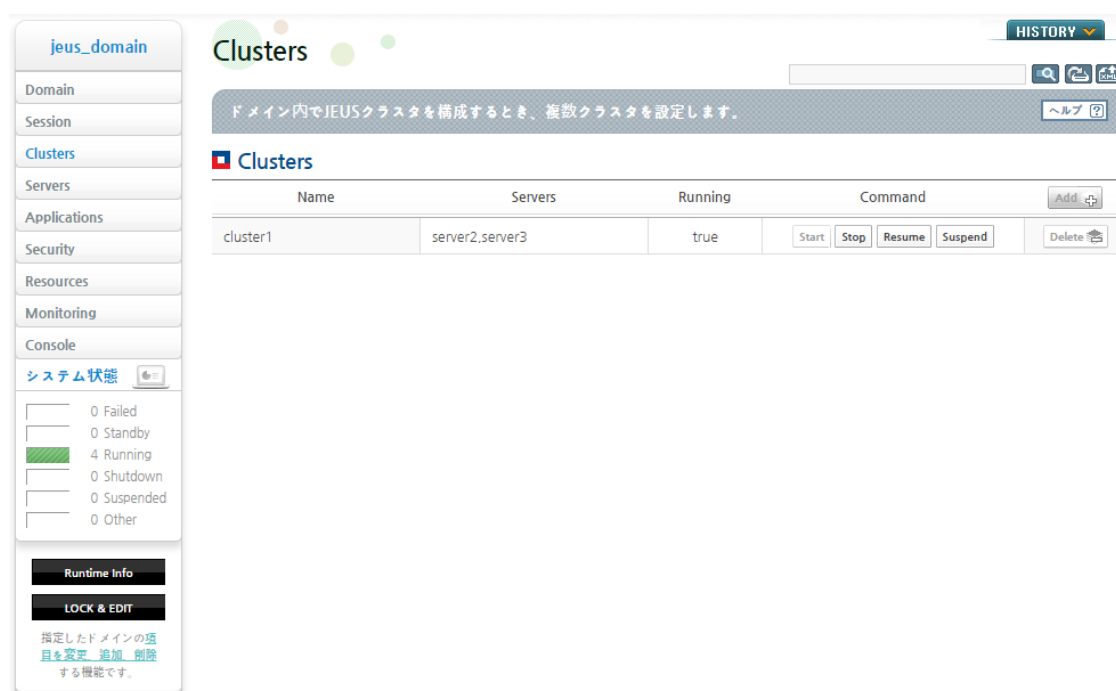
クラスターに登録されたデータソースは、クラスターに属するすべてのサーバーで有効です。つまり、クラスターに属するサーバーはクラスターに登録されたデータソースをまるで自身が登録したデータソースのように使用することができます。クラスターにもデータソースを動的に登録できます。

WebAdminの使用

以下では、WebAdminを使ってデータソースをクラスターに登録する方法について説明します。

1. WebAdminの左のメニューで[Clusters]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面で任意のクラスターを選択すると、そのクラスターにデータソースを動的に登録できる画面に移動します。

[図 6.22] クラスターへのデータソースの登録(1)



2. 以下は、cluster1にds2、ds3を登録する例です。「Data Source」項目で「ds2」、「ds3」のチェックボックスにチェックを入れて[確認]ボタンをクリックします。

[図 6.23] クラスターへのデータソースの登録(2)



3. ds2、ds3をcluster1に登録することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、ds2、ds3がcluster1に登録されていることが確認できます。

[図 6.24] クラスターへのデータソースの登録(3)

The screenshot shows the JBoss Cluster configuration interface. At the top, there's a 'Cluster' header with a 'HISTORY' button. Below it, a message states: 'クラスター構成のための詳細設定を行います。' (Perform detailed settings for cluster configuration). A log window displays the following configuration details:

```
domain.xmlの設定を動的に適用しました。
domain.xml : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - server1
previous value : null, edited value : ds2, result value : ds2
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - server2
previous value : null, edited value : ds2, result value : ds2
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - adminServer
previous value : null, edited value : ds2, result value : ds2
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - server3
previous value : null, edited value : ds2, result value : ds2
clusters.cluster.{? name == 'cluster1' }.dataSources.ds3 : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.ds3 : ACTIVATED - server1
previous value : null, edited value : ds3, result value : ds3
clusters.cluster.{? name == 'cluster1' }.dataSources.ds3 : ACTIVATED - server2
previous value : null, edited value : ds3, result value : ds3
clusters.cluster.{? name == 'cluster1' }.dataSources.ds3 : ACTIVATED - adminServer
previous value : null, edited value : ds3, result value : ds3
clusters.cluster.{? name == 'cluster1' }.dataSources.ds3 : ACTIVATED - server3
previous value : null, edited value : ds3, result value : ds3
```

Below the log window, there are tabs for 'Basic', 'Res Ref', 'Session Cluster Config', 'Jms Resource', and 'Lifecycle Invocation'. The 'Basic' tab is selected. It shows the 'Name' field set to 'cluster1' with a note: 'このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。' (If you want to change the settings on this page, click the [Lock & Edit] button from the left menu). Below this, the 'Servers' section is expanded, showing a list of servers to be added to the cluster:

| Server Name | Selected |
|-------------|-------------------------------------|
| adminServer | <input type="checkbox"/> |
| server1 | <input type="checkbox"/> |
| server2 | <input checked="" type="checkbox"/> |
| server3 | <input checked="" type="checkbox"/> |

A note at the bottom states: 'クラスターに参加するサーバの名前を指定します。' (Specify the name of the server to join the cluster).

5. cluster1にds2とds3が登録されたので、cluster1に属するserver2とserver3は、ds2とds3を自身が登録したデータソースのように使用することができます。これは、server2とserver3のそれぞれのJNDIリポトリにds2とds3がJNDIバインドされたことを意味します。server2とserver3のJNDIモニタリング情報を確認すると、ds2とds3がJNDIバインドされていることが確認できます。

[図 6.25] サーバーにJNDIバインドされたデータソースの確認(1)

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| ds3 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

[図 6.26] サーバーにJNDIバインドされたデータソースの確認(2)

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| ds3 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

コンソール・ツールの使用

コンソール・ツールで**add-data-sources-to-cluster**を実行すると、データソースを動的にクラスターに登録できます。add-data-sources-to-clusterのより詳しい使用方法は、『*JEUS リファレンスガイド*』の「4.2.4.4. add-data-sources-to-cluster」を参照してください。

以下は、add-data-sources-to-clusterを実行してds2、ds3をcluster1に登録する例です。

```
[DAS]domain1.adminServer>addstocluster -cluster cluster1 -ids ds2,ds3
Successfully performed the ADD operation for data sources to the cluster [cluster1].
Check the results using "add-data-sources-to-cluster -cluster cluster1"
```

上記のメッセージのように、ds2、ds3がcluster1に登録されたことを、「**add-data-sources-to-cluster -cluster cluster1**」を実行して確認できます。

```
[DAS]domain1.adminServer>addstocluster -cluster cluster1
Shows the current configuration.
The data sources registered in the cluster [cluster1].
=====
+-----+-----+
| data sources | ds2, ds3 |
+-----+-----+
=====
```

cluster1にデータソースds2とds3が登録されたので、cluster1に属するserver2とserver3は、ds2とds3を自身が登録したデータソースのように使用することができます。これは、server2とserver3のそれぞれのJNDIリポジトリにds2とds3がJNDIバインドされたことを意味します。**jndilist**を実行してserver2のJNDIリポジトリを確認すると、ds2とds3がJNDIバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server2
The JNDI list on the server2
List of the context /
=====
+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| ds3 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+
=====
```

また、jndilistを実行してserver3のJNDIリポジトリを確認すると、ds2とds3がJNDIバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server3
The JNDI list on the server3
List of the context /
=====
+-----+-----+-----+
|      Name      |      Value      | Local Binding |
+-----+-----+-----+
| ds2            | jeus.jdbc.info.JDBCBindInfo | true         |
| ds3            | jeus.jdbc.info.JDBCBindInfo | true         |
| JEUSMQ_DLQ    | jeus.jms.common.destination.JeusQueue | false        |
| mgmt           | jeus.jndi.JNSContext   | false        |
+-----+-----+-----+
=====
```

6.6.5. クラスターからのデータソースの削除

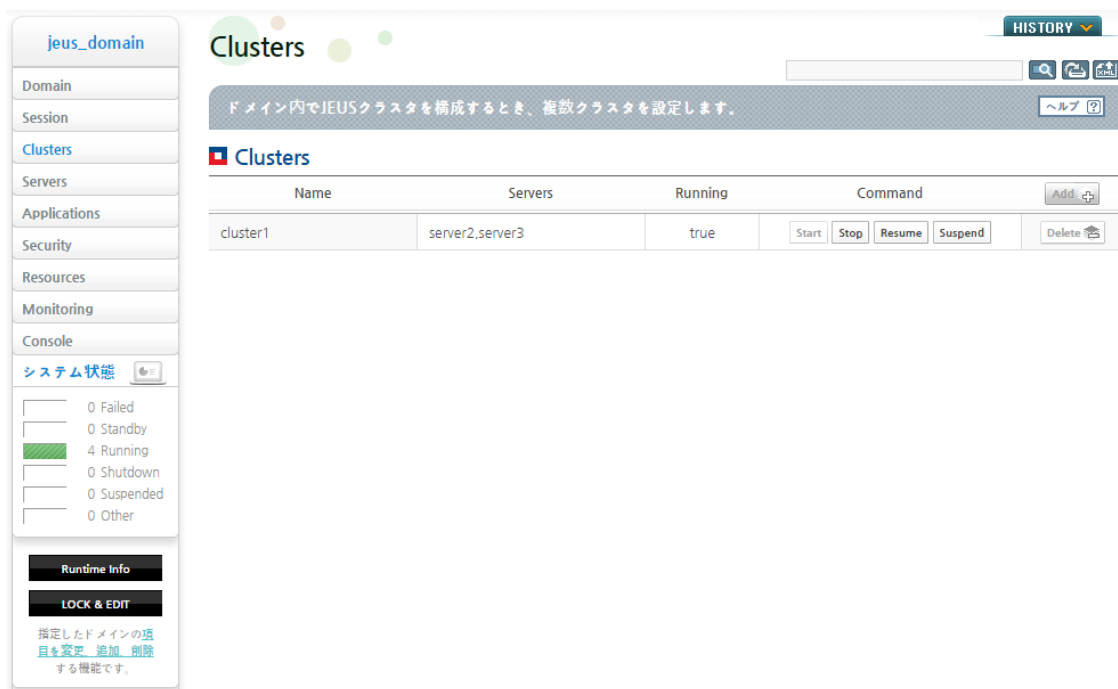
サーバーからデータソースを動的に削除したように、クラスターからもデータソースを動的に削除することができます。クラスターに登録されたデータソースが削除されると、データソースを使用していたクラスター内のすべてのサーバーはそのデータソースを使用できなくなります。つまり、クラスター内のサーバーはクラスターからデータソースが削除されるときに、データソース情報をJNDIアンバインドし、コネクション・プールが作成されている場合、コネクション・プールを破棄します。

WebAdminの使用

以下では、WebAdminを使ってクラスターからデータソースを削除する方法について説明します。

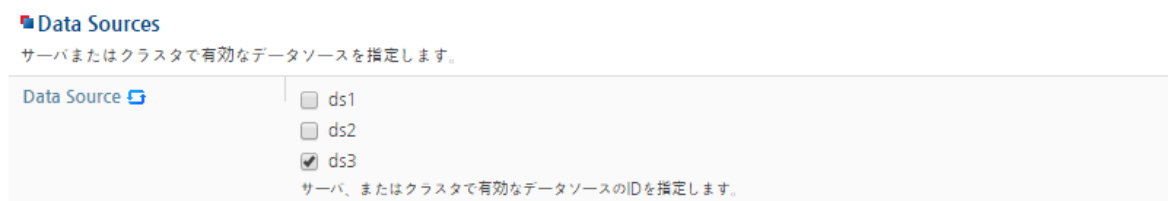
1. WebAdminの左のメニューで[Clusters]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面で任意のクラスターを選択すると、そのクラスターからデータソースを動的に削除できる画面に移動します。

【図 6.27】 クラスターからのデータソースの削除(1)



2. 以下は、cluster1からds2を削除する例です。「Data Source」項目で「ds2」のチェックボックスのチェックを解除して[確認]ボタンをクリックします。

【図 6.28】 クラスターからのデータソースの削除(2)



3. ds2をserver1から削除することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のようにds2がcluster1から削除されたことが確認できます。

[図 6.29] クラスターからのデータソースの削除(3)

Cluster

HISTORY

クラスタ構成のための詳細設定を行います。

ヘルプ

domain.xmlの設定を動的に適用しました。

domain.xml : ACTIVATED

```
clusters.cluster.{? name == 'cluster1' }.dataSources : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - server1
  previous value : ds2, edited value : null, result value : null
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - server2
  previous value : ds2, edited value : null, result value : null
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - adminServer
  previous value : ds2, edited value : null, result value : null
clusters.cluster.{? name == 'cluster1' }.dataSources.ds2 : ACTIVATED - server3
  previous value : ds2, edited value : null, result value : null
```

Basic

Res Ref

Session Cluster Config

Jms Resource

Lifecycle Invocation

動的設定

必須項目

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

TIP

Name

cluster1

クラスタの固有の名前を指定します。この名前はドメイン内で一意である必要があり、クラスタを構成する際、固有の識別子(ID)として使用されます。

Servers

Servers

クラスタに参加するサーバリストを設定します。

Server Name

☐ adminServer

☐ server1

☒ server2

☒ server3

クラスタに参加するサーバの名前を指定します。

5. cluster1からデータソースds2が削除されたので、cluster1に属するserver2とserver3はds2を使用できなくなります。これは、server2とserver3のそれぞれのJNDIリポジトリからds2がJNDIアンバインドされ、ds2のコネクション・プールが作成されていた場合、そのコネクション・プールも破棄されたことを意味します。

JNDIモニタリング情報を確認すると、server2がds2をJNDIバインドしていないことが確認できます。

[図 6.30] サーバーからJNDIアンバインドされたデータソースの確認(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

JNDI

HISTORY

JNDIのモニタリング情報を照会します。

The JNDI list on the server2

adminServer

server1

server2

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds3 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server3

また、server3もds2をJNDIバインドしていないことが確認できます。

[図 6.31] サーバーからJNDIアンバインドされたデータソースの確認(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

JNDI

HISTORY

JNDIのモニタリング情報を照会します。

The JNDI list on the server3

adminServer

server1

server2

server3

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds3 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

コンソール・ツールの使用

コンソール・ツールで**remove-data-sources-from-cluster**を実行すると、データソースを動的にクラスターから削除できます。remove-data-sources-from-clusterのより詳しい使用法は『*JEUS リファレンスガイド*』の「4.2.4.25. remove-data-sources-from-cluster」を参照してください。

以下は、remove-data-sources-from-clusterを実行してds2をcluster1から削除する例です。

```
[DAS]domain1.adminServer>rmdsfromcluster -cluster cluster1 -ids ds2
Successfully performed the REMOVE operation for data sources from the cluster [cluster1].
Check the results using "remove-data-sources-from-cluster -cluster cluster1"
```

remove-data-sources-from-cluster -cluster cluster1を実行すると、cluster1からds2が削除され、ds3だけが残っていることが確認できます。

```
[DAS]domain1.adminServer>rmdsfromcluster -cluster cluster1
Shows the current configuration.
The data sources registered in the cluster [cluster1].
=====
+-----+-----+-----+
| data sources | ds3 |
+-----+-----+-----+
=====
```

cluster1からデータソースds2が削除されたので、cluster1に属するserver2とserver3はds2を使用できなくなります。これは、server2とserver3のそれぞれのJNDIリポジトリからds2がJNDIアンバインドされ、ds2のコネクション・プールが作成されていた場合、そのコネクション・プールも破棄されたことを意味します。

jndilistを実行してserver2のJNDIリポジトリを確認すると、server2がds2をJNDIバインドしていないことが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server2
The JNDI list on the server2
List of the context /
=====
+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+
| ds3 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+
=====
```

また、jndilistを実行してserver3のJNDIリポジトリを確認すると、server3もds2をJNDIバインドしていないことが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server3
The JNDI list on the server3
List of the context /
=====
+-----+-----+-----+
|      Name      |      Value      | Local Binding |
+-----+-----+-----+
| ds3             | jeus.jdbc.info.JDBCBindInfo | true          |
| JEUSMQ_DLQ      | jeus.jms.common.destination.JeusQueue | false         |
| mgmt            | jeus.jndi.JNSContext      | false         |
+-----+-----+-----+
=====
```

6.6.6. クラスターへのサーバーの追加

クラスターにサーバーを動的に追加することができます。クラスターとサーバーの設定を変更しますが、クラスターにサーバーを追加することでデータソースの設定も変更されることがあります。

クラスターにサーバーを動的に追加すると、クラスターに登録されたデータソースはクラスターに追加されたサーバーで有効になります。つまり、クラスターに追加されたサーバーは、クラスターに登録されたデータソースを自身が登録したデータソースのように使用することができます。

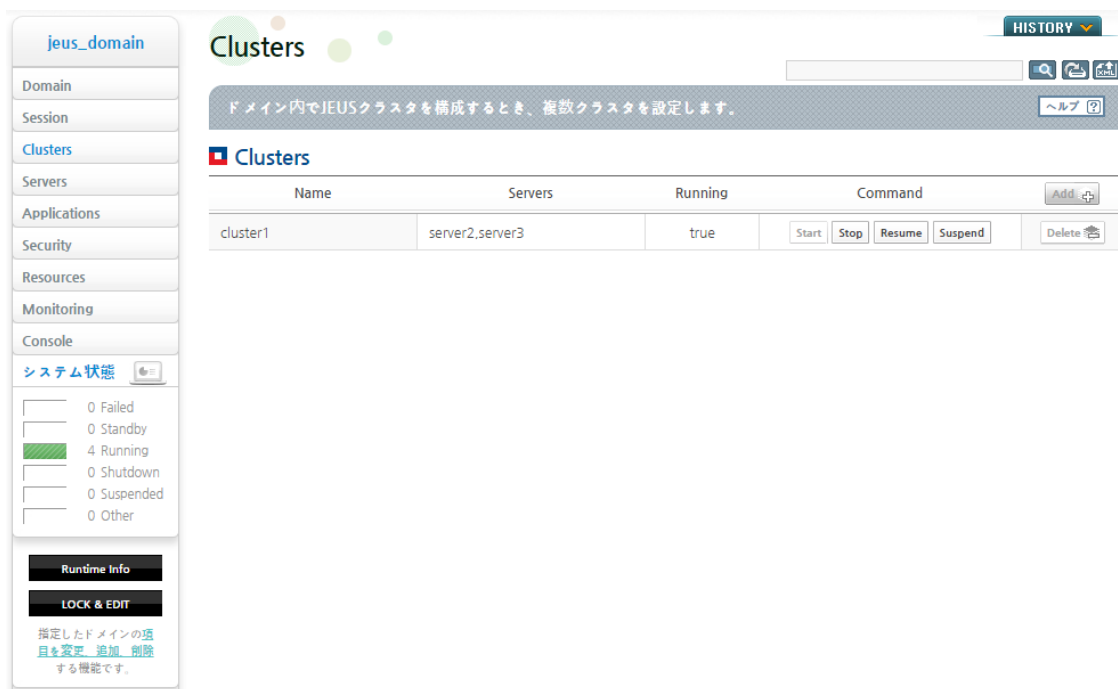
一方、サーバー自身が登録したデータソースは無効になってしまうため、そのデータソースはJNDIアンバインドされ、関連コネクション・プールが作成されている場合は、コネクション・プールが破棄されます。

WebAdminの使用

以下では、WebAdminを使ってクラスターにサーバーを追加する方法について説明します。

1. WebAdminの左のメニューで[Clusters]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面で任意のクラスターを選択すると、クラスターにサーバーを動的に追加できる画面に移動します。

【図 6.32】 クラスターへのサーバーの追加(1)



2. 以下は、cluster1にserver1を追加する例です。「Servers」項目で「server1」のチェックボックスにチェックを入れて[確認] ボタンをクリックします。

【図 6.33】 クラスターへのサーバーの追加(2)



3. server1をcluster1に追加することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、server1がcluster1に追加されたことが確認できます。

【図 6.34】 クラスターへのサーバーの追加(3)

The screenshot shows the JBoss JMX console interface. The left sidebar contains a navigation menu with items like Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. The 'Clusters' item is selected. The main content area is titled 'Cluster' and shows the configuration for 'cluster1'. A message box at the top indicates that the configuration was successfully applied. Below this, there are tabs for 'Basic', 'Res Ref', 'Session Cluster Config', 'Jms Resource', and 'Lifecycle Invocation'. The 'Basic' tab is active, showing the 'Name' field set to 'cluster1'. Below this, there is a section for 'Servers' with a list of servers: 'adminServer', 'server1', 'server2', and 'server3'. The 'server1' checkbox is checked, indicating it is part of the cluster. The 'server2' and 'server3' checkboxes are also checked. The 'adminServer' checkbox is unchecked.

5. server1がcluster1に追加されたので、server1はcluster1に登録されたds3を自身が登録したデータソースのように使用することができます。これは、server1のJNDIリポジトリにds3がJNDIバインドされたことを意味します。一方、server1自身が登録したds1も、依然としてserver1で有効です。

server1のJNDIモニタリング情報を確認すると、ds1と共にds3がJNDIバインドされたことを確認できます。

【図 6.35】 サーバーにJNDIバインドされたデータソースの確認

The screenshot shows the JBoss JMX console interface. The left sidebar contains a navigation menu with items like Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. The 'Monitoring' item is selected. The main content area is titled 'JNDI' and shows the monitoring information for the JNDI. A message box at the top indicates that the JNDI list on the server1 is displayed. Below this, there is a section for 'List of the context /' with a table showing the following entries:

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| ds3 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMO_DLO | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
| server2 | | |
| server3 | | |

コンソール・ツールの使用

コンソール・ツールで**add-servers-to-cluster**を実行すると、サーバーを動的にクラスターに登録できます。
add-servers-to-clusterのより詳しい使用方法是、『*JEUS* リファレンスガイド』の「4.2.4.8. add-servers-to-cluster」を参照してください。

以下は、add-servers-to-clusterを実行してserver1をcluster1に追加する例です。

```
[DAS]domain1.adminServer>add-servers-to-cluster cluster1 -servers server1
Successfully performed the ADD operation for The server list for cluster(cluster1)..
Check the results using "list-clusters cluster1 or add-servers-to-cluster cluster1"
```

上記のメッセージのように、server1がcluster1に追加されたことを、「**list-clusters cluster1**」または「**add-servers-to-cluster cluster1**」を実行して確認できます。

```
[DAS]domain1.adminServer>add-servers-to-cluster cluster1
Shows the current configuration.
The server list for cluster(cluster1).
=====
+-----+-----+
| List of Servers | server2, server3, server1 |
+-----+-----+
=====
```

server1がcluster1に追加されたので、server1はcluster1に登録されたds3を自身が登録したデータソースのように使用することができます。これは、server1のJNDIリポジトリにds3がJNDIバインドされたことを意味します。また、server1自身が登録していたds1は依然としてserver1で有効です。

jndilistを実行してserver1のJNDIレポジトリを確認すると、ds1と共にds3がJNDIバインドされたことを確認することができます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
+-----+-----+-----+-----+
|      Name      |          Value          | Local Binding |
+-----+-----+-----+-----+
| ds1             | jeus.jdbc.info.JDBCBindInfo | true         |
| ds3             | jeus.jdbc.info.JDBCBindInfo | true         |
| JEUSMQ_DLQ     | jeus.jms.common.destination.JeusQueue | false        |
| mgmt           | jeus.jndi.JNSContext      | false        |
+-----+-----+-----+-----+
=====
```


6.6.7. クラスターからのサーバーの削除

クラスターからサーバーを動的に削除することができます。これも、クラスターとサーバーの設定を変更することですが、クラスターからサーバーを削除することでデータソースの設定も変更されることがあります。

クラスターからサーバーを動的に削除すると、クラスターに登録されたデータソースはクラスターから削除されたサーバーでもう有効ではありません。つまり、クラスターから削除されたサーバーは、クラスターに登録されたすべてのデータソース情報をJNDIアンバインドされ、コネクション・プールが作成されている場合は、コネクション・プールが破棄されます。

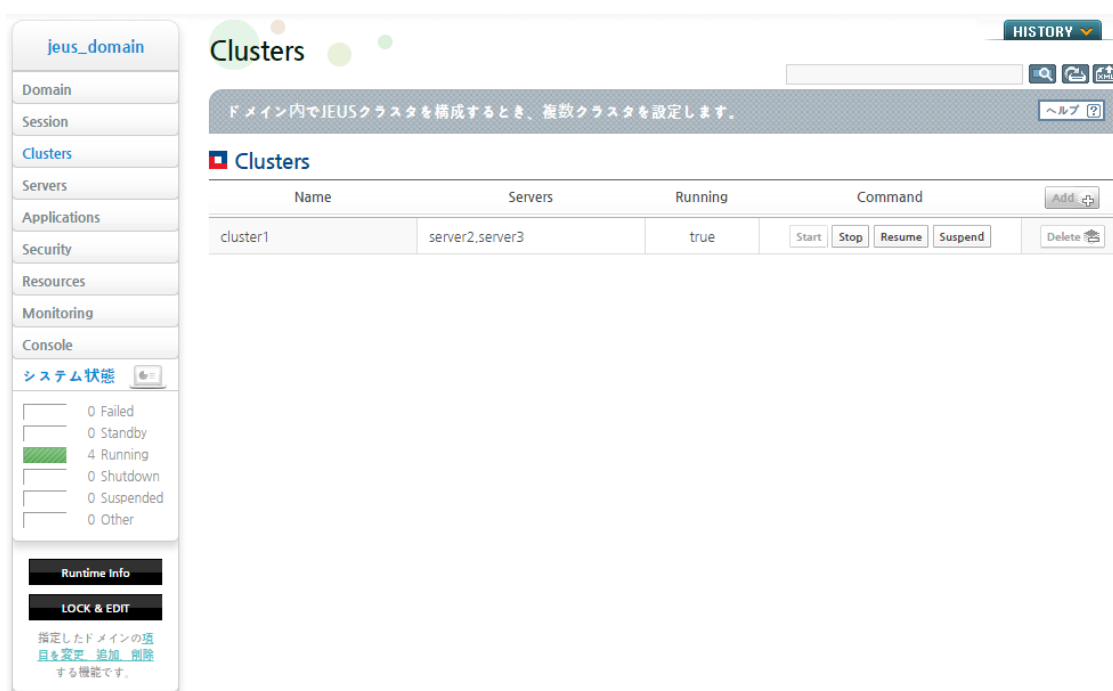
一方、サーバー自身が登録したデータソースは再びそのサーバーで有効になるので、該当するデータソースはJNDIバインドされ、コネクション・プールを作成して使用することができるようになります。

WebAdminの使用

以下では、WebAdminを使ってクラスターからサーバーを削除する方法について説明します。

1. WebAdminの左のメニューで[Clusters]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面で任意のクラスターを選択すると、クラスターからサーバーを動的に削除できる画面に移動します。

[図 6.36] クラスターからのサーバーの削除(1)



2. 以下は、cluster1からserver3を削除する例です。「**Servers**」項目で「server3」のチェックボックスのチェックを解除し、**[確認]**ボタンをクリックします。

[図 6.37] クラスターからのサーバーの削除(2)

Servers

▼ Servers
クラスタに参加するサーバリストを設定します。

| Server Name | Check |
|-------------|-------------------------------------|
| adminServer | <input type="checkbox"/> |
| server1 | <input checked="" type="checkbox"/> |
| server2 | <input checked="" type="checkbox"/> |
| server3 | <input type="checkbox"/> |

クラスタに参加するサーバの名前を指定します。

3. server3をcluster1から削除することを最終反映するために**[Activate Changes]**ボタンをクリックします。

4. 以下のように、server3がcluster1から削除されたことが確認できます。

[図 6.38] クラスターからのサーバーの削除(3)

Cluster

クラスタ構成のための詳細設定を行います。

domain.xmlの設定を動的に適用しました。
domain.xml : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.servers.serverName : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.servers.serverName.{? serverName == 'server3' } : ACTIVATED
previous value : server3, edited value : null, result value : null

Basic Res Ref Session Cluster Config Jms Resource Lifecycle Invocation

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 **TIP**

Name cluster1
クラスタの固有の名前を指定します。この名前はドメイン内で一意である必要があり、クラスタを構成する際、固有の識別子(ID)として使用されます。

Servers

▼ Servers
クラスタに参加するサーバリストを設定します。

| Server Name | Check |
|-------------|-------------------------------------|
| adminServer | <input type="checkbox"/> |
| server1 | <input checked="" type="checkbox"/> |
| server2 | <input checked="" type="checkbox"/> |
| server3 | <input type="checkbox"/> |

クラスタに参加するサーバの名前を指定します。

▼ Dynamic Servers

5. server3がcluster1から削除されたので、server3はcluster1に登録されたds3を自身が登録したデータソースのように使用することができなくなります。これは、server3のJNDIリポジトリからds3がJNDIアンバインドされたことを意味し、ds3のコネクション・プールが作成されている場合は、それも破棄されたことを意味します。

JNDIモニタリング情報を確認すると、server3にどのデータソースもJNDIバインドされていないことが確認できます。

[図 6.39] サーバーからJNDIアンバインドされたデータソースの確認



コンソール・ツールの使用

コンソール・ツールで**remove-servers-from-cluster**を実行すると、サーバーを動的にクラスターから削除できます。remove-servers-from-clusterのより詳しい使用法は、『*JEUS リファレンスガイド*』の「4.2.4.29. remove-servers-from-cluster」を参照してください。

以下は、remove-servers-from-clusterを実行してserver3をcluster1から削除する例です。

```
[DAS]domain1.adminServer>remove-servers-from-cluster cluster1 -servers server3
Successfully performed the REMOVE operation for The server list for cluster(cluster1)..
Check the results using "list-clusters cluster1 or remove-servers-from-cluster cluster1"
```

上記のメッセージのように、server3がcluster1から削除されたことを、「**list-clusters cluster1**」または「**remove-servers-from-cluster cluster1**」を実行して確認できます。

```
[DAS]domain1.adminServer>remove-servers-from-cluster cluster1
Shows the current configuration.
The server list for cluster(cluster1).
=====
+-----+-----+
| List of Servers | server2, server1 |
```

```
+-----+-----+
=====
```

server3がcluster1から削除されたので、server3はcluster1に登録されたds3を自身が登録したデータソースのように使用することができなくなります。これは、server3のJNDIリポジトリからds3がJNDIアンバインドされ、ds3のコネクション・プールが作成されている場合は、それも破棄されたことを意味します。

jndilistを実行してserver3のJNDIリポジトリを確認すると、ds3が削除されていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server3
The JNDI list on the server3
List of the context /
=====
+-----+-----+-----+-----+
|      Name      |          Value          | Local Binding |
+-----+-----+-----+-----+
| JEUSMQ_DLQ     | jeus.jms.common.destination.JeusQueue | false        |
| mgmt           | jeus.jndi.JNSContext    | false        |
+-----+-----+-----+-----+
=====
```

6.6.8. クラスターの削除

ドメインからクラスターを動的に削除することができます。ドメインとクラスターの設定を変更しますが、ドメインからクラスターを削除することで、データソースの設定も変更されることがあります。

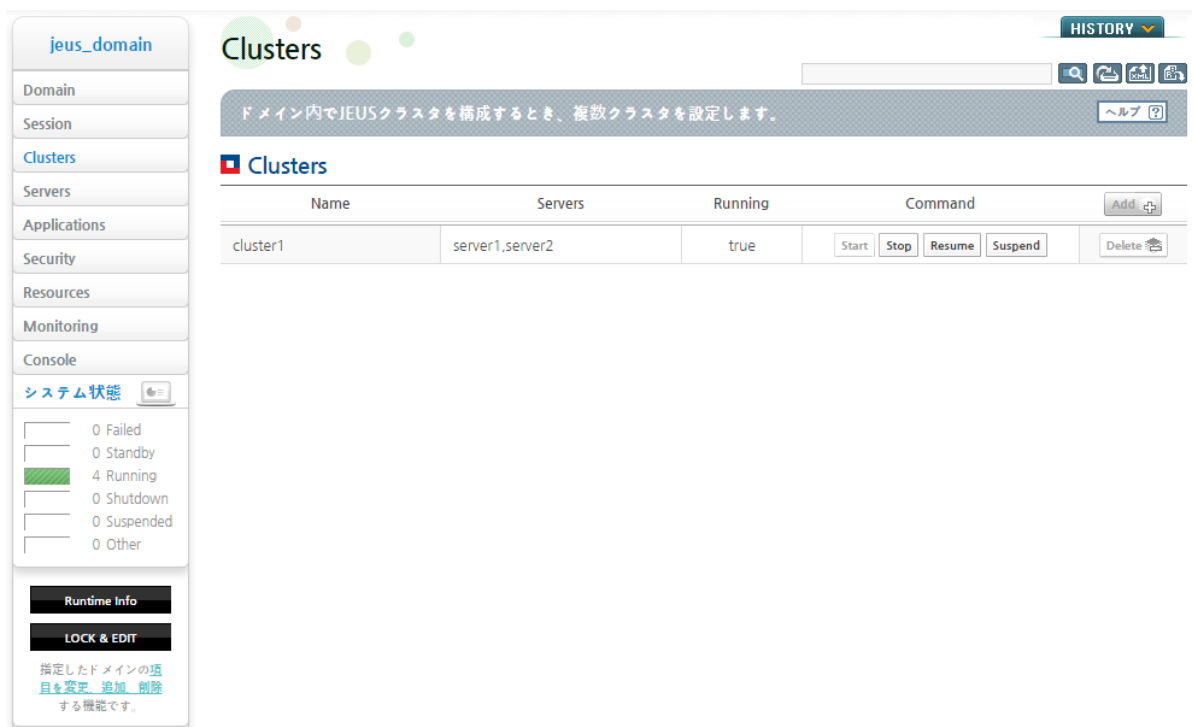
クラスターをドメインから削除すると、クラスターに登録されているすべてのデータソースはクラスターに属するサーバーで無効になります。つまり、削除されたクラスターに属するサーバーはクラスターに登録されていたすべてのデータソース情報をJNDIアンバインドし、コネクション・プールが作成されている場合は、コネクション・プールも破棄します。一方、サーバー自身が登録したデータソースは再び有効になるため、そのようなデータソースはJNDIバインドされ、コネクション・プールを作成して使用できるようになります。

WebAdminの使用

以下では、WebAdminを使ってクラスターを削除する方法について説明します。

1. WebAdminの左のメニューで**[Clusters]**を選択して**[LOCK & EDIT]**ボタンをクリックした後、表示される画面で任意のクラスターに対し**[Delete]**ボタンをクリックすると、そのクラスターをドメインから動的に削除できます。

[図 6.40] クラスターの削除(1)



2. cluster1をドメインから削除することを最終反映するために[Activate Changes]ボタンをクリックします。

3. 以下のように、cluster1がドメインから削除されていることが確認できます。

[図 6.41] クラスターの削除(2)



4. cluster1がdomain1から削除されたので、cluster1に属するserver1とserver2はcluster1に登録されているds3を自身が登録したデータソースのように使用できなくなります。これは、server1とserver2のそれぞれのJNDIリポジトリからds3がJNDIアンバインドされ、ds3のコネクション・プールが作成されている場合は、そのコネクション・プールも破棄されたことを意味します。

server1のJNDIリポジトリを確認すると、ds3はJNDIアンバインドされ、自身が登録したds1のみJNDIバインドされていることが確認できます。

[図 6.42] サーバーからJNDIアンバインドされたデータソースの確認(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

JNDI

HISTORY

JNDIのモニタリング情報を照会します。

The JNDI list on the server1

adminServer

server1

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server2

server3

また、server2のJNDIリポジトリを確認すると、どのデータソースもJNDIバインドされていないことが確認できます。

[図 6.43] サーバーからJNDIアンバインドされたデータソースの確認(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

JNDI

HISTORY

JNDIのモニタリング情報を照会します。

The JNDI list on the server2

adminServer

server1

server2

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server3

コンソール・ツールの使用

コンソール・ツールで**remove-cluster**を実行すると、クラスターを動的にドメインから削除できます。**remove-cluster**のより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.4.22. remove-cluster」を参照してください。

以下は、**remove-cluster**を実行してcluster1をdomain1から削除する例です。

```
[DAS]domain1.adminServer>remove-cluster cluster1
Successfully performed the REMOVE operation for cluster (cluster1).
Check the results using "list-clusters or remove-cluster"
```

上記のメッセージのように、cluster1がdomain1から削除されたことを、**list-clusters**または**remove-cluster**を実行して確認できます。

```
[DAS]domain1.adminServer>remove-cluster
Shows the current configuration.
cluster list
=====
+-----+-----+-----+
| List of Clusters | empty |
+-----+-----+-----+
=====
```

cluster1がdomain1から削除されたので、cluster1に属するserver1とserver2はcluster1に登録されているds3を自身が登録したデータソースのように使用できなくなります。これは、server1とserver2のそれぞれのJNDIリポジトリからds3がJNDIアンバインドされ、ds3のコネクション・プールが作成されている場合は、そのコネクション・プールも破棄されたことを意味します。

jndilistを実行してserver1のJNDI Repositoryリポジトリを確認すると、ds3はJNDIアンバインドされ、自身が登録したds1のみJNDIバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+
=====
```

また、**jndilist**を実行してserver2のJNDIリポジトリを確認すると、どのデータソースもJNDIバインドされていないことが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server2
The JNDI list on the server2
List of the context /
=====
+-----+-----+-----+
|      Name      |      Value      | Local Binding |
+-----+-----+-----+
| JEUSMQ_DLQ     | jeus.jms.common.destination.JeusQueue | false        |
| mgmt           | jeus.jndi.JNSContext | false        |
+-----+-----+-----+
=====
```

6.6.9. データソースの削除

ドメインから動的にデータソースを削除できます。データソースがドメインから削除されると、削除されたデータソースは、それを使用していたサーバーとクラスターでもう有効ではありません。つまり、削除されたデータソースを使用していたサーバーはそのデータソース情報をJNDIアンバインドし、コネクション・プールが作成されている場合は、そのコネクション・プールを破棄します。

WebAdminの使用

現在、ds1はserver1に登録されているので、ds1をドメインから削除するためには、まずWebAdminの**[環境設定]**メニューで**Delete Reference**チェックボックスにチェックを入れる必要があります。WebAdminの**[環境設定]**メニューの調整については、『*JEUS WebAdminガイド*』の「2.3.1. ヘッダー領域」を参照してください。

以下では、WebAdminを使ってデータソースを削除する方法について説明します。

1. WebAdminの左のメニューで**[Resources] > [Data Source]**を選択して**[LOCK & EDIT]**ボタンをクリックした後、表示される画面の**Database**リストで**[Delete]**ボタンをクリックすると、データソースを動的にドメインから削除できます。

以下は、ds1をドメインから削除する例です。

[図 6.44] データソースの削除(1)

The screenshot shows the 'jeus_domain' DataSource configuration page. On the left is a sidebar with navigation links: Domain, Session, Clusters, Servers, Applications, Security, Resources, DataSource (selected), Mail Source, URL Source, Message Bridge, Custom Resource, External Source, External Resource, Concurrency Utilities Resource, and Monitoring. The main content area is titled 'DataSource' and has a 'HISTORY' dropdown. Below the title is a message: 'アプリケーションで使用できるデータソースを定義します。' (Define data sources that can be used by the application.) with a 'ヘルプ' (Help) link. The 'Database' section contains a table with three data sources:

| DataSource ID | DataSource Class Name | DataSource Type | Command | Actions |
|---------------|---|-----------------------------|---------|-------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |
| ds3 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |

Below the table is the 'Cluster DS' section, which is currently empty with the message '該当する内容が存在しません。' (No content exists for this category.)

2. ds1をドメインから削除することを最終反映するために[Activate Changes]ボタンをクリックします。

3. 以下のように、ds1がドメインから削除されていることが確認できます。

[図 6.45] データソースの削除(2)

The screenshot shows the 'jeus_domain' DataSource configuration page after the 'Activate Changes' action. The 'Database' section now only contains two data sources (ds2, ds3), as ds1 has been removed. The 'Cluster DS' section remains empty with the message '該当する内容が存在しません。' (No content exists for this category.)

The 'Activate Changes' dialog box is open, showing the following log:

```

domain.xmlの設定を動的に適用しました。
domain.xml : ACTIVATED
servers.server.[? name == 'server1' ].dataSources : ACTIVATED
servers.server.[? name == 'server1' ].dataSources.ds1 : ACTIVATED
previous value : ds1, edited value : null, result value : null
resources.dataSource.database : ACTIVATED
resources.dataSource.database.ds1 : ACTIVATED
resources.dataSource.database.ds1 : ACTIVATED - server1
previous value : jeus.xml.binding.jeusDD.DatabaseType@163985a, edited value : null, result value : null
resources.dataSource.database.ds1 : ACTIVATED - server2
previous value : jeus.xml.binding.jeusDD.DatabaseType@6c47fd, edited value : null, result value : null
resources.dataSource.database.ds1 : ACTIVATED - adminServer
previous value : jeus.xml.binding.jeusDD.DatabaseType@159d240, edited value : null, result value : null
resources.dataSource.database.ds1 : ACTIVATED - server3
previous value : jeus.xml.binding.jeusDD.DatabaseType@19c46f3, edited value : null, result value : null

```

4. ds1がdomain1から削除されたので、server1はもうds1を使用できなくなります。これは、server1のJNDIリポジトリからds1がJNDIアンバインドされ、ds1のコネクション・プールが作成されている場合は、そのコネクション・プールも破棄されたことを意味します。

server1のJNDIリポジトリを確認すると、ds1が削除されていることが確認できます。

【図 6.46】 サーバーからJNDIアンバインドされたデータソースの確認

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| JEUSMO_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

コンソール・ツールの使用

コンソール・ツールで**remove-data-source**を実行すると、データソースを動的にドメインから削除できます。remove-data-sourceのより詳しい使用方は、『*JEUS リファレンスガイド*』の「4.2.11.2. remove-data-source」を参照してください。

以下は、remove-data-sourceを実行してds1をdomain1から削除する例です。

```
[DAS]domain1.adminServer>rmds -id ds1
Successfully performed the REMOVE operation for data source [ds1] from the domain.
=====
+-----+-----+-----+
| resources.dataSource.database | MODIFY | ACTIVATED |
| servers.server.{? name == 'server1' }.dataSources | MODIFY | ACTIVATED |
+-----+-----+-----+
=====
Check the results using "remove-data-source"
```

上記の結果を見ると、remove-data-sourceを実行してds1を削除するとき、ds1を登録していたserver1にも変化があることが予想できます。上記の結果は、ds1がdomain1から削除されると同時に、server1に登録されていたds1情報も削除されることを表しています。このように、動的に設定を変更するコマンドを実行する

と、それと関連して発生するすべての設定変更のランタイム反映の有無を確認できます。動的な設定変更コマンドのより詳しい使用方法是、『JEUS ドメインガイド』の「第3章 ドメインの設定変更」を参照してください。

上記のメッセージのように、ds1が削除されたことを、**remove-data-source**をオプションなしで実行して確認できます。

```
[DAS]domain1.adminServer>rmlds
Shows the current configuration.
Data sources in domain
=====
+-----+-----+
| ds2 | common data source |
| ds3 | common data source |
+-----+-----+
=====
```

ds1がdomain1から削除されたので、server1はもうds1を使用できなくなります。これは、server1のJNDIリポジトリからds1がJNDIアンバインドされ、ds1のConnection Poolコネクション・プールが作成されている場合は、それも破棄されたことを意味します。

jndilistを実行してserver1のJNDIリポジトリを確認すると、ds1が削除されていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
+-----+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+-----+
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+-----+
=====
```

6.6.10. データソース設定の変更

これまで、データソース、サーバー、クラスターを追加、削除するとき、データソースを管理する方法について説明しました。本節では、データソースの設定の変更方法について説明します。

「6.4. データソース設定」で触れたように、データソースの設定には大きくドライバーの設定などに必要な基本設定とコネクション・プールの設定があります。このうち、ドライバーの設定は動的に変更できませんが、その他の一部の基本設定とコネクション・プール設定のほとんどは動的に変更ができます。動的に変更できない設定に対して変更を試みると、変更事項が設定ファイルには記録されるけれど、ランタイムには反映されません。つまり、変更事項はサーバーを再起動するときに反映されることを意味します。一方、動的に変更できる設定は、変更を試みるときに変更事項が設定ファイルにも記録され、ランタイムに直ちに反映されます。

動的に変更できるデータソースの設定は色々ありますが、その中で比較的に変更頻度が高いと予想されるコネクション・プールのコネクションの最小値、最大値の設定を例に挙げて説明します。変更対象となるデータソースはds2です。コネクション・プールのコネクションの最小値、最大値の設定変更が動的に反映されたかどうか確認するために、ds2をserver1に登録してコネクション・プールを作成します。データソースをサーバーに登録する方法については前で説明したので、ここではds2コネクション・プールをserver1に作成する方法から説明します。

WebAdminの使用

以下では、WebAdminを使ってコネクション・プールを作成する方法について説明します。

1. WebAdminの左のメニューで[Monitoring] > [Connection Pools]を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールを作成できます。ds2のコネクション・プールを作成するためにds2をクリックします。

[図 6.47] コネクション・プールの作成(1)

The screenshot shows the 'Connection Pool' management interface in WebAdmin. On the left, a sidebar menu has 'Monitoring' selected. The main panel is titled 'Connection Pool' and contains a search bar and a list of servers: 'adminServer' and 'server1'. Below the server list, a section titled 'The connection pool information on the server [server1].' contains a table with the following data:

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

Below the table, a message states: '* : has not been created, total = active + idle + disposable'. At the bottom of the panel, there are buttons for 'server2' and 'server3'.

2. ds2のコネクション・プールのリストが照会されます。コネクション・プールを作成するためには、**[create]**ボタンをクリックします。

【図 6.48】コネクション・プールの作成(2)

The screenshot shows the 'Connection Pool' management page. On the left is a sidebar with a tree view containing 'jeus_domain' and various sub-items like 'Domain', 'Session', 'Clusters', 'Servers', 'Applications', 'Security', 'Resources', 'Monitoring', 'Thread', 'Transaction', 'MBean', 'JNDI', 'Web', 'Servers', 'JMS', 'Connection Pool', 'EJB Timer', 'System Info', 'Server Log', 'Statistic', and 'Patch Info'. The 'Connection Pool' item is selected. The main content area has a title 'Connection Pool' and a subtitle 'サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。'. Below this is a list of servers: 'adminServer' and 'server1'. A section titled 'Information about connections in the server [server1]'s connection pool [ds2]. contains a table with columns: Connection ID, State, State Time(sec), Use Count, and Type. The table is empty, with a message '該当する内容が存在しません。' (No content exists). Below this is a section titled 'The connection pool information on the server [server1].' containing a table with columns: Connection Pool ID, JNDI Export Name, Min, Max, Active, Idle, Disposable, Total, Wait, Enabled, and a 'stmt' button. The table has one row for 'ds2' with values: ds2, ds2, 2, 30, 0, 0, 0, 0, false, false. Below the table is a message 'has not been created, total = active + idle + disposable'. At the bottom are buttons: Enable, Shrink, Disable, Refresh, and Create.

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|------|
|---------------|-------|-----------------|-----------|------|

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds2 | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

3. ds2に対するコネクション・プールが作成されたことが確認できます。

【図 6.49】コネクション・プールの作成(3)

The screenshot shows the 'Connection Pool' management page, similar to the previous one, but with updated data. The 'Information about connections in the server [server1]'s connection pool [ds2]. section now contains a table with two rows of active connections. The 'The connection pool information on the server [server1].' section also shows updated values for the 'ds2' pool, including an 'Active' count of 2 and an 'Enabled' status of 'true'.

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds2-1 | idle | 5.059 | 0 | pooled |
| ds2-2 | idle | 5.02 | 0 | pooled |

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds2 | ds2 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |

ds2コネクション・プールのランタイム・コネクションの最小値が2、最大値が30に設定されていることが確認できます。これは、ds2を最初にドメインに追加するときにコネクションの最小値、最大値を特に設定しなかったため、デフォルトで適用された値です。以下では、それぞれ2、30に設定されたds2のコネクションの最小値と最大値を変更する手順を説明します。

1. WebAdminの左のメニューで[Resources] > [Data Source]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面のDatabaseリストで任意のデータソースを選択すると、そのデータソースの設定を動的に変更できます。

[図 6.50] データソース設定の変更(1)

The screenshot shows the 'DataSource' configuration page. The left sidebar has a menu with 'jeus_domain' at the top, followed by 'Domain', 'Session', 'Clusters', 'Servers', 'Applications', 'Security', and 'Resources'. Under 'Resources', 'DataSource' is selected. The main content area is titled 'DataSource' and has a 'HISTORY' button. Below the title is a search bar and a 'ヘルプ' button. The main section is 'Database' and contains a table with the following data:

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|-------------------|
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |
| ds3 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |

Below the table is a 'Cluster DS' section with a table that is currently empty, showing a message '該当する内容が存在しません。'.

2. 以下は、ds2コネクション・プールのコネクションの最小値、最大値をそれぞれ10、50に変更する例です。「Min」、「Max」設定値をそれぞれ10、50に入力した後、[確認]ボタンをクリックします。

[図 6.51] データソース設定の変更(2)

The screenshot shows the 'Connection Pool' configuration page. The left sidebar has a menu with 'jeus_domain' at the top, followed by 'Domain', 'Session', 'Clusters', 'Servers', 'Applications', 'Security', and 'Resources'. Under 'Resources', 'DataSource' is selected. The main content area is titled 'Connection Pool' and has a 'HISTORY' button. Below the title is a search bar and a 'ヘルプ' button. The main section is 'Pooling' and contains a table with the following data:

| Field | Value | Description |
|--------|------------|---|
| Min | 10 | [デフォルト: 2] プーリングされるオブジェクトの最小値を設定します。 |
| Max | 50 | [デフォルト: 30] プーリングされるオブジェクトの最大値を設定します。 |
| Step | 1 | [デフォルト: 1] プールにコネクションが不足する場合、現在作成されたコネクションが最大値以下の場合は新規作成します。このとき新規作成するコネクション数を設定します。 |
| Period | 3600000 ms | [デフォルト: 3600000] 指定した周期になったらコネクションプールのサイズを最小値に合わせて調整します。コネクションプールのサイズが最小値を超過する場合は使用しないコネクションをクローズし、コネクションプールのサイズが最小値に満たない場合にはコネクションを新たに確立します。ミリ秒単位で設定します。 |

3. ds2コネクション・プールのコネクションの最小値、最大値を、それぞれ10、50に変更することを最終反映するために[**Activate Changes**]ボタンをクリックします。
4. 以下のように、ds2コネクション・プールのコネクションの最小値、最大値がそれぞれ10、50に変更されていることが確認できます。

[図 6.52] データソース設定の変更(3)

jeus_domain

Domain
Session
Clusters
Servers
Applications
Security
Resources

DataSource
Mail Source
URL Source
Message Bridge
Custom Resource
External Source
External Resource
Concurrency Utilities
Resource

Monitoring
Console
システム状態

0 Failed
0 Standby
4 Running
0 Shutdown
0 Suspended
0 Other

Runtime Info
LOCK & EDIT
指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル
Domain
Server もっと見る

Connection Pool

HISTORY

データソース別にコネクションプール情報を設定します。

domain.xml の設定を動的に適用しました。
domain.xml : ACTIVATED

```
resources.dataSource.database.{? dataSourceId == 'ds2' } : ACTIVATED
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool : ACTIVATED
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling : ACTIVATED
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.min : ACTIVATED
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.min : ACTIVATED - server1
previous value : 2, edited value : 10, result value : 10
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.min : ACTIVATED - server2
previous value : 2, edited value : 10, result value : 10
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.min : ACTIVATED - adminServer
previous value : 2, edited value : 10, result value : 10
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.min : ACTIVATED - server3
previous value : 2, edited value : 10, result value : 10
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.max : ACTIVATED
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.max : ACTIVATED - server1
previous value : 30, edited value : 50, result value : 50
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.max : ACTIVATED - server2
previous value : 30, edited value : 50, result value : 50
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.max : ACTIVATED - adminServer
previous value : 30, edited value : 50, result value : 50
resources.dataSource.database.{? dataSourceId == 'ds2' }.connectionPool.pooling.max : ACTIVATED - server3
previous value : 30, edited value : 50, result value : 50
```

Basic **Connection Pool**

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 **TIP**

Pooling
JDBCコネクションプールのサイズおよび調整に関する設定です。

| | |
|------|--|
| Min | 10 [デフォルト: 2] プーリングされるオブジェクトの最小値を設定します。 |
| Max | 50 [デフォルト: 30] プーリングされるオブジェクトの最大値を設定します。 |
| Step | 1 [デフォルト: 1] プールにコネクションが不足する場合、現在作成されたコネクションが最大値以下の場合には新規作成します。このとき新規作成するコネクション数を設定します。 |

5. 以下は、server1に再び作成されたds2コネクション・プールのランタイム・コネクションの最小値、最大値を確認した結果です。

【図 6.53】コネクション・プールのランタイム情報の確認

jeus_domain

Domain
Session
Clusters
Servers
Applications
Security
Resources
Monitoring

Thread
Transaction
MBean
JNDI
Web
Servers
JMS
Connection Pool
EJB Timer
System Info
Server Log
Statistic
Patch Info

Console
システム状態

0 Failed
0 Standby
4 Running
0 Shutdown
0 Suspended
0 Other

Connection Pool

HISTORY

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

ヘルプ

adminServer
server1

Information about connections in the server [server1]'s connection pool [ds2].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds2-6 | idle | 37.294 | 0 | pooled |
| ds2-1 | idle | 170.1 | 0 | pooled |
| ds2-3 | idle | 37.443 | 0 | pooled |
| ds2-4 | idle | 37.383 | 0 | pooled |
| ds2-10 | idle | 37.046 | 0 | pooled |
| ds2-9 | idle | 37.085 | 0 | pooled |
| ds2-5 | idle | 37.339 | 0 | pooled |
| ds2-2 | idle | 170.061 | 0 | pooled |
| ds2-8 | idle | 37.125 | 0 | pooled |
| ds2-7 | idle | 37.165 | 0 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds2 | ds2 | 10 | 50 | 0 | 10 | 0 | 10 | false | true |

* : has not been created, total = active + idle + disposable

server2
server3

コンソール・ツールの使用

コネクション・プールはコンソール・ツールで**create-connection-pool**を実行して作成できます。create-connection-poolのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.12.1. create-connection-pool」を参照してください。

以下は、create-connection-poolを実行してds2のコネクション・プールを作成する例です。ds2のコネクション・プールがserver1に作成されていることを結果メッセージから確認できます。

```
[DAS]domain1.adminServer>createcp -id ds2
Servers that successfully created a connection pool : server1
Servers that failed to create a connection pool : none.
```


connection-pool-infoを実行すると、コネクション・プールのランタイム・コネクションの最小値、最大値を確認できます。このコマンドについてのより詳しい説明は、『*JEUS リファレンスガイド*』の「4.2.12.7. connection-pool-info」を参照してください。

以下は、connection-pool-infoを実行してserver1に作成されたds2コネクション・プールのランタイム・コネクションの最小値、最大値を確認する例です。

```
[DAS]domain1.adminServer>cpinfo -server server1
The connection pool information on the server [server1].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Connection | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
| Pool ID   |     |     |         |      |              |       |      |          |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ds2       | 2   | 30  | 0       | 2   | 0           | 2     |      | true    |
+-----+-----+-----+-----+-----+-----+-----+-----+
* : has not been created, total = active + idle + disposable
=====
```

ds2コネクション・プールのランタイム・コネクションの最小値が2、最大値が30に設定されていることが確認できます。これは、ds2を最初にドメインに追加するときにコネクションの最小値、最大値を特に設定しなかったため、デフォルトで適用された値です。

以下は、それぞれ2、30に設定されているds2のコネクションの最小値と最大値を変更する方法についての説明です。

modify-data-sourceを実行すると、動的にデータソースの設定を変更できます。helpを実行するか、『*JEUS リファレンスガイド*』の「4.2.11.3. modify-data-source」を参照すると、modify-data-sourceにより変更できるすべての設定をオプションの名前から類推して確認できます。動的変更が可能な設定のオプションにはタグが付けられるため、どの設定が動的に変更できるかも分かります。

以下は、modify-data-sourceを実行してds2のコネクションの最小値と最大値をそれぞれ10、50に変更する例です。

```
[DAS]domain1.adminServer>modifyds -id ds2 -min 10 -max 50
Successfully performed the MODIFY operation for configuration of the data source [ds2].
Check the results using "modify-data-source -id ds2"
```

上記のメッセージのように、modify-data-sourceを実行して変更されたds2の設定は「modify-data-source -id ds2」を実行して確認できます。min、maxがそれぞれ10、50に設定されていることが確認できます。

```
[DAS]domain1.adminServer>modifyds -id ds2
Shows the current configuration.
configuration of the data source [ds2]
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id     |                                     | ds2                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

| | |
|-----------------------------|---|
| export-name | ds2 |
| data-source-class-name | oracle.jdbc.pool.OracleConnectionPoolDataSource |
| data-source-type | ConnectionPoolDataSource |
| server-name | 192.168.1.165 |
| port-number | 1521 |
| database-name | ora10g |
| user | jeustest1 |
| password | jeustest1 |
| login-timeout | 0 |
| auto-commit | DRIVER |
| stmt-query-timeout | 0 |
| pool-destroy-timeout | 10000 |
| property | driverType;java.lang.String;thin |
| support-xa-emulation | false |
| min | 10 |
| max | 50 |
| step | 1 |
| period | 3600000 |
| enable-wait | false |
| wait-time | 10000 |
| max-use-count | 0 |
| dbaTimeout | -1 |
| stmt-caching-size | -1 |
| stmt-fetch-size | -1 |
| connection-trace | false |
| get-connection-trace | true |
| auto-commit-trace | false |
| use-sql-trace | false |
| keep-connection-handle-open | false |

=====

以下は、再び**connection-pool-info**を実行してserver1に作成されているds2コネクション・プールのラインタイム・コネクションの最小値、最大値を確認する例です。ds2のランタイム・コネクションの最小値、最大値がそれぞれ10、50に変更されていることが確認できます。

```
[DAS]domain1.adminServer>cpinfo -server server1
The connection pool information on the server [server1].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Connection | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
| Pool ID   |     |     |         |      |              |       |      |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ds2       | 10 | 50 | 0       | 10  | 0           | 10    | false | true   |
+-----+-----+-----+-----+-----+-----+-----+
* : has not been created, total = active + idle + disposable
=====
```

6.6.11. データソース設定の確認

本節では、WebAdminとコンソール・ツールを使用して、データソースの設定を確認する方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってデータソースの設定を確認する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [Data Source]メニューを選択した後、表示される画面の **Database** リストで任意のデータソースを選択すると、そのデータソースの設定を確認できます。

[図 6.54] データソース設定の確認(1)

The screenshot shows the WebAdmin interface for configuring data sources. The left sidebar is titled 'jeus_domain' and lists various system components. The 'Resources' section is expanded, and 'DataSource' is selected. The main content area is titled 'DataSource' and includes a search bar, a 'HISTORY' dropdown, and a 'ヘルプ' (Help) button. Below this is a message: 'アプリケーションで利用できるデータソースを定義します。' (Define data sources available for use in the application). The 'Database' section contains a table with the following data:

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|------------------|
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete Duplicate |
| ds3 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete Duplicate |

Below the table is the 'Cluster DS' section, which has a table with the following headers:

| DataSource ID | Export Name |
|---------------|-------------|
|---------------|-------------|

The message '該当する内容が存在しません。' (No matching content exists.) is displayed below the table.

2. 以下は、ds2の設定を確認する例です。

[図 6.55] データソース設定の確認(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

DataSource

Mail Source

URL Source

Message Bridge

Custom Resource

External Source

External Resource

Concurrency Utilities Resource

Monitoring

Console

システム状態

0 Failed

0 Standby

4 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル

Domain

Server

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

Data Source Id

ds2

データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。

Export Name

ds2

データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。

Vendor

oracle

入力

JDBCドライバベンダの名前を指定します。

Data Source Class Name

oracle.jdbc.pool,OracleConnectionPoolDataSource

oracle.jdbc.pool,OracleConnectionPoolDataSource

JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。

Data Source Type

ConnectionPoolDataSource

データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。

Server Name

192.168.2.49

DBが実行されるホスト名、またはIPを設定します。

Port Number

1521

DBリスナーのポート番号を設定します。

Database Name

ora10g

DBの名前を指定します。Oracleの場合、DBのSIDを設定します。

User

jeusdb2

DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。

Password

.....

入力

{DES}FQrLbQ/D8O1IDVS71L28rw==

DBユーザのパスワードを設定します。暗号化して保存するときは、{algorithm}ciphertextの形式で記述します。

Support Xa Emulation

☐

[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEU56までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。

詳細設定

すべてを開く

Description

Login Timeout

0

s

Isolation Level

Auto Commit

DRIVER

Stmt Query Timeout

0

ms

Pool Destroy Timeout

10000

ms

Property

driverType:java.lang.String=thin

EX name:type=value

Action On Connection Leak

コンソール・ツールの使用

コンソール・ツールで**list-data-sources**コマンドを実行すると、ドメインに存在するすべてのデータソースのリストを確認できます。

以下は、list-data-sourcesを実行してドメインに存在するすべてのデータソース・リストを確認する例です。

```
[DAS]domain1.adminServer>lsds
The list of data sources
=====
+-----+-----+-----+
| Data Source ID | JNDI Export Name | Data Source Type |
+-----+-----+-----+
| ds2            | ds2              | ConnectionPoolDataSource |
| ds3            | ds3              | ConnectionPoolDataSource |
+-----+-----+-----+
```

特定のデータソースのIDをオプション値として入力すると、そのデータソースの全体設定の詳細を照会できます。

以下は、list-data-sourcesを実行してds2の詳細設定を確認する例です。

```
[DAS]domain1.adminServer>lsds -id ds2
The configuration of the data source [ds2]
=====
+-----+-----+
| Configuration name | Configuration value |
+-----+-----+
| id                 | ds2                  |
| export-name        | ds2                  |
| data-source-class-name | oracle.jdbc.pool.OracleConnectionPoolDataSource |
| data-source-type    | ConnectionPoolDataSource |
| server-name        | 192.168.1.165        |
| port-number        | 1521                  |
| database-name       | ora10g                |
| user               | jeustest1             |
| password           | jeustest1             |
| login-timeout       | 0                      |
| auto-commit         | DRIVER                |
| stmt-query-timeout  | 0                      |
| pool-destroy-timeout | 10000                 |
| property            | [driverType;java.lang.String;thin] |
| support-xa-emulation | false                 |
| min                 | 10                     |
```

| | | | |
|-----------------------------|-------|---------|--|
| max | | 50 | |
| step | | 1 | |
| period | | 3600000 | |
| enable-wait | false | | |
| wait-time | | 10000 | |
| max-use-count | | 0 | |
| dbaTimeout | | -1 | |
| stmt-caching-size | | -1 | |
| stmt-fetch-size | | -1 | |
| connection-trace | false | | |
| get-connection-trace | true | | |
| auto-commit-trace | false | | |
| use-sql-trace | false | | |
| keep-connection-handle-open | false | | |
| +-----+ | | | |
| ===== | | | |

6.7. クラスター・データソース関連設定の動的変更

本節では、クラスター・データソースと関連する設定の動的変更方法について説明します。理解をより深めるためには、JEUSのドメイン構造やドメインでのデータソース管理構造などを理解し、クラスター・データソースの各設定項目の使用について熟知している必要があります。

JEUSのドメイン構造については、『*JEUS ドメインガイド*』の「1.3. 構成要素」を参照してください。ドメインでのデータソース管理構造とクラスター・データソースの各設定項目の使用については、それぞれ「[6.3. データソースとコネクション・プールの管理](#)」と「[6.5. クラスター・データソースの設定](#)」を参照してください。

動的変更は、コンソール・ツールまたはWebAdminを使って実行することができ、本節では両方の方法を説明します。コンソール・ツールを使った方法を説明するときは、便宜上縮約されたコマンドおよびオプションの名前を使用します。縮約コマンドおよびオプション名とコマンドの詳しい使用方法については、コンソール・ツールでコマンド名を引数としてhelpを実行するか、『*JEUS リファレンスガイド*』の「4.2.12. コネクション・プールのモニタリングおよび制御コマンド」を参考して確認できます。

説明を行うために必要な事前作業をまず行います。

まずdomain1という名前のドメインとadminServerという名前のDASを作成します。DASを起動した後、それぞれserver1、server2、server3という名前を持つ3つのMSをdomain1に追加して起動します。そして、server2とserver3はcluster1という名前のクラスターで構成します。ドメイン、クラスター、およびサーバーの設定方法については、『*JEUS ドメインガイド*』の「第2章 ドメインの作成」および『*JEUS ドメインガイド*』の「第5章 JEUS クラスタリング」、「[第2章 JEUSの設定](#)」を参照してください。以降の説明では、この事前作業がすべて完了したことを仮定します。

参考

それぞれの例をシナリオを構成して説明するので、本節の内容は、記述された順に従って読み進んでください。

6.7.1. クラスター・データソースの追加

一般のデータソースのようにクラスター・データソースも動的に追加することができます。クラスター・データソースを追加するときに常に考慮すべきことは、クラスター・データソースで構成されるコンポーネント・データソースも一緒に追加する必要があるということです。

本節では、データソースIDがそれぞれds1、ds2のデータソースをコンポーネント・データソースとして持つクラスター・データソースをdomain1に追加する例について説明します。クラスター・データソースのデータソースIDはcds1です。

WebAdminの使用

cds1のコンポーネント・データソースとして動作するds1とds2をdomain1に追加する必要があります。WebAdminを使ってデータソースをドメインに追加する方法は、「6.6.1. データソースの追加」を参照してください。以降の説明では、ds1、ds2がdomain1に追加されていることを仮定します。

1. WebAdminの左のメニューで[Resources] > [Data Source]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面のCluster Dsリストで[Add]ボタンをクリックすると、動的にデータソースをドメインに追加できる画面に移動します。

【図 6.56】 クラスター・データソースの追加(1)

2. 以下は、必要な設定を入力してデータソースIDがcds1のクラスター・データソースをドメインに追加する例です。設定項目を入力した後、[確認]ボタンをクリックします。

[図 6.57] クラスター・データソースの追加(2)

The screenshot shows the 'Cluster DS' configuration window in the Oracle JDeveloper IDE. The left sidebar displays the 'jeus_domain' project structure with various components like Domain, Session, Clusters, Servers, Applications, Security, and Resources. The 'Resources' section is expanded, showing 'DataSource', 'Mail Source', 'URL Source', 'Message Bridge', 'Custom Resource', 'External Source', 'External Resource', and 'Concurrency Utilities Resource'. The 'Monitoring' section shows 'Console' and 'システム状態' (System Status). The 'Runtime Info' tab is selected, showing 'Activate Changes' and 'Undo All Changes' buttons. The main area is titled 'Cluster DS' and contains a description of the cluster data source. Below the description, there are several configuration fields: 'Data Source Id' (cds1), 'Export Name' (empty), 'Data Source Selector' (foo.bar.MyDataSourceSelector), 'Load Balance' (unchecked), and 'Component Data Sources' (ds2, ds1). At the bottom, there are checkboxes for 'Is Pre Conn' (unchecked), 'Use Failback' (checked), 'Xa Affinity' (checked), and 'Ons Support' (unchecked). The '確認' (Confirm) and '再設定' (Reset) buttons are at the bottom right.

3. cds1をドメインに追加することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、cds1がドメインに追加されたことが確認できます。

[図 6.58] クラスター・データソースの追加(3)

The screenshot shows the JBoss JConsole interface. On the left is a navigation pane with tabs for 'jeus_domain', 'Domain', 'Session', 'Clusters', 'Servers', 'Applications', 'Security', 'Resources', 'Monitoring', and 'Console'. The 'Resources' tab is selected, and the 'DataSource' sub-tab is active. The main area displays the 'DataSource' configuration for 'jeus_domain'. It shows a list of data sources and their properties. The 'Database' section shows two data sources, 'ds1' and 'ds2', both of type 'CONNECTION_POOL_DATA_SOURCE'. The 'Cluster DS' section shows a single data source, 'cds1', of type 'cds1'.

| DataSource ID | DataSource Class Name | DataSource Type | Command |
|---------------|---|-----------------------------|---------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test |
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test |

| DataSource ID | Export Name |
|---------------|-------------|
| cds1 | cds1 |

コンソール・ツールの使用

cds1のコンポーネント・データソースとして動作するds1とds2をdomain1に追加する必要があります。

以下は、**add-data-source**を実行してds1とds2をdomain1に追加する例です。add-data-sourceのより詳しい使用法は、『*JEUS リファレンスガイド*』の「4.2.11.1. add-data-source」を参照してください。

```
[DAS]domain1.adminServer>addds -id ds1 -dst ConnectionPoolDataSource -dscn
oracle.jdbc.pool.OracleConnectionPoolDataSource -sn 192.168.1.165 -pn 1521 -dn ora10g -user
jeustest1 -password jeustest1 -property driverType;java.lang.String;thin
Successfully performed the ADD operation for data source [ds1] to domain.
Check the results using "add-data-source"

[DAS]domain1.adminServer>addds -id ds2 -dst ConnectionPoolDataSource -dscn
oracle.jdbc.pool.OracleConnectionPoolDataSource -sn 192.168.1.165 -pn 1521 -dn ora10g -user
jeustest1 -password jeustest1 -property driverType;java.lang.String;thin
Successfully performed the ADD operation for data source [ds2] to domain.
Check the results using "add-data-source"
```

ds1とds2をdomain1に追加したので、cds1をdomain1に追加します。**add-cluster-data-source**を実行すると、動的にクラスター・データソースをドメインに追加できます。add-cluster-data-sourceのより詳しい使用方法は、『*JEUS リファレンスガイド*』の「4.2.11.6. add-cluster-data-source」を参照してください。add-cluster-data-sourceを実行してデータソースを追加するには、クラスター・データソースの詳細設定のオプション値を指定する必要があります。

以下は、**add-cluster-data-source**を実行して必要な設定を入力し、クラスター・データソースIDがcds1のデータソースをdomain1に追加する例です。

```
[DAS]domain1.adminServer>addcds -id cds1 -cds ds1,ds2
Successfully performed the ADD operation for cluster data source [cds1] to domain.
Check the results using "add-cluster-data-source"
```

上記のメッセージのように、cds1が追加されたことを、**add-cluster-data-source**をオプションなしで実行して確認できます。

```
[DAS]domain1.adminServer>addcds
Shows the current configuration.
Data sources in domain
=====
+-----+-----+-----+
| ds1 | common data source |
| ds2 | common data source |
| cds1 | cluster data source |
+-----+-----+-----+
=====
```

6.7.2. サーバーへのクラスター・データソースの登録

クラスター・データソースを使用するためには、一般データソースのようにサーバーに登録する必要があります。クラスター・データソースをサーバーに登録すると、クラスター・データソースの情報がサーバーのJNDIリポジトリにバインドされ、以降サーバーにデプロイされたアプリケーションがこれをルックアップできます。ここで注意することは、クラスター・データソースとして構成されたコンポーネント・データソースもサーバーに登録する必要があるということです。

実際にコネクション・プール・サービスを提供するのはそれぞれのコンポーネント・データソースであり、コネクション・プールはサーバーに生成されるため、クラスター・データソースを正常に使用するためには、クラスター・データソースを登録するサーバーにそのクラスター・データソースを構成するコンポーネント・データソースと一緒に登録する必要があります。

本節では、cds1をserver1に登録する方法について説明します。

WebAdminの使用

cds1のコンポーネント・データソースであるds1とds2もserver1に登録する必要があります。WebAdminを使ってデータソースをサーバーに登録する方法は「[6.6.2. サーバーへのデータソースの登録](#)」を参照してください。以降の説明では、ds1とds2がserver1に登録されていることを仮定します。

1. WebAdminの左のメニューで[**Servers**]を選択して[**LOCK & EDIT**]ボタンをクリックした後、表示される画面で任意のサーバーを選択すると、そのサーバーにクラスター・データソースを動的に登録できる画面に移動します。

[図 6.59] サーバーへのクラスター・データソースの登録(1)

The screenshot shows the 'Servers' management interface. On the left, a sidebar lists navigation options: Domain, Session, Clusters, Servers (selected), Applications, Security, Resources, Monitoring, and Console. Below these is a 'システム状態' (System Status) section showing counts for Failed, Standby, Running (4), Shutdown, Suspended, and Other. At the bottom of the sidebar are 'Runtime Info' and 'LOCK & EDIT' buttons. The main content area is titled 'Servers' and includes a search bar and a 'HISTORY' dropdown. A message states: 'ドメイン内でJEUSサーバを構成するとき、各サーバの設定を行います。' (When configuring JEUS servers within the domain, configure each server). Below this is a table of servers with columns: Name, Status, Pid, Need To Restart, Command, and actions (Delete, Duplicate). The table lists four servers: adminServer (*), server1, server2, and server3, all with a status of 'RUNNING'. Below the table is a 'Server Templates' section with a table that is currently empty, displaying the message '該当する内容が存在しません。' (No matching content exists).

| Name | Status | Pid | Need To Restart | Command | Actions |
|-----------------|-------------------|-------|-----------------|------------|------------------|
| adminServer (*) | RUNNING(00:37:32) | 10821 | false | Start Stop | Delete Duplicate |
| server1 | RUNNING(00:31:18) | 11149 | false | Start Stop | Delete Duplicate |
| server2 | RUNNING(00:44:27) | 10337 | false | Start Stop | Delete Duplicate |
| server3 | RUNNING(00:44:27) | 10338 | false | Start Stop | Delete Duplicate |

2. 以下は、server1にcds1を登録する例です。「**Data Source**」項目の「cds1」のチェックボックスにチェックを入れて[確認]ボタンをクリックします。

[図 6.60] サーバーへのクラスター・データソースの登録(2)

The screenshot shows the 'Data Sources' configuration page. The left sidebar is the same as in the previous figure, but with 'Data Sources' selected. The main content area is titled 'Data Sources' and includes a message: 'サーバまたはクラスターで有効なデータソースを指定します。' (Specify the data source to be used on the server or cluster). Below this is a table with columns: Data Source, and a checkbox column. The table lists three data sources: ds2, ds1, and cds1, all of which have their checkboxes checked. Below the table is a message: 'サーバ、またはクラスターで有効なデータソースのIDを指定します。' (Specify the ID of the data source to be used on the server or cluster).

| Data Source | Check |
|-------------|-------------------------------------|
| ds2 | <input checked="" type="checkbox"/> |
| ds1 | <input checked="" type="checkbox"/> |
| cds1 | <input checked="" type="checkbox"/> |

3. cds1をserver1に登録することを最終反映するために[**Activate Changes**]ボタンをクリックします。

4. 以下のように、cds1がserver1に登録されたことが確認できます。

[図 6.61] サーバーへのクラスター・データソースの登録(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

4 Running

0 Shutdown

0 Suspended

0 Other

Server

HISTORY

ドメイン内で使用するJEUSサーバの詳細設定を定義します。

domain.xmlの設定を動的に適用しました。

domain.xml : ACTIVATED

servers.server.{? name == 'server1' }.dataSources : ACTIVATED

servers.server.{? name == 'server1' }.dataSources.ds1 : ACTIVATED

previous value : null, edited value : ds1, result value : ds1

servers.server.{? name == 'server1' }.dataSources.cds1 : ACTIVATED

previous value : null, edited value : cds1, result value : cds1

Basic Resource Engine

Basic Info Res Ref Naming Server System Thread Pool System Logging User Logging Tm Config

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

Auto Generated

サーバが動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスタが削除されると、該当するサーバも自動的に削除されます。

Name * server1

サーバ名です。

5. JNDIモニタリング情報を確認すると、server1のJNDIリポジトリにcds1とそのコンポーネント・データソースであるds1、ds2がバインドされていることが確認できます。cds1のJNDI名を特に入力していないため、データソースIDがJNDI名として使用されています。

[図 6.62] サーバーにJNDIバインドされたクラスター・データソースの確認

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

JNDI

Thread

Transaction

MBean

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

JNDI

HISTORY

JNDIのモニタリング情報を照会します。

The JNDI list on the server1

adminServer

server1

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| cds1 | jeus.jdbc.info.JDBCBindInfo | false |
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| ds2 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server2

server3

コンソール・ツールの使用

cds1のコンポーネント・データソースであるds1とds2もserver1と一緒に登録する必要があります。

以下は、**add-data-sources-to-server**を実行してds1、ds2、cds1をserver1に登録する例です。
add-data-sources-to-serverのより詳しい使用法は、『*JEUS リファレンスガイド*』の「4.2.3.2. add-data-sources-to-server」を参照してください。

```
[DAS]domain1.adminServer>adddstosvr -server server1 -ids ds1,ds2,cds1
Successfully performed the ADD operation for data sources to the server [server1].
Check the results using "add-data-sources-to-server -server server1"
```

上記のメッセージのように、ds1、ds2、cds1がserver1に登録されていることを、「**add-data-sources-to-server -server server1**」を実行して確認できます。

```
[DAS]domain1.adminServer>adddstosvr -server server1
Shows the current configuration.
Data sources registered in the server [server1].
=====
+-----+-----+
| data sources | ds1, ds2, cds1 |
+-----+-----+
=====
```

jndilistを実行してserver1のJNDIリポジトリにds1、ds2、cds1がバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+
| cds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+
=====
```

6.7.3. サーバーからのクラスター・データソースの削除

サーバーに登録されたクラスター・データソースも、一般データソースのようにサーバーから動的に削除することができます。

一般データソースと同様に、クラスター・データソースをサーバーから削除すると、クラスター・データソース情報がJNDIアンバインドされます。クラスター・データソース自体はコネクション・プールを持たず、実際のコネクション・プールはクラスター・データソースに属するコンポーネント・データソースから作成されるので、クラスター・データソースがサーバーから削除されるときは、一般データソースとは異なってコネクション・プールの破棄は発生しません。クラスター・データソースに属するコンポーネント・データソースのコネクション・プールが破棄されるのではないかとされるかも知れませんが、コンポーネント・データソースはそれぞれ独立したデータソースとしても機能する必要があり得るので、それらのJNDIアンバインドや破棄は発生しません。

WebAdminの使用

以下では、WebAdminを使ってサーバーからクラスター・データソースを削除する方法について説明します。

1. WebAdminの左のメニューで[Servers]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面で任意のサーバーを選択すると、そのサーバーからクラスター・データソースを動的に削除できる画面に移動します。

[図 6.63] サーバーからのクラスター・データソースの削除(1)

The screenshot displays the WebAdmin interface for managing servers. On the left, a sidebar menu includes options like Domain, Session, Clusters, Servers (selected), Applications, Security, Resources, Monitoring, and Console. The main content area is titled 'Servers' and contains a table with the following data:

| Name | Status | Pid | Need To Restart | Command | Actions |
|-----------------|-------------------|-------|-----------------|------------|------------------|
| adminServer (*) | RUNNING(00:37:32) | 10821 | false | Start Stop | Delete Duplicate |
| server1 | RUNNING(00:31:18) | 11149 | false | Start Stop | Delete Duplicate |
| server2 | RUNNING(00:44:27) | 10337 | false | Start Stop | Delete Duplicate |
| server3 | RUNNING(00:44:27) | 10338 | false | Start Stop | Delete Duplicate |

Below the table, there is a 'Server Templates' section with a header 'Name' and an 'Add' button. The message '該当する内容が存在しません。' (No content found) is displayed below this section.

2. 以下は、server1からcds1を削除する例です。「Data Source」項目で「cds1」のチェックボックスのチェックを解除し、[確認]ボタンをクリックします。

[図 6.64] サーバーからのクラスター・データソースの削除(2)

| Data Source | |
|-------------|-------------------------------------|
| ds2 | <input checked="" type="checkbox"/> |
| ds1 | <input checked="" type="checkbox"/> |
| cds1 | <input type="checkbox"/> |

3. cds1をserver1から削除することを最終反映するために[Activate Changes]ボタンをクリックします。
4. 以下のように、cds1がserver1から削除されていることが確認できます。

[図 6.65] サーバーからのクラスター・データソースの削除(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

4 Running

0 Shutdown

0 Suspended

Server

HISTORY

ドメイン内で使用するJEUSサーバの詳細設定を定義します。

domain.xmlの設定を動的に適用しました。

domain.xml : ACTIVATED

servers.server.{? name == 'server1' }.dataSources : ACTIVATED

servers.server.{? name == 'server1' }.dataSources.cds1 : ACTIVATED

previous value : cds1, edited value : null, result value : null

Basic Resource Engine

Basic Info Res Ref Naming Server System Thread Pool System Logging User Logging Tm Config

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

Auto Generated

サーバが動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスターが削除されると、該当するサーバも自動的に削除されます。

Name

server1

サーバ名です。

5. JNDIモニタリング情報を確認すると、server1のJNDIリポジトリからcds1がアンバインドされていることが確認できます。

【図 6.66】 サーバーからJNDIアンバインドされたクラスター・データソースの確認

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| ds2 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

cds1がserver1から削除された後も、ds1とds2は継続してserver1に登録された状態であり、これらはserver1で独立したデータソースとして機能することができます。しかし、cds1をサーバーに登録するときにcds1のコンポーネント・データソースのds1とds2と一緒に登録したように、cds1をサーバーから削除するときもcds1のコンポーネント・データソースのds1とds2と一緒に削除するのがデータソースの管理観点では直感的です。

コンソール・ツールの使用

コンソール・ツールで**remove-data-sources-from-server**を実行すると、クラスター・データソースをサーバーから動的に削除することができます。remove-data-sources-from-serverのより詳しい使用方法は、『JEUS リファレンスガイド』の「4.2.3.34. remove-data-sources-from-server」を参照してください。

以下は、remove-data-sources-from-serverを実行してcds1をserver1から削除する例です。

```
[DAS]domain1.adminServer>rmdsfromsvr -server server1 -ids cds1
Successfully performed the REMOVE operation for data sources from the server [server1].
Check the results using "remove-data-sources-from-server -server server1"
```

上記のメッセージのように、cds1がserver1から削除されたことを、「**remove-data-sources-from-server -server server1**」を実行して確認できます。

```
[DAS]domain1.adminServer>rmdsfromsvr -server server1
Shows the current configuration.
Data sources registered in the server [server1].
```



```
=====
+-----+-----+
| data sources | ds1, ds2 |
+-----+-----+
=====
```

cds1が削除され、ds1とds2は継続して登録されていることが確認できます。

jndilistを実行すると、server1のJNDIリポジトリからcds1がアンバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
=====
```

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

```
=====
```

6.7.4. クラスターへのクラスター・データソースの登録

クラスター・データソースも一般データソースのように、動的にクラスターに登録することができます。クラスターに登録されたクラスター・データソースは、クラスターに属するすべてのサーバーで有効です。つまり、クラスターに属するサーバーはクラスターに登録されたクラスター・データソースを自身が登録したクラスター・データソースのように使用することができます。クラスター・データソースをサーバーに登録するときのように、クラスター・データソースをクラスターに登録するときも、クラスター・データソースを構成するコンポーネント・データソースをクラスターと一緒に登録する必要があります。

本節では、cds1をcluster1に登録する手順について説明します。

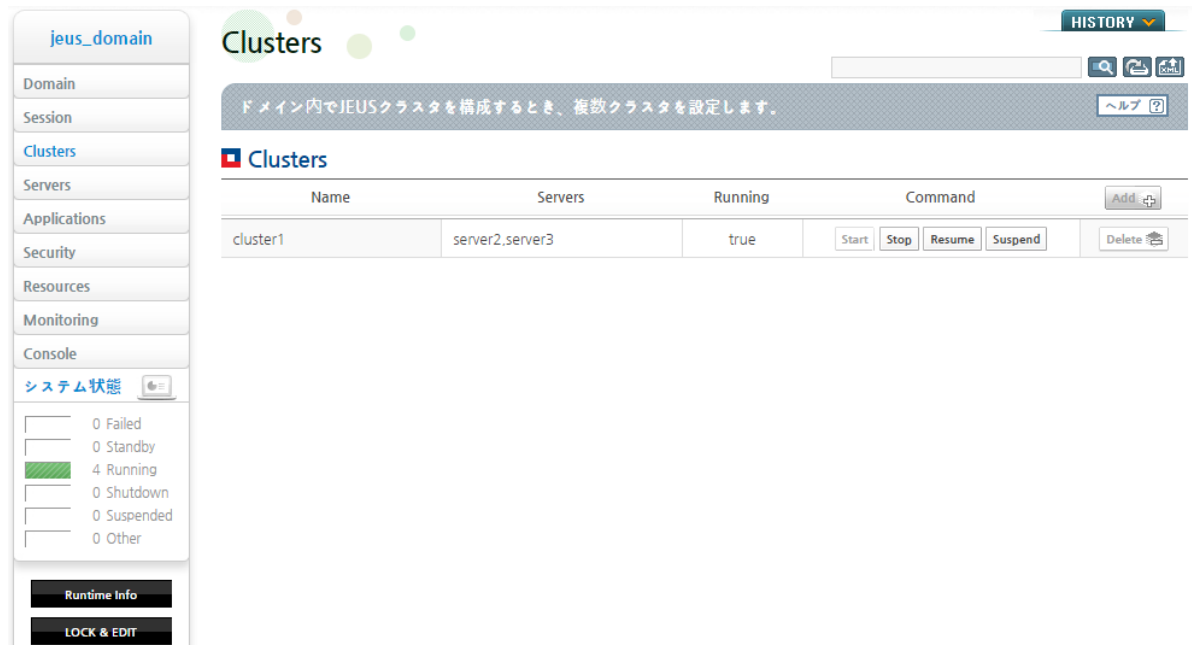
WebAdminの使用

cds1のコンポーネント・データソースであるds1とds2もcluster1と一緒に登録する必要があります。WebAdminを使ってデータソースをクラスターに登録する方法は、[「6.6.4. クラスターへのデータソースの登録」](#)を参照してください。以降の説明では、ds1、ds2がcluster1に登録されていることを仮定します。

以下では、WebAdminを使ってクラスターにクラスター・データソースを登録する方法について説明します。

1. WebAdminの左のメニューで[Clusters]を選択して[LOCK & EDIT]ボタンをクリックした後、画面で表示される任意のクラスターを選択すると、そのクラスターにクラスター・データソースを動的に登録できる画面に移動します。

[図 6.67] クラスターへのクラスター・データソースの登録(1)



2. 以下は、cluster1にcds1を登録する例です。「Data Source」項目で「cds1」のチェックボックスにチェックを入れて[確認]ボタンをクリックします。

[図 6.68] クラスターへのクラスター・データソースの登録(2)



3. cds1をcluster1に登録することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、cds1がcluster1に登録されていることが確認できます。

[図 6.69] クラスターへのクラスター・データソースの登録(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

4 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル

Domain

Server [もっと見る](#)

Cluster

HISTORY

クラスター構成のための詳細設定を行います。

domain.xmlの設定を動的に適用しました。

domain.xml : ACTIVATED

```
clusters.cluster.{? name == 'cluster1' }.dataSources : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - server1
previous value : null, edited value : cds1, result value : cds1
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - server2
previous value : null, edited value : cds1, result value : cds1
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - adminServer
previous value : null, edited value : cds1, result value : cds1
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - server3
previous value : null, edited value : cds1, result value : cds1
```

Basic Res Ref Session Cluster Config Jms Resource Lifecycle Invocation

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 [TIP](#)

Name * cluster1

クラスターの固有の名前を指定します。この名前はドメイン内で一意である必要があり、クラスターを構成する際、固有の識別子(ID)として使用されます。

Servers

Servers

クラスターに参加するサーバリストを設定します。

Server Name * [+](#)

☐ adminServer

☐ server1

☒ server2

☒ server3

クラスターに参加するサーバの名前を指定します。

5. cluster1にds1、ds2、cds1が登録されたので、cluster1に属するserver2とserver3はds1、ds2、cds1を自身が登録したデータソースのように使用することができます。

server2のJNDIリポジトリを確認すると、ds1、ds2、cds1がJNDIバインドされていることが確認できます。

【図 6.70】 サーバーにJNDIバインドされたクラスター・データソースの確認

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| cds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

コンソール・ツールの使用

cds1のコンポーネント・データソースであるds1とds2もcluster1と一緒に登録する必要があります。

以下は、コンソール・ツールで**add-data-sources-to-cluster**を実行してds1、ds2、cds1をcluster1に登録する例です。add-data-sources-to-clusterのより詳しい使用方法は、『JEUS リファレンスガイド』の「4.2.4.4. add-data-sources-to-cluster」を参照してください。

```
[DAS]domain1.adminServer>addstocluster -cluster cluster1 -ids ds1,ds2,cds1
Successfully performed the ADD operation for data sources to the cluster [cluster1].
Check the results using "add-data-sources-to-cluster -cluster cluster1"
```

上記のメッセージのように、ds1、ds2、cds1がcluster1に登録されたことを、「**add-data-sources-to-cluster -cluster cluster1**」を実行して確認できます。

```
[DAS]domain1.adminServer>addstocluster -cluster cluster1
Shows the current configuration.
The data sources registered in the cluster [cluster1].
=====
+-----+-----+
| data sources | ds1, ds2, cds1 |
```

```
+-----+-----+
=====
```

cluster1にds1、ds2、cds1が登録されたので、cluster1に属するserver2とserver3はds1、ds2、cds1を自身が登録したデータソースのように使用することができます。

jndilistを実行してserver2のJNDIリポジトリを確認すると、ds1、ds2、cds1がJNDIバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server2
The JNDI list on the server2
List of the context /
=====
+-----+-----+-----+
|      Name      |      Value      | Local Binding |
+-----+-----+-----+
| cds1           | jeus.jdbc.info.JDBCBindInfo | true         |
| ds1            | jeus.jdbc.info.JDBCBindInfo | true         |
| ds2            | jeus.jdbc.info.JDBCBindInfo | true         |
| JEUSMQ_DLQ     | jeus.jms.common.destination.JeusQueue | false        |
| mgmt           | jeus.jndi.JNSContext      | false        |
+-----+-----+-----+
=====
```

6.7.5. クラスターからのクラスター・データソースの削除

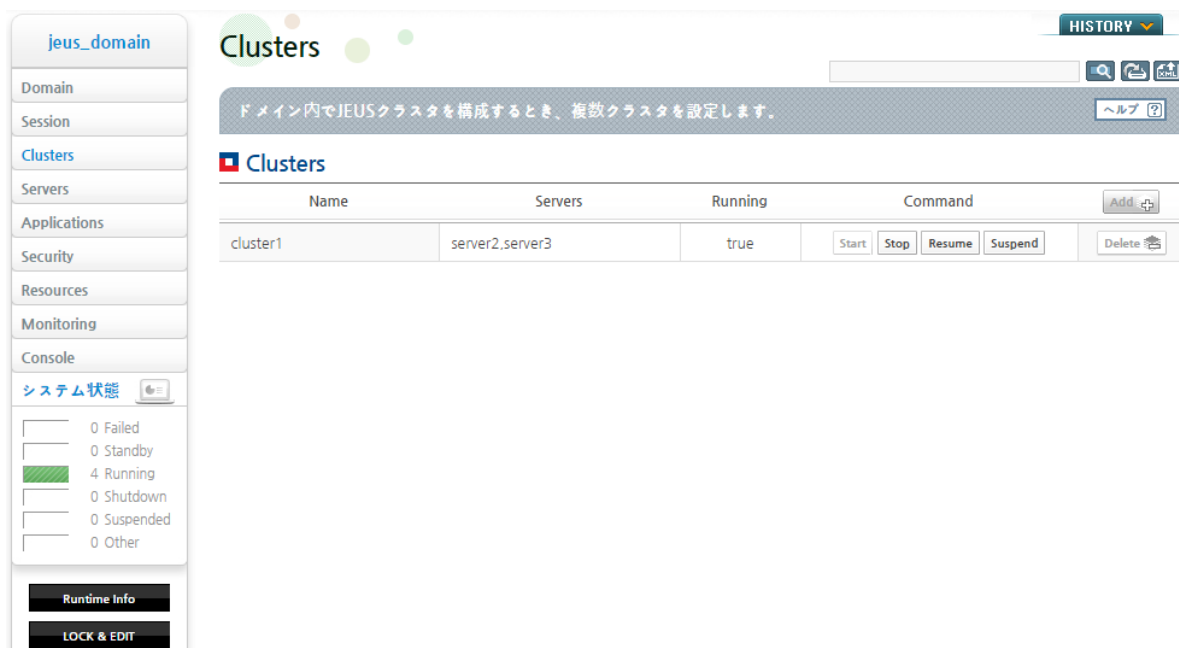
クラスターに登録されたクラスター・データソースも、一般データソースのようにクラスターから動的に削除することができます。クラスターに登録されたクラスター・データソースを削除すると、クラスター・データソースを使用していたクラスター内のすべてのサーバーはそのクラスター・データソースを使用できなくなります。つまり、クラスター内のサーバーたちは、クラスターからクラスター・データソースが削除されるときに、クラスター・データソース情報をJNDIアンバインドします。しかし、クラスター・データソースをサーバーから削除するときと同じく、コンポーネント・データソースのコネクション・プールの破棄は発生しません。

WebAdminの使用

以下では、WebAdminを使ってクラスターからクラスター・データソースを削除する方法について説明します。

- 1. WebAdminの左のメニューで**[Clusters]**を選択して**[LOCK & EDIT]**ボタンをクリックした後、画面で表示される任意のクラスターを選択すると、そのクラスターからクラスター・データソースを動的に削除できる画面に移動します。

【図 6.71】 クラスターからのクラスター・データソースの削除(1)



2. 以下は、cluster1からcds1を削除する例です。「Data Source」項目で「cds1」チェックボックスのチェックを解除し[確認] ボタンをクリックします。

【図 6.72】 クラスターからのクラスター・データソースの削除(2)



3. cds1をcluster1から削除することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、cds1がcluster1から削除されていることが確認できます。

[図 6.73] クラスターからのクラスター・データソースの削除(3)

The screenshot shows the Oracle JDeveloper interface for configuring a cluster. The left sidebar displays the project structure for 'jeus_domain', with 'Clusters' selected. The main area shows the 'Basic' tab for the 'cluster1' configuration. The 'Name' field is set to 'cluster1'. The 'Servers' section shows a list of servers: 'adminServer', 'server1', 'server2', and 'server3'. The 'server2' and 'server3' checkboxes are checked, indicating they are part of the cluster. The 'cds1' data source is shown as being removed from the cluster configuration.

jeus_domain

- Domain
- Session
- Clusters**
- Servers
- Applications
- Security
- Resources
- Monitoring
- Console

システム状態

- 0 Failed
- 0 Standby
- 4 Running
- 0 Shutdown
- 0 Suspended
- 0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を変更、追加、削除する機能です。

運用マニュアル

- Domain
- Server [もっと見る](#)

Cluster

クラスター構成のための詳細設定を行います。

domain.xml の設定を動的に適用しました。

domain.xml : ACTIVATED

```
clusters.cluster.{? name == 'cluster1' }.dataSources : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - server1
previous value : cds1, edited value : null, result value : null
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - server2
previous value : cds1, edited value : null, result value : null
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - adminServer
previous value : cds1, edited value : null, result value : null
clusters.cluster.{? name == 'cluster1' }.dataSources.cds1 : ACTIVATED - server3
previous value : cds1, edited value : null, result value : null
```

Basic | **Res Ref** | **Session Cluster Config** | **Jms Resource** | **Lifecycle Invocation**

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 [ヘルプ](#)

Name * cluster1
クラスターの固有の名前を指定します。この名前はドメイン内で一意である必要があり、クラスターを構成する際、固有の識別子(ID)として使用されます。

Servers

クラスターに参加するサーバリストを設定します。

Server Name *

- ☐ adminServer
- ☐ server1
- ☒ server2
- ☒ server3

クラスターに参加するサーバの名前を指定します。

5. cluster1からクラスター・データソースのcds1が削除されたので、cluster1に属するserver2とserver3はもうcds1を自身が登録したデータソースのように使用できなくなります。これは、server2とserver3のJNDIリポジトリからcds1がJNDIアンバインドされたことを意味します。

server2のJNDIリポジトリを確認すると、server2のJNDIリポジトリからcds1がアンバインドされたことが確認できます。

[図 6.74] サーバーからJNDIアンバインドされたクラスター・データソースの確認

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

cds1がcluster1から削除された後も、ds1とds2は継続してcluster1に登録された状態であり、これらはcluster1を構成するserver2とserver3で有効であるので、それぞれのサーバーで独立したデータソースとして機能することができます。しかし、cds1をクラスターに登録するときにcds1のコンポーネント・データソースであるds1とds2と一緒に登録したように、cds1をクラスターから削除するときもcds1のコンポーネント・データソースであるds1とds2と一緒に削除するのがデータソースの管理観点では直感的です。

コンソール・ツールの使用

コンソール・ツールで**remove-data-sources-from-cluster**を実行すると、クラスター・データソースをクラスターから動的に削除できます。remove-data-sources-from-clusterのより詳しい使用法は、『*JEUS リファレンスガイド*』の「4.2.4.25. remove-data-sources-from-cluster」を参照してください。

以下は、remove-data-sources-from-clusterを実行してcds1をcluster1から削除する例です。

```
[DAS]domain1.adminServer>rmdsfromcluster -cluster cluster1 -ids cds1
Successfully performed the REMOVE operation for data sources from the cluster [cluster1].
Check the results using "remove-data-sources-from-cluster -cluster cluster1"
```

remove-data-sources-from-cluster -cluster cluster1を実行すると、cluster1からcds1が削除され、ds1とds2は残っていることが確認できます。


```
[DAS]domain1.adminServer>rmclsfromcluster -cluster cluster1
Shows the current configuration.
The data sources registered in the cluster [cluster1].
=====
+-----+-----+
| data sources | ds1, ds2 |
+-----+-----+
=====
```

cluster1からクラスター・データソースのcds1が削除されたので、cluster1に属するserver2とserver3はもうcds1を自身が登録したデータソースのように使用することができません。これは、server2とserver3のJNDIリポジトリでcds1がJNDIアンバインドされたことを意味します。

jndilistを実行すると、server2のJNDIリポジトリからcds1がJNDIアンバインドされていることが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server2
The JNDI list on the server2
List of the context /
=====
+-----+-----+-----+
| Name | Value | Local Binding |
+-----+-----+-----+
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |
+-----+-----+-----+
=====
```

6.7.6. クラスターへのサーバーの追加

クラスターにサーバーを動的に追加すると、クラスターに登録されたクラスター・データソースも、クラスターに登録された一般データソースと同様に、クラスターに追加されたサーバーで有効になります。つまり、クラスターに追加されたサーバーは、クラスターに登録されたクラスター・データソースを自身が登録したデータソースのように使用することができます。

全体的なメカニズムは、「[6.6.6. クラスターへのサーバーの追加](#)」での説明と似ているので別途説明しません。詳細は該当節を参照してください。

6.7.7. クラスターからのサーバーの削除

クラスターからサーバーを動的に削除すると、クラスターに登録されたクラスター・データソースはクラスターに登録された一般データソースと同様に、クラスターから削除されたサーバーでもう有効ではなくなります。

つまり、クラスターから削除されたサーバーは、クラスターに登録されたすべてのクラスター・データソース情報をJNDIアンバインドします。

全体的なメカニズムは、「[6.6.7. クラスターからのサーバーの削除](#)」での説明と似ているので別途説明しません。詳細は該当節を参照してください。

6.7.8. クラスターの削除

クラスターをドメインから動的に削除すると、クラスターに登録されたクラスター・データソースはクラスターに登録された一般データソースと同様に、クラスターに属するサーバーでもう有効ではなくなります。つまり、削除されたクラスターに属するサーバーはクラスターに登録されたすべてのクラスター・データソース情報をJNDIアンバインドします。全体的なメカニズムは、「[6.6.8. クラスターの削除](#)」での説明と似ているので別途説明しません。詳しくは当該節を参照してください。

6.7.9. クラスター・データソースの削除

クラスター・データソースも一般データソースのようにドメインから動的に削除することができます。クラスター・データソースをドメインから削除すると、削除されたクラスター・データソースはそれを使用していたサーバーとクラスターで有効ではなくなります。つまり、削除されたクラスター・データソースを使用していたサーバーはそのデータソース情報をJNDIアンバインドします。

クラスター・データソースが削除されるときに、サーバーおよびクラスターに登録されたクラスター・データソースがどのように処理されるか例を通して説明します。以降の説明では、server1とcluster1にcds1が登録されていることを仮定します。

本節では、cds1をdomain1から削除する手順について説明します。

WebAdminの使用

現在、cds1はserver1とcluster1に登録されているので、cds1をドメインから削除するためには、まずWebAdminの[環境設定]メニューで**Delete Reference**チェックボックスにチェックを入れる必要があります。WebAdminの[環境設定]メニューの調整については『*JEUS WebAdminガイド*』の「2.3.1. ヘッダー領域」を参照してください。

以下では、WebAdminを使ってクラスター・データソースを削除する方法について説明します。以前登録したcds1をドメインから削除する例です。

1. WebAdminの左のメニューで[Resources] > [Data Source]を選択して[LOCK & EDIT]ボタンをクリックした後、画面で表示されるCluster Dsリストで[Delete]ボタンをクリックすると、クラスター・データソースを動的にドメインから削除することができます。

[図 6.75] クラスター・データソースの削除(1)

The screenshot shows the WebAdmin interface for 'jeus_domain'. The left sidebar contains a navigation menu with categories: Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. The 'Resources' category is expanded, showing 'DataSource' as the selected item. Below it are 'Mail Source', 'URL Source', 'Message Bridge', 'Custom Resource', 'External Source', 'External Resource', and 'Concurrency Utilities Resource'. The 'Monitoring' section shows 'システム状態' (System Status) with a bar chart and a 'Runtime Info' button. The 'Console' section shows 'LOCK & EDIT' buttons.

The main area displays the 'DataSource' configuration page. At the top, there is a 'DataSource' header with a search bar and a 'HISTORY' dropdown. Below the header is a blue banner with the text 'アプリケーションで使用できるデータソースを定義します。' (Define data sources that can be used by the application.) and a 'ヘルプ' (Help) button.

The 'Database' section contains a table with the following data:

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete Duplicate |
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete Duplicate |

The 'Cluster DS' section contains a table with the following data:

| DataSource ID | Export Name | |
|---------------|-------------|--------|
| cds1 | cds1 | Delete |

2. cds1をドメインから削除することを最終反映するために[Activate Changes]ボタンをクリックします。

3. 以下のように、cds1がドメインから削除されたことが確認できます。

[図 6.76] クラスター・データソースの削除(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

DataSource

Mail Source

URL Source

Message Bridge

Custom Resource

External Source

External Resource

Concurrency Utilities Resource

Monitoring

Console

システム状態

0 Failed

0 Standby

4 Running

0 Shutdown

0 Suspended

0 Other

DataSource

アプリケーションで使用できるデータソースを定義します。

domain.xmlの設定を動的に適用しました。

domain.xml : ACTIVATED

resources.dataSource.clusterDs : ACTIVATED

resources.dataSource.clusterDs.cds1 : ACTIVATED

resources.dataSource.clusterDs.cds1 : ACTIVATED - server1

previous value : jeus.xml.binding.jeusDD.ClusterDsType@74f940, edited value : null, result value : null

resources.dataSource.clusterDs.cds1 : ACTIVATED - server2

previous value : jeus.xml.binding.jeusDD.ClusterDsType@2adb13, edited value : null, result value : null

resources.dataSource.clusterDs.cds1 : ACTIVATED - adminServer

previous value : jeus.xml.binding.jeusDD.ClusterDsType@2f7263, edited value : null, result value : null

resources.dataSource.clusterDs.cds1 : ACTIVATED - server3

previous value : jeus.xml.binding.jeusDD.ClusterDsType@1181b59, edited value : null, result value : null

Database

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|--|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | <div>Delete</div> <div>Duplicate</div> |
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | <div>Delete</div> <div>Duplicate</div> |

Cluster DS

| DataSource ID | Export Name |
|---------------|-------------|
|---------------|-------------|

該当する内容が存在しません。

4. cds1がdomain1から削除されたので、cds1を登録して使用していたserver1と、cluster1に属してcluster1に登録されたcds1を使用することができたserver2とserver3はcds1を使用できなくなります。これは、server1、server2、server3のそれぞれのJNDIリポジトリからcds1がJNDIアンバインドされたことを意味します。

server1のJNDIリポジトリを確認すると、cds1がJNDIバインドされていないことが確認できます。

【図 6.77】 サーバーにJNDIアンバインドされたクラスター・データソースの確認(1)

JNDI

JNDIのモニタリング情報を照会します。

The JNDI list on the server1

adminServer

server1

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | false |
| ds2 | jeus.jdbc.info.JDBCBindInfo | false |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server2

server3

cluster1に属するserver2のJNDIリポジトリを確認すると、cds1がJNDIバインドされていないことが確認できます。

【図 6.78】 サーバーにJNDIアンバインドされたクラスター・データソースの確認(2)

JNDI

JNDIのモニタリング情報を照会します。

The JNDI list on the server2

adminServer

server1

server2

List of the context /

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

server3

コンソール・ツールの使用

コンソール・ツールで**remove-cluster-data-source**を実行すると、ドメインからクラスター・データソースを動的に削除することができます。remove-cluster-data-sourceのより詳しい使用方法は、『*JEUS リファレンスガイド*』の「4.2.11.7. remove-cluster-data-source」を参照してください。

以下は、remove-cluster-data-sourceを実行してcds1をdomain1から削除する例です。

```
[DAS]domain1.adminServer>rmcds -id cds1
Successfully performed the REMOVE operation for cluster data source [cds1] from domain.
=====
+-----+-----+-----+
| resources.dataSource.clusterDs          | MODIFY | ACTIVATED |
| servers.server.{? name == 'server1' }.dataSources | MODIFY | ACTIVATED |
| clusters.cluster.{? name == 'cluster1' }.dataSources | MODIFY | ACTIVATED |
+-----+-----+-----+
=====
Check the results using "remove-cluster-data-source"
```

上記の結果を見ると、remove-cluster-data-sourceを実行してcds1を削除するとき、cds1が登録されていたserver1とcluster1にも変化があることが予想できます。上記の結果は、cds1がdomain1から削除されると同時に、server1とcluster1に登録されていたcds1情報も削除されることを表しています。

このように、動的に設定を変更するコマンドを実行すると、それと関連して発生するすべての設定変更のランタイム反映の有無を確認できます。動的な設定変更コマンドのより詳しい使用方法は、『*JEUS ドメインガイド*』の「第3章 ドメインの設定変更」を参照してください。

上記のメッセージのように、**remove-cluster-data-source**を実行してcds1が削除されたことを確認できます。

```
[DAS]domain1.adminServer>rmcds
Shows the current configuration.
Data sources in domain
=====
+-----+-----+-----+
| ds1 | common data source |
| ds2 | common data source |
+-----+-----+-----+
=====
```

cds1がdomain1から削除されたので、cds1を登録して使用していたserver1と、cluster1に属していてcluster1に登録されたcds1を使用することができたserver2とserver3はcds1を使用できなくなります。これは、server1、server2、server3のそれぞれのJNDIリポジトリからcds1がJNDIアンバインドされたことを意味します。**jndilist**を実行してserver1のJNDIリポジトリを確認すると、cds1がJNDIバインドされていないことが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server1
The JNDI list on the server1
List of the context /
```

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

jndilistを実行してcluster1に属するserver2のJNDIリポジトリを確認すると、cds1がJNDIバインドされていないことが確認できます。

```
[DAS]domain1.adminServer>jndilist -server server2
The JNDI list on the server2
List of the context /
```

| Name | Value | Local Binding |
|------------|---------------------------------------|---------------|
| ds1 | jeus.jdbc.info.JDBCBindInfo | true |
| ds2 | jeus.jdbc.info.JDBCBindInfo | true |
| JEUSMQ_DLQ | jeus.jms.common.destination.JeusQueue | false |
| mgmt | jeus.jndi.JNSContext | false |

6.7.10. クラスター・データソース設定の変更

これまで、クラスター・データソース、サーバー、クラスターを追加、削除するときのデータソースの管理方法について説明しました。これからは、クラスター・データソース自体の設定変更について説明します。クラスター・データソースの設定はすべて動的変更が可能です。ここでは、クラスター・データソースを構成するコンポーネント・データソースの変更を例に挙げて説明します。

クラスター・データソースがドメインに存在しないとし、add-cluster-data-sourceを実行してコンポーネント・データソースds1、ds2を持つクラスター・データソースcds1をdomain1に追加します。以降の説明では、ds1、ds2をコンポーネント・データソースとして持つcds1がdomain1に追加されたことを仮定します。

本節では、cds1のコンポーネント・データソース・リストでds2を削除する手順について説明します。

WebAdminの使用

以下では、WebAdminを使ってクラスター・データソースの設定を変更する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [Data Source]を選択して[LOCK & EDIT]ボタンをクリックした後、表示される画面のCluster Dsリストで任意のクラスター・データソースを選択すると、そのクラスター・データソースの設定を動的に変更することができます。

[図 6.79] クラスター・データソース設定の変更(1)



The screenshot shows the 'DataSource' configuration page in WebAdmin. The left sidebar has a menu with 'Resources' expanded, showing 'DataSource' and other options like 'Mail Source', 'URL Source', etc. The main content area is titled 'DataSource' and has a 'Database' section. It contains a table with columns: DataSource ID, DataSource Class Name, DataSource Type, Command, and actions (Add, Delete, Duplicate). The table lists two data sources: ds1 and ds2, both of type 'CONNECTION_POOL_DATA_SOURCE'. Below this is a 'Cluster DS' section with a table with columns: DataSource ID and Export Name. It lists one cluster data source: cds1.

| DataSource ID | DataSource Class Name | DataSource Type | Command | |
|---------------|---|-----------------------------|---------|-------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |

| DataSource ID | Export Name | |
|---------------|-------------|--------|
| cds1 | cds1 | Delete |

2. 以下は、ds2をcds1のコンポーネント・データソース・リストから削除する例です。「Component Data Sources」項目で「ds2」のチェックボックスのチェックを解除して[確認]ボタンをクリックします。

[図 6.80] クラスター・データソース設定の変更(2)



The screenshot shows the 'Component Data Sources' dialog box. It has two lists: 'ds2' on the left and 'ds1' on the right. The 'ds2' list has a checkbox next to it. Below the lists is a button labeled '確認' (Confirm). The dialog box also has a title bar with 'Component Data Sources' and a star icon.

3. ds2をcds1のコンポーネント・データソース・リストから削除することを最終反映するために[Activate Changes]ボタンをクリックします。

4. 以下のように、ds2がcds1のコンポーネント・データソース・リストから削除されていることが確認できます。

【図 6.81】 クラスター・データソース設定の変更(3)

The screenshot shows the JBoss console interface for configuring data sources. The left sidebar has a navigation menu with 'Resources' selected. The main area is titled 'DataSource' and shows a configuration page for 'jeus_domain'. It displays a list of data sources with columns: DataSource ID, DataSource Class Name, DataSource Type, Command, and Action. The table lists ds1 and ds2. ds2 is highlighted, indicating it is the selected data source. Below the table, the 'Cluster DS' section shows the mapping of ds1 to cds1.

| DataSource ID | DataSource Class Name | DataSource Type | Command | Action |
|---------------|---|-----------------------------|---------|-------------------|
| ds1 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |
| ds2 | oracle.jdbc.pool.OracleConnectionPoolDataSource | CONNECTION_POOL_DATA_SOURCE | Test | Delete, Duplicate |

| DataSource ID | Export Name | Action |
|---------------|-------------|--------|
| cds1 | cds1 | Delete |

コンソール・ツールの使用

コンソール・ツールで**modify-cluster-data-source**を実行すると、クラスター・データソース設定を動的に変更できます。

helpを実行するか、『*JEUS リファレンスガイド*』の「4.2.11.8. modify-cluster-data-source」を参照すると、modify-cluster-data-sourceにより変更できるすべての設定をオプションの名前から類推して確認できます。動的変更が可能な設定のオプションにはタグが付けられるため、どの設定が動的に変更できるかも分かります。

以下は、modify-cluster-data-sourceを実行してds2をcds1のコンポーネント・データソース・リストから削除する例です。

```
[DAS]domain1.adminServer>modifycds -id cds1 -cds ds1
Successfully performed the MODIFY operation for configuration of the cluster data source [cds1].
Check the results using "modify-cluster-data-source -id cds1"
```

cds1のコンポーネント・データソース・リストからds2が削除されたことを、「**modify-cluster-data-source-id cds1**」を実行して他の設定値と一緒に確認できます。

```
[DAS]domain1.adminServer>modifycds -id cds1
Shows the current configuration.
configuration of the cluster data source [cds1]
=====
+-----+-----+
| id                | cds1 |
| export-name       | cds1 |
| load-balance      | false|
| is-pre-conn       | false|
| use-failback      | true |
| component-data-sources | ds1  |
+-----+-----+
=====
```

6.7.11. クラスター・データソース設定の確認

本節では、WebAdminとコンソール・ツールを使用して、クラスター・データソースの設定を確認する方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってクラスター・データソースの設定を確認する方法について説明します。

1. WebAdminの左のメニューで[Resources] > [Data Source]メニューを選択した後、表示される画面の **Cluster Ds** リストで任意のクラスター・データソースを選択すると、そのクラスター・データソースの設定を確認できます。

[図 6.82] クラスター・データソース設定の確認(1)

The screenshot shows the WebAdmin interface for configuring data sources. On the left, the 'Resources' menu is expanded, and 'DataSource' is selected. The main content area is titled 'DataSource' and includes a search bar and a 'HISTORY' button. Below this, there's a section for 'Database' with a table listing data sources 'ds1' and 'ds2'. The 'Cluster DS' section is also visible, showing a table with 'DataSource ID' and 'Export Name' columns, containing the entry 'cds1'.

2. 以下は、cds1の設定を確認する例です。

【図 6.83】 クラスター・データソース設定の確認(2)

The screenshot displays the Oracle JDeveloper console for a domain named 'jeus_domain'. The left-hand navigation pane shows the 'Resources' section expanded, with 'DataSource' selected. The main workspace shows the configuration for a 'Cluster DS'. The configuration is as follows:

| Property | Value |
|------------------------|------------------------------|
| Data Source Id | cds1 |
| Export Name | cds1 |
| Data Source Selector | foo.bar.MyDataSourceSelector |
| Load Balance | false |
| Component Data Sources | ds2, ds1 |
| Is Pre Conn | false |
| Use Failback | true |
| Xa Affinity | true |
| Ons Support | false |
| Ons Config | nodes=host1:6200,host2:6200 |

コンソール・ツールの使用

コンソール・ツールで**list-cluster-data-sources**コマンドを実行すると、ドメインに存在するすべてのクラスター・データソースのリストを確認できます。

以下は、**list-cluster-data-sources**を実行してドメインに存在するすべてのクラスター・データソース・リストを確認する例です。

```
[DAS]domain1.adminServer>ls cds
The list of cluster data sources
=====
+-----+-----+-----+
| Data Source ID | JNDI Export Name | Component Data Sources |
+-----+-----+-----+
```

| | | |
|------|------|-------|
| cds1 | cds1 | [ds1] |
|------|------|-------|

特定のクラスター・データソースのIDをidオプション値として入力すると、そのクラスター・データソースの全体設定を詳細に照会できます。以下は、**list-cluster-data-sources**を実行してcds1の詳細設定を確認する例です。

```
[DAS]domain1.adminServer>lscds -id cds1
The configuration of cluster data source [cds1]
```

| Configuration Name | Configuration Value |
|------------------------|---------------------|
| id | cds1 |
| export-name | cds1 |
| load-balance | false |
| is-pre-conn | false |
| use-failback | true |
| component-data-sources | [ds1] |

6.8. JDBCコネクション・プールのモニタリング

本節では、サーバーに存在するコネクション・プールのモニタリング方法を例を通して説明します。理解をより深めるためには、JEUSのドメイン構造やドメインでのデータソースおよびコネクション・プールの管理構造などについて理解している必要があります。JEUSのドメイン構造については、『[JEUSドメインガイド](#)』の「[1.3. 構成要素](#)」を参照してください。ドメインでのデータソースおよびコネクション・プールの管理構造については、『[6.3. データソースとコネクション・プールの管理](#)』を参照してください。

JDBCコネクション・プールのモニタリングはコンソール・ツールまたはWebAdminを使って実行することができ、本節では両方の方法を説明します。コンソール・ツールを使った方法を説明するときは、便宜上縮約されたコマンドおよびオプションの名前を使用します。縮約コマンドおよびオプション名とコマンドの詳しい使用方法については、コンソール・ツールでコマンド名を引数としてhelpを実行するか、『[JEUS リファレンスガイド](#)』の「[4.2.12. コネクション・プールのモニタリングおよび制御コマンド](#)」を参考して確認できます。

説明をするために必要な事前作業を行います。まずdomain1という名前のドメインとadminServerという名前のDASを作成します。DASを起動した後、server1、server2という名前を持つ2つのMSをdomain1に追加して起動します。そして、ds1、ds2というデータソースIDを持つデータソースをdomain1に追加した後、server1にはds1とds2の両方を登録し、server2にはds1だけを登録します。最後に、server1とserver2にds1のコネクション・プールをそれぞれ作成します。

ドメインおよびサーバーの設定方法については、『[JEUSドメインガイド](#)』の「[第2章ドメインの作成](#)」および「[第2章 JEUSの設定](#)」を、データソースの設定方法については、『[6.6. データソース関連設定の動的変更](#)』を参

照してください。コネクション・プールの作成方法については、「6.9. JDBCコネクション・プールの制御」を参照してください。以降の説明では、上記の事前作業がすべて完了したことを仮定します。

参考

データソースが登録される対象は究極的にはサーバーです。サーバー自身がデータソースを登録する場合はもちろん、クラスターがデータソースを登録する場合も、クラスターに登録されたデータソースは実際にはクラスターに属するサーバーに意味があります。クラスターはサーバーを構成する論理的なグループであるだけで、実際にデータソースを利用してコネクション・プールを作成して使用する主体はサーバーであるからです。このような理由で、コネクション・プールのモニタリング単位はサーバーです。

6.8.1. JDBCコネクション・プール・リストの確認

本節では、コンソール・ツールとWebAdminを使ってJDBCコネクション・プール・リストを確認する方法について説明します。

WebAdminの使用

以下は、WebAdminでserver1でのコネクション・プールのリストを確認する例です。

1. WebAdminの左のメニューで[Monitoring] > [Connection Pools]を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールのリストが確認できます。

[図 6.84] コネクション・プール・リストの確認(1)

The screenshot shows the 'Connection Pool' monitoring interface in WebAdmin. The left sidebar contains a menu with 'Monitoring' selected. The main content area is titled 'Connection Pool' and shows a list of servers: 'adminServer' and 'server1'. Below this, a message states: 'サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。'. A table titled 'The connection pool information on the server [server1].' displays the following data:

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled | |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|------|
| ds1 * | ds1 | 10 | 50 | 0 | 0 | 0 | 0 | false | false | stmt |
| ds2 | ds2 | 10 | 50 | 0 | 10 | 0 | 10 | false | true | stmt |

Below the table, a message states: '* : has not been created, total = active + idle + disposable'. At the bottom, there are sections for 'server2' and 'server3'.

server1がds1とds2の両方を登録し、ds1とds2のコネクション・プールが存在することが確認できます。事前作業の結果どおり、ds1コネクション・プールは作成された状態で、ds2コネクション・プールは作成されていない状態であることが確認できます。コラムは、コネクション・プールID、コネクションの最小値、コネクション・プールの最大値、アクティブ・コネクションの数、アイドル・コネクションの数、使い捨てコネクション

の数、コネクションの総数、コネクションが不足する場合の待機の有無、コネクション・プールの有効の有無を表示します。

2. 以下は、server2でのコネクション・プールのリストを確認する例です。

【図 6.85】コネクション・プール・リストの確認(2)

The screenshot shows the JBoss JMX console interface. On the left is a navigation tree with 'jeus_domain' at the top, followed by 'Domain', 'Session', 'Clusters', 'Servers', 'Applications', 'Security', 'Resources', and 'Monitoring'. Under 'Monitoring', there is a list of sub-items: 'Thread', 'Transaction', 'MBean', 'JNDI', 'Web', 'Servers', 'JMS', 'Connection Pool' (which is highlighted), and 'EJB Timer'. The main content area is titled 'Connection Pool' and has a 'HISTORY' dropdown. Below the title is a message: 'サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。' (Control the JDBC/JCA connection pools created on the server and query the runtime status). Below this is a list of servers: 'adminServer', 'server1', and 'server2'. A red square icon indicates an error or warning, followed by the text 'The connection pool information on the server [server2].'. Below this is a table with the following columns: 'Connection Pool ID', 'JNDI Export Name', 'Min', 'Max', 'Active', 'Idle', 'Disposable', 'Total', 'Wait', and 'Enabled'. The table contains one row for 'ds1' with values: 'ds1', '10', '50', '0', '10', '0', '10', 'false', and 'true'. Below the table is a message: '* : has not been created, total = active + idle + disposable'. At the bottom, there is a section for 'server3' which is currently empty.

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 10 | 50 | 0 | 10 | 0 | 10 | false | true |

server2はds1だけを登録し、ds1のコネクション・プールだけが存在するので、事前作業の結果どおり作成されていることが分かります。

コンソール・ツールの使用

コンソール・ツールで**connection-pool-info**を実行すると、サーバーに存在するすべてのコネクション・プールのリストが確認できます。connection-pool-infoはJDBCコネクション・プールだけでなく、JCAコネクション・プールに対しても同じ機能を実行するため、JDBCコネクション・プールに対して使用するためには-jdbcオプションを設定します。connection-pool-infoのより詳しい使用方法是、『JEUS リファレンスガイド』の「4.2.12.7. connection-pool-info」を参照してください。

以下は、connection-pool-infoを実行してserver1に存在するすべてのJDBCコネクション・プールのリストを確認する例です。

```
[DAS]domain1.adminServer>cpinfo -server server1 -jdbc
The connection pool information on the server [server1].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Connection | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
| Pool ID   |     |     |         |     |             |       |      |          |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

| | | | | | | | | |
|-------|---|----|---|---|---|---|-------|-------|
| ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server1がds1とds2の両方を登録しているので、ds1とds2のコネクション・プールが存在することが確認できます。事前作業の結果どおり、ds1コネクション・プールは作成され、ds2コネクション・プールは作成されていないことが確認できます。コラムは、コネクション・プールID、コネクションの最小値、コネクション・プールの最大値、アクティブ・コネクションの数、アイドル・コネクションの数、使い捨てコネクションの数、コネクションの総数、コネクションが不足する場合の待機の有無、コネクション・プールの有効の有無を表示します。

以下のように、activeオプションを使用すると、現在作成されたコネクション・プールのリストだけ確認できます。

```
[DAS]domain1.adminServer>cpinfo -server server1 -active -jdbc
```

The connection pool information on the server [server1].

| | | | | | | | | |
|--------------------|-----|-----|--------|------|------------|-------|-------|---------|
| Connection Pool ID | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
| ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |

* : has not been created, total = active + idle + disposable

また、jndiオプションを使用すると、データソースのJNDI名も一緒に表示します。

```
[DAS]domain1.adminServer>cpinfo -server server1 -jndi -jdbc
```

The connection pool information on the server [server1].

| | | | | | | | | | |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| Connection Pool ID | JNDI export name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2に存在するすべてのコネクション・プールのリストも以下のように確認できます。

```
[DAS]domain1.adminServer>cpinfo -server server2 -jdbc
The connection pool information on the server [server2].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Connection | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
| Pool ID   |     |     |         |      |              |       |      |         |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ds1 *     | 2   | 30  | 0       | 0    | 0           | 0     |      | false  |
+-----+-----+-----+-----+-----+-----+-----+-----+

* : has not been created, total = active + idle + disposable
=====
```

server2はds1だけを登録し、ds1のコネクション・プールだけが存在するので、事前作業の結果どおり作成されていることが分かります。

6.8.2. 特定のJDBCコネクション・プールの詳細情報の確認

本節では、コンソール・ツールとWebAdminを使って特定のJDBCコネクション・プールの詳細情報を確認する方法について説明します。

WebAdminの使用

以下では、WebAdminを使って特定のコネクション・プールの詳細情報を確認する方法について説明します。

1. WebAdminの左のメニューで**[Monitoring] > [Connection Pools]**を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールのリストが確認できます。特定のコネクション・プールのIDをクリックすると、そのコネクション・プールの詳細情報が確認できます。

以下は、server1でds1コネクション・プールの詳細情報を確認する例です。

[図 6.86] 特定のコネクション・プールの詳細情報の確認(1)

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 10 | 50 | 0 | 10 | 0 | 10 | false | true |
| ds2 * | ds2 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

2. まだ一度も使われていない10のコネクションが230秒ほどアイドル状態のままであることが確認できます。
10のコネクションは両方とも使い捨てコネクションではありません。

[図 6.87] 特定のコネクション・プールの詳細情報の確認(2))

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-2 | idle | 230.63 | 0 | pooled |
| ds1-1 | idle | 230.679 | 0 | pooled |
| ds1-6 | idle | 230.468 | 0 | pooled |
| ds1-7 | idle | 230.429 | 0 | pooled |
| ds1-9 | idle | 230.351 | 0 | pooled |
| ds1-5 | idle | 230.505 | 0 | pooled |
| ds1-3 | idle | 230.58 | 0 | pooled |
| ds1-10 | idle | 230.313 | 0 | pooled |
| ds1-8 | idle | 230.388 | 0 | pooled |
| ds1-4 | idle | 230.543 | 0 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 10 | 50 | 0 | 10 | 0 | 10 | false | true |
| ds2 * | ds2 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |

システム状態

0 Failed

0 Standby

4 Running

コンソール・ツールの使用

コンソール・ツールで**connection-pool-info**を実行すると、サーバーに存在する特定の接続・プールの詳細情報が確認できます。connection-pool-infoはJDBC接続・プールだけでなく、JCA接続・プールに対しても同じ機能を実行するので、JDBC接続・プールに対して使用したい場合は-jdbcオプションを設定します。connection-pool-infoのより詳しい使用方法は、『*JEUS リファレンスガイド*』の「4.2.12.7. connection-pool-info」を参照してください。

以下は、connection-pool-infoを実行してserver1に存在するds1接続・プールの詳細情報を確認する例です。

```
[DAS]domain1.adminServer>cpinfo -server server1 -id ds1 -jdbc
Information about connections in the server [server1]'s connection pool [ds1].
=====
+-----+-----+-----+-----+-----+
| Connection ID | State | State Time(sec) | Use Count | Type |
+-----+-----+-----+-----+-----+
| ds1-1         | idle  | 965.837         | 0         | pooled |
| ds1-2         | idle  | 965.806         | 0         | pooled |
+-----+-----+-----+-----+-----+
=====
```

まだ一度も使われていない2つの接続が965秒ほどアイドル状態に留まっていることが確認できます。2つの接続は両方とも使い捨て接続ではありません。

以下のように、-tオプションを使用すると、接続を所有するスレッドの名前も一緒に確認できます。

```
[DAS]domain1.adminServer>cpinfo -server server1 -id ds1 -t -jdbc
Information about connections in the server [server1]'s connection pool [ds1].
=====
+-----+-----+-----+-----+-----+-----+
| Connection ID | State | State Time(sec) | Use Count | Type | Thread name |
+-----+-----+-----+-----+-----+-----+
| ds1-1         | active | 1105.954        | 1         | pooled | http-w1     |
| ds1-2         | idle  | 1105.923        | 0         | pooled |             |
+-----+-----+-----+-----+-----+-----+
=====
```

6.9. JDBC接続・プールの制御

本節では、サーバーに存在する接続・プールの制御方法について説明します。理解をより深めるためには、JEUSのドメイン構造やドメインでのデータソースおよび接続・プールの管理構造などを理解する必要があります。JEUSのドメイン構造については、『*JEUS ドメインガイド*』の「1.3. 構成要素」を参照してください。ドメインでのデータソースおよび接続・プールの管理構造については、[「6.3. データソースと接続・プールの管理」](#)を参照してください。

JDBCコネクション・プールの制御は、コンソール・ツールまたはWebAdminを使って行うことができ、本節では両方の方法を説明します。

説明を行なうために必要な事前作業を行います。まずdomain1という名前のドメインとadminServerという名前のDASを作成します。DASを起動した後、server1、server2という名前を持つ2つのMSをdomain1に追加して起動します。そして、ds1、ds2というデータソースIDを持つデータソースをdomain1に追加した後、server1、server2にそれぞれds1とds2の両方を登録します。ドメインおよびサーバーの設定方法については『*JEUS ドメインガイド*』の「第2章ドメインの作成」および「[第2章 JEUSの設定](#)」を参照し、データソースの設定方法については「[6.6. データソース関連設定の動的変更](#)」を参照してください。

参考

コネクション・プールの制御後の結果については、別途例を挙げて説明しません。「[6.8. JDBCコネクション・プールのモニタリング](#)」を参照して結果を確認してください。

6.9.1. コネクション・プールの作成

本節では、コンソール・ツールとWebAdminを使ってコネクション・プールを作成する方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってコネクション・プールを作成する方法について説明します。

1. WebAdminの左のメニューで[Monitoring] > [Connection Pools]を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールを作成することができます。ds1のコネクション・プールを作成するためにds1をクリックします。

[図 6.88] コネクション・プールの作成(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 * | ds1 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |
| ds2 * | ds2 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

2. ds1のコネクション・プールのリストが照会されます。コネクション・プールを作成するために[create]ボタンをクリックします。

[図 6.89] コネクション・プールの作成(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|------|
|---------------|-------|-----------------|-----------|------|

該当する内容が存在しません。

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 * | ds1 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |
| ds2 * | ds2 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

3. ds1コネクション・プールが作成されたことが確認できます。

【図 6.90】コネクション・プールの作成(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 10 | 50 | 0 | 10 | 0 | 10 | false | true |
| ds2 * | ds2 | 10 | 50 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

コンソール・ツールの使用

コンソール・ツールで**create-connection-pool**を実行すると、サーバーにデータソースのコネクション・プールを作成できます。create-connection-poolのより詳しい使用方法は、『JEUS リファレンスガイド』の「4.2.12.1. create-connection-pool」を参照してください。

以下は、create-connection-poolを実行してserver1にds1コネクション・プールを作成する例です。

```
[DAS]domain1.adminServer>createcp -server server1 -id ds1
Servers that successfully created a connection pool : server1
Servers that failed to create a connection pool : none.
```

serverオプション値を指定しない場合、データソースが登録されたすべてのサーバーを対象にコネクション・プールが作成されます。

以下は、create-connection-poolを実行して、ds2が登録されたserver1、server2にそれぞれds2コネクション・プールを作成する例です。

```
[DAS]domain1.adminServer>createcp -id ds2
Servers that successfully created a connection pool : server1, server2
Servers that failed to create a connection pool : none.
```

control-connection-poolを実行する方法でもJDBCコネクション・プールを作成できます。control-connection-poolのより詳しい使用方法は、『JEUS リファレンスガイド』の「4.2.12.6. control-connection-pool」を参照してください。

以下は、control-connection-poolを実行してserver1にds1コネクション・プールを作成する例です。

```
[DAS]domain1.adminServer>ctrlcp -server server1 -id ds1 -create
Servers that successfully created a connection pool : server1
Servers that failed to create a connection pool : none.
```

6.9.2. コネクション・プールの無効化

本節では、コンソール・ツールとWebAdminを使ってコネクション・プールを無効にする方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってコネクション・プールを無効にする方法について説明します。

1. WebAdminの左のメニューで**[Monitoring] > [Connection Pools]**を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールを無効にできます。ds1のコネクション・プールを無効にするために、ds1をクリックします。

[図 6.91] コネクション・プールの無効化(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

Connection Pool

HISTORY

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

2. ds1のコネクション・リストが照会されると、[disable]ボタンをクリックします。

【図 6.92】コネクション・プールの無効化(2)

The screenshot shows the 'Connection Pool' management page. On the left is a navigation menu with 'jeus_domain' at the top and various system components listed below, including 'Monitoring' which is expanded. The main content area is titled 'Connection Pool' and contains a description: 'サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。'. Below this, there are sections for 'Information about connections in the server [server1]'s connection pool [ds1]' and 'The connection pool information on the server [server1]'. The first table shows connection details for ds1, with buttons 'Enable', 'Shrink', 'Disable', 'Refresh', and 'Create' at the bottom right. The second table shows pool statistics for ds1 and ds2, with a 'stmt' button for each row. A note at the bottom states: '* : has not been created, total = active + idle + disposable'.

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-10 | idle | 93.408 | 0 | pooled |
| ds1-9 | idle | 93.444 | 0 | pooled |

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

3. ds1コネクション・プールが無効になっていることが確認できます。

【図 6.93】コネクション・プールの無効化(3)

This screenshot is similar to the previous one, showing the 'Connection Pool' management page. The 'Disable' button has been clicked, and the 'Enabled' status for the 'ds1' pool in the second table is now 'false'. The 'ds1-10' and 'ds1-9' connections in the first table now have state times of 130.331 and 130.367 respectively. The 'stmt' button for the 'ds1' row is still present.

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-10 | idle | 130.331 | 0 | pooled |
| ds1-9 | idle | 130.367 | 0 | pooled |

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | false |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

コンソール・ツールの使用

コンソール・ツールで**disable-connection-pool**を実行すると、サーバーに存在するコネクション・プールを無効化できます。disable-connection-poolのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.12.3. disable-connection-pool」を参照してください。

以下は、disable-connection-poolを実行してserver1に作成されたds1コネクション・プールを無効にする例です。

```
[DAS]domain1.adminServer>disablecp -server server1 -id ds1  
Servers that successfully disabled a connection pool : server1  
Servers that failed to disable a connection pool : none.
```

serverオプション値を指定しない場合、データソースが登録されたすべてのサーバーを対象にコネクション・プールの無効化が試みられます。

以下は、disable-connection-poolを実行してds2が登録されたserver1、server2に対してそれぞれds2コネクション・プールの無効化を実行する例です。

```
[DAS]domain1.adminServer>disablecp -id ds2  
Servers that successfully disabled a connection pool : server1, server2  
Servers that failed to disable a connection pool : none.
```

control-connection-poolを実行する方法でもJDBCコネクション・プールを無効にできます。control-connection-poolのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.12.6. control-connection-pool」を参照してください。

以下は、control-connection-poolを実行して、server1に作成されたds1コネクション・プールを無効にする例です。

```
[DAS]domain1.adminServer>ctrlcp -server server1 -id ds1 -disable  
Servers that successfully disabled a connection pool : server1  
Servers that failed to disable a connection pool : none.
```

6.9.3. コネクション・プールの有効化

本節では、コンソール・ツールとWebAdminを使ってコネクション・プールを有効にする方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってコネクション・プールを有効にする方法について説明します。

1. WebAdminの左のメニューで[Monitoring] > [Connection Pools]を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールを有効にできます。

【図 6.94】コネクション・プールの有効化(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

Connection Pool

HISTORY

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | false |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

2. 以下は、ds1のコネクション・プールを有効にする例です。ds1を選択した後、[enable]ボタンをクリックします。

【図 6.95】コネクション・プールの有効化(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

Connection Pool

HISTORY

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-10 | idle | 214.908 | 0 | pooled |
| ds1-9 | idle | 214.944 | 0 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | false |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

3. ds1コネクション・プールが有効になっていることが確認できます。

【図 6.96】コネクション・プールの有効化(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-10 | idle | 241.272 | 0 | pooled |
| ds1-9 | idle | 241.308 | 0 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

コンソール・ツールの使用

コンソール・ツールで**enable-connection-pool**を実行すると、サーバーに存在するコネクション・プールを有効にできます。enable-connection-poolのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.12.2. enable-connection-pool」を参照してください。

以下は、enable-connection-poolを実行して、server1に作成されたds1コネクション・プールを有効にする例です。

```
[DAS]domain1.adminServer>enablecp -server server1 -id ds1
Servers that successfully enabled a connection pool : server1
Servers that failed to enable a connection pool : none.
```

serverオプション値を指定しない場合、データソースが登録されたすべてのサーバーを対象にコネクション・プールの有効化が試みられます。

以下は、enable-connection-poolを実行して、ds2が登録されたserver1、server2に対してそれぞれds2コネクション・プールを有効にする例です。

```
[DAS]domain1.adminServer>enablecnp -id ds2
Servers that successfully enabled a connection pool : server1, server2
Servers that failed to enable a connection pool : none.
```

control-connection-poolを実行する方法でもJDBCコネクション・プールを有効にできます。
control-connection-poolのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.12.6. control-connection-pool」を参照してください。

以下は、control-connection-poolを実行して、server1に生成されたds1コネクション・プールを有効にする例です。

```
[DAS]domain1.adminServer>ctrlcnp -server server1 -id ds1 -enable
Servers that successfully enabled a connection pool : server1
Servers that failed to enable a connection pool : none.
```

6.9.4. コネクション・プールのコネクションの切り替え

本節では、コンソール・ツールとWebAdminを使ってコネクション・プールのコネクションを切り替える方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってコネクション・プールのコネクションを切り替える方法について説明します。

1. WebAdminの左のメニューで**[Monitoring] > [Connection Pools]**を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールのコネクションを切り替えることができます。本例では、ds1のコネクションを切り替えるので、ds1をクリックします。

[図 6.97] コネクション・プールのコネクションの切り替え(1)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

2. ds1コネクション・プールのコネクションのリストが照会されます。[refresh]ボタンをクリックします。

[図 6.98] コネクション・プールのコネクションの切り替え(2)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-10 | idle | 292.924 | 0 | pooled |
| ds1-9 | idle | 292.96 | 0 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

3. ds1コネクション・プールのコネクションが切り替えられたことが確認できます。

[図 6.99] コネクション・プールのコネクションの切り替え(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-11 | idle | 3.034 | 0 | pooled |
| ds1-12 | idle | 2.987 | 0 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

コンソール・ツールの使用

コンソール・ツールで**refresh-connection-pool**を実行すると、サーバーに存在するコネクション・プールのコネクションを新しいコネクションに切り替えられます。refresh-connection-poolのより詳しい使用方法は、『JEUS リファレンスガイド』の「4.2.12.4. refresh-connection-pool」を参照してください。

以下は、refresh-connection-poolを実行して、server1に生成されたds1コネクション・プールのコネクションを新しいコネクションに切り替える例です。

```
[DAS]domain1.adminServer>refreshcp -server server1 -id ds1
Servers that successfully refreshed a connection pool : server1
Servers that failed to refresh a connection pool : none.
```

serverオプション値を指定していない場合、データソースが登録されたすべてのサーバーを対象にコネクション・プールのコネクションの切り替えが試みられます。

以下は、refresh-connection-poolを実行して、ds2が登録されたserver1、server2に対してそれぞれds2コネクション・プールのコネクションを新しいコネクションに切り替える例です。

```
[DAS]domain1.adminServer>refreshcp -id ds2
Servers that successfully refreshed a connection pool : server1, server2
Servers that failed to refresh a connection pool : none.
```

control-connection-poolを実行する方法でもJDBCコネクション・プールのコネクションを新しいコネクションに切り替えられます。control-connection-poolのより詳しい使用法は、『*JEUS リファレンスガイド*』の「4.2.12.6. control-connection-pool」を参照してください。

以下は、control-connection-poolを実行して、server1に生成されたds1コネクション・プールのコネクションを新しいコネクションに切り替える例です。

```
[DAS]domain1.adminServer>ctrlcp -server server1 -id ds1 -refresh
Servers that successfully refreshed a connection pool : server1
Servers that failed to refresh a connection pool : none.
```

6.9.5. コネクション・プールのコネクション数の最小化

本節では、コンソール・ツールとWebAdminを使ってコネクション・プールのコネクション数を最小化する方法について説明します。

WebAdminの使用

以下では、WebAdminを使ってコネクション・プールのコネクション数を最小化する方法について説明します。

1. WebAdminの左のメニューで**[Monitoring] > [Connection Pools]**を選択して表示される画面で任意のサーバーを選択すると、そのサーバーで有効なコネクション・プールのコネクション数を最小化することができます。本例では、ds1コネクション・プールのコネクション数を最小化するので、ds1をクリックします。

[図 6.100] コネクション・プールのコネクション数の最小化(1)

jeus_domain

Domain
Session
Clusters
Servers
Applications
Security
Resources
Monitoring

Thread
Transaction
MBean
JNDI
Web
Servers
JMS
Connection Pool
EJB Timer

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer
server1

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2
server3

2. ds1コネクション・プールのコネクション・リストが照会されます。現在コネクションの数は7個ですが、最小値の2に調整します。[shrink]ボタンをクリックします。

[図 6.101] コネクション・プールのコネクション数の最小化(2)

jeus_domain

Domain
Session
Clusters
Servers
Applications
Security
Resources
Monitoring

Thread
Transaction
MBean
JNDI
Web
Servers
JMS
Connection Pool
EJB Timer
System Info
Server Log
Statistic
Patch Info

Console
システム状態

0 Failed
0 Standby
4 Running
0 Shutdown
0 Suspended
0 Other

Connection Pool

HISTORY

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer
server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-14 | idle | 4.572 | 1 | pooled |
| ds1-17 | idle | 4.572 | 1 | pooled |
| ds1-15 | idle | 4.572 | 1 | pooled |
| ds1-11 | idle | 4.572 | 1 | pooled |
| ds1-13 | idle | 4.572 | 1 | pooled |
| ds1-16 | idle | 4.572 | 1 | pooled |
| ds1-12 | idle | 4.572 | 1 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 7 | 0 | 7 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2
server3

3. ds1コネクション・プールのコネクション数が最小化されていることが確認できます。

[図 6.102] コネクション・プールのコネクション数の最小化(3)

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Thread

Transaction

MBean

JNDI

Web

Servers

JMS

Connection Pool

EJB Timer

System Info

Server Log

Statistic

Patch Info

Console

Connection Pool

サーバに作成されたJDBC/JCAコネクションプールを制御し、ランタイム状態を照会します。

adminServer

server1

Information about connections in the server [server1]'s connection pool [ds1].

| Connection ID | State | State Time(sec) | Use Count | Type |
|---------------|-------|-----------------|-----------|--------|
| ds1-17 | idle | 33.138 | 1 | pooled |
| ds1-16 | idle | 33.138 | 1 | pooled |

Enable Shrink Disable Refresh Create

The connection pool information on the server [server1].

| Connection Pool ID | JNDI Export Name | Min | Max | Active | Idle | Disposable | Total | Wait | Enabled |
|--------------------|------------------|-----|-----|--------|------|------------|-------|-------|---------|
| ds1 | ds1 | 2 | 30 | 0 | 2 | 0 | 2 | false | true |
| ds2 * | ds2 | 2 | 30 | 0 | 0 | 0 | 0 | false | false |

* : has not been created, total = active + idle + disposable

server2

server3

コンソール・ツールの使用

コンソール・ツールで**shrink-connection-pool**を実行すると、サーバーに存在するコネクション・プールのコネクション数を設定したコネクションの最小値に調整できます。shrink-connection-poolのより詳しい使用方は、『JEUS リファレンスガイド』の「4.2.12.5. shrink-connection-pool」を参照してください。

以下は、shrink-connection-poolを実行して、server1に生成されたds1コネクション・プールのコネクション数を設定したコネクションの最小値に調整する例です。

```
[DAS]domain1.adminServer>shrinkcp -server server1 -id ds1
Servers that successfully shrank a connection pool : server1
Servers that failed to shrink a connection pool : none.
```

serverオプション値を指定していない場合、データソースが登録されたすべてのサーバーを対象にコネクション・プールのコネクション数の調整が試みられます。

以下は、shrink-connection-poolを実行して、ds2が登録されたserver1、server2に対してそれぞれds2コネクション・プールのコネクション数を設定したコネクションの最小値に調整する例です。


```
[DAS]domain1.adminServer>shrinkcp -id ds2
Servers that successfully shrank a connection pool : server1, server2
Servers that failed to shrink a connection pool : none.
```

control-connection-poolを実行する方法でもJDBCコネクション・プールのコネクション数を設定したコネクションの最小値に調整できます。control-connection-poolのより詳しい使用方法是、『*JEUS リファレンスガイド*』の「4.2.12.6. control-connection-pool」を参照してください。

以下は、control-connection-poolを実行して、server1に生成されたds1コネクション・プールのコネクション数を設定したコネクションの最小値に調整する例です。

```
[DAS]domain1.adminServer>ctrlcp -server server1 -id ds1 -shrink
Servers that successfully shrank a connection pool : server1
Servers that failed to shrink a connection pool : none.
```

6.10. JEUS JDBCプログラミング

本節では、JDBCアプリケーションのプログラミング・ルールについて簡単に説明します。

6.10.1. データソースからのコネクションの取得

以下のコードは、データソースを使ってコネクションを取得する方法の例です。

```
Context ctx = new InitialContext();
DataSource ds = (DataSource) ctx.lookup("ds1");
Connection con = ds.getConnection();
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from test");
...
con.close();
```

注意

コネクションを使用した後は必ずコネクションを閉じてください。

6.10.2. トランザクション・プログラミング・ルール

JEUSはトランザクション・プログラミングのための標準手順を定義します。UserTransactionを使用するすべてのアプリケーションは標準手順に従う必要があります。

1. トランザクションを開始します。
2. DBコネクションを取得します。

3. トランザクションの残りをコーディングします。

注意

他のトランザクション処理を行いたい場合は、新しいコネクションを取得する必要があります。

6.10.3. JDBCドライバーのコネクション実装インスタンスの取得

JEUSではアプリケーションにコネクションを渡すときに、コネクションの状態を管理するためにラッパーを使用します。しかし、OracleのCLOB、XMLTypeなどは、JDBCドライバーのコネクションの実装インスタンスを直接必要とするため、ドライバー内部でクラス・キャストを行い、ClassCastExceptionが発生します。これは、JEUSだけでなく、ほとんどのWAS製品で発生し得る問題であり、アプリケーションの作成者はWASから渡されるコネクションがドライバーのコネクション実装インスタンスであると仮定してアプリケーション・コードを作成してはなりません。しかし、これを迂回する方法があります。

アプリケーションでは以下のような方法でドライバーのコネクション実装インスタンスを取得できます。

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
...
Connection conn = datasource.getConnection();
DatabaseMetaData metaData = conn.getMetaData();
OracleConnection oraConn = (OracleConnection)metaData.getConnection();
```

6.10.4. スタンドアロン・クライアントでのコネクション・プール

スタンドアロン・クライアントでJEUSに登録されているデータソースをルックアップすると、そのデータソースの情報をJEUSから取得して自身のJVMにコネクション・プールを構成することができます。このとき、スタンドアロン・クライアントはclientcontainer.jarまたはjclient.jarをクラス・パスに追加する必要があります。

このような方式は、スタンドアロン・クライアントでJNDIルックアップ方式により便利にデータベースにアクセスできるというメリットはありますが、データベース側からみれば、WAS以外にJDBCドライバーにアクセスするクライアントが増えることになるので、クライアントが特にDBコネクションを効率的に管理する必要がなければ、クライアントにコネクション・プールを構成することはリソースの無駄になり得ます。したがって、実際にサービスを実装するときは、この方式の使用を推奨しません。

注

JEUSサーバーに構成されたコネクション・プールからコネクションを取得して使用するものではありません。そうしたい場合は、コネクションを使用するロジックをEJBで実装してJEUSにデプロイし、EJBを検索して(Lookup)使用する必要があります。

6.11. グローバル・トランザクション(XA)とコネクションの共有

コネクションの共有(Connection Sharing)は、同じトランザクション内では1つのリソースに対して常に1つのコネクションだけ使用することを保証するという概念で、JCA標準に記述されています。JDBC観点でみるリソースは、ほとんどデータベース(あるいは、データソース)です。JEUSは特に設定がなくても、基本的にXAデータソースおよびXAエミュレーション機能を使用するコネクション・プール・データソースに対してコネクションの共有を提供します。

アプリケーション・フレームワークでは、同じトランザクション内で1つのデータソースに対してコネクションを複数回要求するケースが多いので、コネクションの共有を使用することが望ましいです。そうしない場合、1つのデータソースに対して複数の物理的なコネクションがトランザクションに参加するようになります。これは、データベースに不要に多いトランザクション・ロックを管理しなければならない負担を与え、トランザクションの性能が全体的に低下する恐れがあります。

コネクションの共有を使用したくない場合は、XAデータソースを使用するJava EEコンポーネント設定ファイル(web.xml、ejb-jar.xmlなど)を以下のように作成します。<res-ref-name>にはデータソースのJNDI名を入力します。

```
<resource-ref>
<res-ref-name>jdbc/xads</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-sharing-scope>Unshareable</res-sharing-scope>
</resource-ref>
```

何も設定していない場合、基本的に<res-sharing-scope>がShareableであるため、サーブレットやEJBで常にコネクションの共有を使用します。ただし、XAエミュレーション機能を使用するコネクション・プール・データソースは常にコネクションを共有しなければならないため、Unshareableに設定するとエラー(java.sql.SQLException)が発生します。

6.12. 複数ユーザーに対するコネクション・プーリング・サービス

JEUSは、データソースの設定上、指定された基本ユーザーに対してのみコネクション・プーリング・サービスをサポートしてきました。そして、基本ユーザーではないユーザー情報でコネクションを要求される場合は、無条件に使い捨て(disposable)コネクションで対応しています。使い捨てコネクションは、その名前どおり、コネクションの要求があるたびに生成されて、使用後は直ちにコネクション・プールから削除されるため、そのような要求が多い場合には、コネクション・プーリング・サービスにかなりの負担になり得ます。

そのため、複数のユーザーに対してコネクション・プーリング・サービスをサポートしています。これは、1つのコネクション・プールを外部に対しては透過的にユーザー別に分けて管理する方式であり、クライアントは従来の方式と同様にコネクション・プーリング・サービスを利用することができます。つまり、コネクション・プールに存在するそれぞれのコネクションは、ユーザー情報は異なっても、コネクション・プールの全体的な設定と運用ポリシーは共通して従います。そのため、多数のユーザー別コネクションを頻繁に使用する場合も、使い捨てコネクションを作成するという負担がなく、より効率的にコネクション・プーリング・サービスを利用できます。

第7章 トランザクション・マネージャー

本章では、JEUSのトランザクション・マネージャーと関連要素について説明します。

7.1. 概要

JEUSTランザクション・マネージャーは、Java EE環境のアプリケーションに多様な形式のトランザクション・サービスを提供します。JEUSTランザクション・マネージャーは、アプリケーションの動作に必要なサービスを提供し、多様なリソース・マネージャーとの接続をサポートします。アプリケーションは必要に応じてトランザクションの制御権限をマネージャーに一任することもでき、直接持つこともできます。トランザクション・マネージャーはリソース・マネージャーやアプリケーションと共に、トランザクション作業を行い、その作業で中心的な役割を果たします。

トランザクション・サービスは、エンタープライズ・システムにおいてリソースに対する大容量のクライアント要求を安全かつ効率よく処理するための基本的で重要なサービスです。近年はリソースの種類が多様化しており、リソースに対する要求規模が増加するに伴い、トランザクション・マネージャーは統一されたインターフェースを提供しつつより安定的な動作を保証する必要性が出てきました。

JEUSのトランザクション・マネージャーは、クライアントに統一されたトランザクション・インターフェースと機能を提供するために、JTS(Java Transaction Service)との互換性を提供し、JTA(Java Transaction API)をサポートします。詳しくは、関連API文書や仕様書を参照してください。また多様なアプリケーションをサポートするために、アプリケーションが動作する様々な環境(クライアント／サーバ・アプリケーション)でのトランザクション・サービスを提供します。

トランザクション・マネージャーは安定的なサービスを提供するために、障害が発生してもトランザクションの整合性を保証する必要があります。JEUSTランザクション・マネージャーはトランザクション回復(Transaction Recovery)機能を提供し、様々な障害の後にトランザクションが回復できるようにします。

JEUSでのトランザクションには、大きく次の3つの要素が参加します。

- サービスを受ける側のアプリケーション

アプリケーションは、Webコンテナ、EJBコンテナ、クライアント・コンテナを介して様々なリソース・マネージャーに接続できます。アプリケーションが自らトランザクションの作業と終了を指定してトランザクションを調整することもでき、あるいはコンテナのトランザクション・マネージャーがそのような作業を代行することもできます。

- アプリケーションにトランザクション管理機能とそれに必要なインターフェースを提供するトランザクション・マネージャー

トランザクション・マネージャーは、アプリケーションが動作する場所によって、サーバ・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られ、リソース・マネージャーにトランザクションの準備、反映要求を転送します。

- トランザクションが実際に影響を与えるリソース・マネージャー(例: DB)

リソース・マネージャーはこのような要求を受けて作業を行います。

各構成要素はトランザクション処理のために、標準APIのJTAを利用して相互通信します。

本節では、トランザクションの各要素について説明し、それぞれの動作方式を記述します。トランザクションに関する基本的な内容は関連書籍や資料などを参照してください。

7.1.1. アプリケーション

アプリケーションはJEUSTランザクション・マネージャーを利用してリソース・マネージャーに対しトランザクション作業を行います。

- ローカル・トランザクション

1つのリソース・マネージャーを使用するトランザクションです。アプリケーションはリソース・マネージャーのドライバを使用してローカル・トランザクションを開始、終了します。

JEUSTランザクション・マネージャーは基本的にローカル・トランザクションの処理には関わりません。1つのリソース・マネージャーを利用する単純なトランザクション作業を行う場合は、ローカル・トランザクションを使用する方が効率的です。

- グローバル・トランザクション

アプリケーションが2つ以上のリソース・マネージャーを利用して一連のトランザクション作業を行う場合、トランザクション・マネージャーはこれらのトランザクションを1つのグローバル・トランザクションとして処理することができます。グローバル・トランザクションは通常、2相コミット・プロトコルを使って使用中のリソースに対する整合性を保証します。

トランザクション・マネージャーはトランザクションのコミット時に、アプリケーションの実行フローを追跡して1相コミットまたは2相コミットのいずれかを決定します。この決定はトランザクション内で使われたリソース・マネージャーの個数とタイプによって行われます。JEUSTランザクションは重複したトランザクションをサポートせず、2つのトランザクションを混合して実行することは許容しません。

- ユーザー管理(User Managed)トランザクション

アプリケーションは`javax.transaction.UserTransaction`オブジェクトを使ってトランザクションの開始と終了時期を指定することができます。トランザクションのコミットまたはロールバックはアプリケーションによって決定されます。しかし、コミットしたにもかかわらず、リソース・マネージャーがトランザクションを処理できない場合は、JEUSTランザクション・マネージャーがロールバックさせることができます。

EJBでは、このユーザー管理によるトランザクションをBean Managed Transaction(BMT)といいます。

- **コンテナ管理(Container Managed)トランザクション**

コンテナ管理によるトランザクションは、コンテナがトランザクションの開始と終了時期を指定する方式です。

トランザクションのコミットとロールバックの決定はコンテナによって行われます。EJBでのみ使用され、Beanのメソッド別にトランザクション属性(NotSupported、Required、Supports、RequiresNew、Mandatory、Never)を指定して使用します。

アプリケーションは利用するリソース・マネージャーの数によって、**ローカル・トランザクション**と**グローバル・トランザクション**の2つを使用できます。また、トランザクションの決定権限を持つ主体によって設定およびプログラミング方式を分けられます。

7.1.2. JEUSTランザクション・マネージャー

JEUSはサーバー・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・マネージャーの2つのトランザクション・マネージャーを提供します。

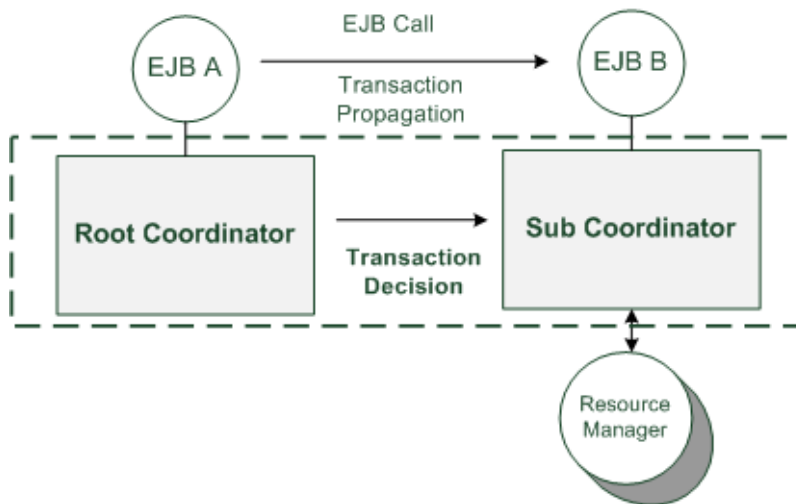
サーバー・トランザクション・マネージャーはグローバル・トランザクションのすべての管理機能を提供します。一方、クライアント・トランザクション・マネージャーはサーバー・トランザクションの代行(proxy)役割のみを行います。

サーバー・トランザクション・マネージャー(Server Transaction Manager)

サーバー・トランザクション・マネージャーはJava Transaction APIをすべて実装し、サーバー・アプリケーションがトランザクションを使用する時に動作します。

グローバル・トランザクションが開始されると、サーバー・トランザクション・マネージャーはグローバル・トランザクションに関わるすべての情報を収集し、トランザクションのコーディネーターとして作業を行います。状況によって**ルート・コーディネーター**と**サブ・コーディネーター**のいずれかの役割を果たします。

【図 7.1】 ルート・コーディネーターとサブ・コーディネーターの関係



- ルート・コーディネーター

トランザクションを開始するトランザクション・マネージャーが担います。開始されたトランザクションが伝播されるマネージャーがサブ・コーディネーターになります。

- サブ・コーディネーター

ルート・コーディネーターから決定を受けて該当する作業を実行します。すなわち、ルート・コーディネーターは能動的にトランザクションを管理し、サブ・コーディネーターはルート・コーディネーターの決定に従って受動的に動作します。

クライアント・トランザクション・マネージャー(Client Transaction Manager)

クライアント・トランザクション・マネージャーはサーバー・トランザクション・マネージャーの代行者(proxy)として動作します。

グローバル・トランザクションの信号を受信したサーバー・トランザクション・マネージャーがルート・コーディネーターの役割を果たし、当該トランザクションに関する情報をすべて収集して処理します。コミット時に、クライアント・トランザクション・マネージャーはコミット信号をルート・コーディネーターに転送し、コミットの結果を受信します。

7.1.3. リソース・マネージャー

アプリケーションは多様なリソース・マネージャーを使用することができます。リソース・マネージャーとの接続はコネクション・マネージャーが管理します。

JEUSでは、JDBCコネクション・マネージャー、JMSコネクション・マネージャー、WebTコネクション・マネージャー、コネクタ・マネージャーの4タイプのコネクション・マネージャーを提供します。コネクション・マネー

ジャーは、グローバル・トランザクションのためにトランザクションに関する情報をトランザクション・マネージャーに報告します。

- JDBCリソース・マネージャー

JDBC(Java Database Connectivity)は、Java言語でアプリケーションを開発時に使用するデータベースとアプリケーション間の接続を定義した仕様です。アプリケーションはJDBCリソース・マネージャーをネーミング・サーバーからルックアップして使用します。詳細については、「[第4章 JNDIネーミング・サーバー](#)」と「[第6章 DBコネクション・プールとJDBC](#)」を参照してください。

- JMSリソース・マネージャー

JMS(Java Message Service)は、キューとトピック、そしてそこに保存されたメッセージにアクセスするためのAPIを定義した仕様です。詳細については、『JEUS MQガイド』やJMS仕様を参照してください。

- Tmax(WebT)リソース・マネージャー

TmaxはTmaxSoftが開発したTPモニターです。JEUSでは、Tmaxのトランザクション・マネージャーとの透過的な双方向サービスのためにWebTというブリッジを提供します。WebTは双方向メソッド呼び出しをサポートするため、Tmaxで一般的なEJBクライアントを動作させたり、JEUSのEJB Beanを同じ方法で呼び出したりすることができます。

- JCAリソース・マネージャー

CA(Java Connector Architecture)はJava EE仕様の1つであり、Java EE互換プラットフォームと各種EIS(Enterprise Information System)との統合メカニズムを提供します。XAトランザクションをサポートするコネクタを使用して、アプリケーションは多様なトランザクションを1つのグローバル・トランザクションとして扱うことができます。詳細については、『JEUS JCAガイド』を参照してください。

7.2. サーバー・トランザクション・マネージャーの設定

本節では、JEUSトランザクション・マネージャーの設定方法について説明します。

JEUSトランザクション・マネージャーの設定は、JEUSトランザクション・マネージャーの性能や動作に大きな影響を与えます。したがって、設定内容を熟知した上で適切な設定を行う必要があります。

トランザクション・マネージャーの設定は、ほとんどWebAdminやコンソール・ツールを利用することができ、利用できない部分はシステム・プロパティで設定できます。

WebAdminの使用

以下では、WebAdminを使って設定を変更する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト照会画面が表示されます。
2. 動的に設定を変更するために**[LOCK & EDIT]**ボタンをクリックします。WebAdminの設定変更モードについての詳しい説明は、『JEUS WebAdminガイド』を参照してください。
3. サーバー・リストでサーバーを選択すると、サーバー設定画面に移動します。サーバー設定画面で**[Basic]** > **[Tm Config]**メニューを選択すると、**TM Config**画面に移動します。
4. トランザクションの設定を変更して**[確認]**ボタンをクリックします。設定情報は左のメニューの下部にある**[Activate Changes]**ボタンをクリックしてサーバーに反映します。

7.2.1. ワーカー・スレッド・プールの設定

JEUSTランザクション・マネージャーでは、他のランザクション・マネージャーとの通信をサポートするために複数のワーカー・スレッドを使用します。ランザクション・マネージャーは基本的にサーバーのシステム・スレッド・プールを使用します。ランザクションが重要かつ迅速な処理を求めるサーバーではスレッド・プールを設定できます。

● 共用スレッド・プール

ランザクションが多くはないけれど、一定のレベルで発生しており、他のサービスより迅速な処理が求められる場合は、サーバー全体サービスが共用するシステム・スレッド・プールを設定します。

共用スレッド・プールを設定する場合、ランザクション・マネージャーのみ使用できるスレッド数を「**Reserved Thread Num**」項目を利用して別途割り当てることができます。その際は、他のサービスも処理に問題がないように、全体スレッド・プールのサイズを考慮してランザクション・マネージャー用のスレッドを割り当てるようにします。

注

他のサービスでもそのサービスのためのスレッドを割り当てることができますが、その総計が全体スレッド・プールのサイズを越えてはいけません。「**Reserved Thread Num**」を動的に変更する場合、(n → m) nまたはmが0なら、動的に反映されず、ペンディング処理されるので再起動する必要があります。

● 専用スレッド・プール

トランザクションがとて多く、負荷の変動が激しい場合は、システム・スレッドを使用せずに、トランザクション・マネージャーだけが使用する専用スレッドを作成して使用します。スレッド・プールを構成する設定はシステム・スレッド・プールの設定と同じであるので、「[2.3.3. スレッド・プールの設定](#)」を参照してください。

上述したように、システム・スレッド・プールを使用することも、専用スレッド・プールを使用することもできますが、サーバーの運用中はプールの種類を変更できません。つまり、変更した後、サーバーを再起動しなければ、プールの種類を反映できません。しかし、プールの設定は、専用スレッド・プールのキュー・サイズ以外は、すべてサーバーを再起動しなくても動的変更が可能です。

WebAdminの使用

以下では、WebAdminを使ってスレッド・プールを変更する手順について説明します。Tm Config画面の下部でスレッド・プールの設定を変更できます。スレッド・プールの設定を行うには、「Pooling」項目にチェックします。

● 共用スレッド・プールの設定

共用スレッド・プールを設定する場合は、「Shared」項目を選択し、「Reserved Thread Num」を設定した後[確認]ボタンをクリックします。

[図 7.2] WebAdminによる共用スレッド・プールの設定

The screenshot shows the 'Pooling' configuration interface in WebAdmin. At the top, the 'Pooling' checkbox is checked. Below it, there are two tabs: 'Shared' (selected) and 'Dedicated'. Under the 'Shared' tab, the 'Reserved Thread Num' is set to 10. Under the 'Dedicated' tab, there are input fields for 'Min', 'Max', 'Keep Alive Time' (with a unit of 'ms'), and 'Queue Size'. Below these tabs is a section titled 'Stuck Thread Handling' which contains 'Max Stuck Thread Time' (with a unit of 'ms'), 'Action On Stuck Thread' (a dropdown menu), and 'Stuck Thread Check Period' (with a unit of 'ms').

● 専用スレッド・プールの設定

専用スレッド・プールを設定する場合は、「Dedicated」項目を選択し、必要な項目を設定した後[確認]ボタンをクリックします。

[図 7.3] WebAdminによる専用スレッド・プールの設定

Pooling ☒

☐ Shared

Reserved Thread Num

☒ Dedicated

Min

Max

Keep Alive Time ms

Queue Size

▼ Stuck Thread Handling

Max Stuck Thread Time ms

Action On Stuck Thread

Stuck Thread Check Period ms

コンソール・ツールの使用

以下では、コンソール・ツールを使ってスレッド・プールを変更する方法について説明します。

● 共用スレッド・プールの設定

システム・スレッド・プールで一部スレッドをトランザクション・マネージャー用に別途割り当てるには、**modify-system-thread-pool**コマンドを使用します。

```
[DAS]domain1.adminServer>modify-system-thread-pool server1 -service transaction
-reservednum 10
Successfully performed the MODIFY operation for the transaction thread pool of the
server (server1).
Check the results using "show-system-thread-pool server1 -service transaction or
modify-system-thread-pool server1 -service transaction"
```

● 専用スレッド・プールの設定

トランザクション・マネージャー専用スレッドを使用するには、**modify-service-thread-pool**コマンドを使用します。

システム・スレッド・プールを使用していて専用スレッド・プールを使用するように設定する場合は、サーバーを再起動して適用する必要がある、専用スレッド・プールを使用しつつ属性のみ変更する場合は、再起動せずに反映されます。

```
[DAS]domain1.adminServer>modify-service-thread-pool server1 -service transaction
-min 10 -max 20
Successfully performed the MODIFY operation for The transaction thread pool of the
server (server1), but all changes were non-dynamic. They will be applied after
restarting.
Check the results using "show-service-thread-pool server1 -service transaction or
modify-service-thread-pool server1 -service transaction"
```

参考

modify-system-thread-poolとmodify-service-thread-poolコマンドについての詳細は、『JEUS リファレンスガイド』の「4.2.5.2. modify-service-thread-pool」を参照してください。

7.2.2. タイムアウトの設定

JEUSトランザクション・マネージャーは例外を処理するために、多様なタイムアウト・メカニズムを使用します。タイムアウトの値を調整することで、アプリケーション・システムに最も適合するようにトランザクション・マネージャーをチューニングすることができます。タイムアウト設定はすべてサーバーの再起動により反映されます。したがって、サーバーを起動する前に設定するか、またはサービス中に変更した場合はサーバーを再起動する必要があります。

WebAdminの使用

以下では、WebAdminを使ってタイムアウトを変更する方法について説明します。Tm Config画面の上部で様々なタイムアウト設定を変更できます。

[図 7.4] WebAdminによるタイムアウトの変更

動的設定 * 必須項目

確認再設定削除

| | | |
|--------------------|-------------------------------------|---|
| Active Timeout | <input type="text" value="600000"/> | [デフォルト: 600000] グローバルトランザクションが開始されたら、設定した時間内にコミットが実行される必要があります。そうでない場合は、トランザクションマネージャーがロールバックさせます。 |
| Automatic Recovery | <input type="checkbox"/> | [デフォルト: false] クラスタリング環境において、現在のトランザクションマネージャーに障害が生じた場合、別の場所で自動的にIndoubtトランザクションを復旧する機能を使用するか否かを設定します。この機能が正常に動作するためには、クラスタリングが設定された上で、Failed TMのログディレクトリを別の場所でアクセスできるように設定する必要があります。 |

詳細設定

すべてを開く

| | | |
|--------------------|---|----|
| Prepare Timeout | <input type="text" value="120000"/> | ms |
| Prepared Timeout | <input type="text" value="60000"/> | ms |
| Commit Timeout | <input type="text" value="240000"/> | ms |
| Recovery Timeout | <input type="text" value="120000"/> | ms |
| Incomplete Timeout | <input type="text" value="86400000"/> | ms |
| Tx Log Dir | <input type="text" value="\${SERVER_HOME}/.workspace/tmlog"/> | |

以下は、設定項目についての説明です。

- 基本情報

| 項目 | 説明 |
|--------------------|---|
| Active Timeout | グローバル・トランザクションが開始されたら、設定した時間内にコミットが実行される必要があります。そうでない場合は、トランザクション・マネージャーがロールバックさせます (デフォルト値: 600000(10分)、単位: ms) |
| Automatic Recovery | クラスタリング環境において、現在のトランザクション・マネージャーに障害が発生した場合、他の場所で自動的にインダウト・トランザクションを復旧する機能を使用するか否かを設定します。この機能が正常に動作するためには、クラスタリングが設定された上で、他の場所から失敗したトランザクション・マネージャーのログ・ディレクトリーにアクセスできるように設定する必要があります |

- 詳細設定

| 項目 | 説明 |
|------------------|--|
| Prepare Timeout | ルート・コーディネーターは設定時間内にスタブ・コーディネーターとリソース・マネージャーから「prepare」信号を受信する必要があります。そうでない場合、ルート・コーディネーターはグローバル・トランザクションをロールバックさせます (デフォルト値: 120000(2分)、単位: ms) |
| Prepared Timeout | サブ・コーディネーターは、自身のルート・コーディネーターから、設定した時間内にコミットするかロールバックするかを示すグローバル・デシジョンを受ける必要があります。そうでない場合、ルート・コーディネーターに再び「prepare」に対する応答メッセージを返します。それでも設定した時間内にグローバル・デシジョンが渡されない場合は、グローバル・トランザクションをロールバックさせます(デフォルト値: 60000(1分)、単位: ms) |
| Commit Timeout | ルート・コーディネーターは、設定した時間内にサブ・コーディネーターとリソース・マネージャーからコミットまたはロールバックの結果を受ける必要があります。結果を受けられなかった場合、ルート・コーディネーターはグローバル・トランザクションを「Incomplete List」に記録し、トランザクションが完全に終了していないことを記録しておきます(デフォルト値: 240000(4分)、単位: ms) |
| Recovery Timeout | トランザクションを回復するときに使用される値です。 トランザクション・マネージャーはトランザクションを回復するために、回復されるトランザクション情報を取得しようとします。設定時間内に他のトランザクション・マネージャーが回復情報を提供しない場合、そのトランザクションをロールバックさせます(デフォルト値: 120000(2分)、単位: ms) |

| 項目 | 説明 |
|--------------------|---|
| Incomplete Timeout | <p>トランザクション・マネージャーは、トランザクション全体の処理を完了するために、失敗したグローバル・トランザクションのリストを保持します。完了できなかったグローバル・トランザクションの情報は回復を処理するとき使用されるため、設定した時間まで保持されます。</p> <p>したがって、設定時間が短すぎると回復情報が早く削除され、トランザクション・マネージャーが該当するグローバル・トランザクションの整合性を保証できなくなります。その結果、グローバル・トランザクションを回復するために、システム管理者が多くの作業を直接処理しなければなりません(デフォルト値: 86400000(12日)、単位: ms)</p> |

コンソール・ツールの使用

以下では、コンソール・ツールを使ってタイムアウトを変更する方法について説明します。

modify-transaction-manager コマンドを使ってトランザクション・マネージャーのタイムアウトを変更できます。このコマンドの詳細については、『*JEUS リファレンスガイド*』の「4.2.13.1. modify-transaction-manager」を参照してください。

```
[DAS]domain1.adminServer>modify-transaction-manager server1 -activeTimeout 20000
Successfully performed the MODIFY operation for transaction of server (server1), but all
changes were non-dynamic. They will be applied after restarting.
Check the results using "show-transaction-manager server1 or modify-transaction-manager
server1"
```

7.2.3. ルート・コーディネーターとサブ・コーディネーターに関する設定

トランザクション・マネージャーはその役割によって、ルート・コーディネーターとサブ・コーディネーターに分けられます。サブ・コーディネーターはルート・コーディネーターから伝播されるトランザクション関連情報を受信するために、ルート・コーディネーターに自身を登録する必要があります。

以下の2つのシステム・プロパティを使って設定できます。性能や安定性を考慮して適切にオプションを使用してください。

- **-Djeus.tm.forcedReg=<true or false>**

この設定はルート・コーディネーターにサブ・コーディネーター自身を登録するタイミングに関する設定です。

| 設定値 | 説明 |
|------|---|
| true | トランザクションの伝播状況でサブ・コーディネーターは直ちに自身を登録します(デフォルト値) |

| 設定値 | 説明 |
|-------|---|
| false | サブ・コーディネーターに接続されたリソース・マネージャーを使用する時にルート・コーディネーターに登録します。これにより、サブ・コーディネーターに登録されたリソース・マネージャーを使用しないEJBコールが多い場合に、トランザクション・マネージャー間の通信回数を減らすことができます |

- `-Djeus.tm.checkReg=<true or false>`

| 設定値 | 説明 |
|-------|---|
| true | サブ・コーディネーターが自身を登録するときに、ルート・コーディネーターからのACKを待ちます。一定時間内にACKが到着しないと登録に失敗したと判断し、待機中のトランザクションをロールバックします(デフォルト値) |
| false | サブ・コーディネーターが自身を登録するときに、ルート・コーディネーターからのACKを待たないため、ネットワークなどの問題で登録されなかった場合も事前に知ることができず、実際にコミットするときに問題になります |

7.2.4. トランザクション結合の設定

JEUSTランザクション・マネージャーは同じリソース・マネージャーのトランザクション・リソースを相互に結合します。これにより、トランザクション・ブランチを追加作成しなくても作業が可能になり、オーバーヘッドを減らすことができます。しかし、場合によっては、(例えば、Oracle OCIをRACのように使用する場合)このような方式が問題になることもあります。

そのような場合は、次のオプションをtrueに指定して問題の発生を防ぎます。

```
-Djeus.tm.disableJoin=true
```

7.3. クライアント・トランザクション・マネージャーの設定

クライアント・トランザクション・マネージャーは、クライアント・アプリケーションがグローバル・トランザクションを開始または終了するために生成されました。本節では、クライアント・トランザクション・マネージャーの設定について説明します。

クライアント・アプリケーションはdomain.xmlの設定などを使わないので、アプリケーションの実行スクリプトにシステム・プロパティ形式で設定します。

7.3.1. トランザクション・マネージャーの使用可否の設定

クライアント・トランザクション・マネージャーは、クライアント・アプリケーションがJNDIルックアップを使用時に初期化されます。しかし、クライアント・アプリケーションによっては、トランザクション・マネージャーを使用し

ない場合があります。このとき、追加的なオーバーヘッドの発生を防ぐために、次の内容を実行スクリプトに保存します。

```
-Djeus.tm.not_use=true
```

7.3.2. トランザクション・マネージャーのタイプの設定

クライアント・トランザクション・マネージャーはクライアント・コンテナでのみ使用されます。基本的にクライアントではクライアント・トランザクション・マネージャーを使用します。特にサーバー・トランザクション・マネージャーをクライアントで使用するには、次の内容をクライアント・アプリケーションのスクリプトに追加します。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・マネージャーの違いについては、[「7.1.2. JEUSTランザクション・マネージャー」](#)を参照してください。

```
-Djeus.tm.version=server
```

7.3.3. トランザクション・マネージャーのTCP/IPポートの設定

クライアント・トランザクション・マネージャーは他のトランザクション・マネージャーとの通信のためにTCP/IPポートを使用します。

クライアント・トランザクション・マネージャーは適切なポートを自ら探して使いますが、直接設定する場合は、クライアント・アプリケーションのスクリプトに次の内容を追加します。

```
-Djeus.tm.port=<port number>
```

7.3.4. ワーカー・スレッド・プールの設定

ワーカー・スレッド・プールの情報を設定します。

- Min

- プールに生成されるワーカー・スレッドの数です。クライアント・トランザクション・マネージャーでこの値を設定するには、以下の内容をクライアント実行スクリプトに含めます。

```
-Djeus.tm.tmMin=<number of threads>
```

- Max

- プールに生成されるスレッドの最大数です。クライアント・トランザクション・マネージャーでこの値を設定するには、以下の内容をクライアント実行スクリプトに含めます。

```
-Djeus.tm.tmMax=<number of threads>
```

7.3.5. タイムアウトの設定

以下では、クライアントでトランザクション・マネージャーを設定する方法について説明します。

● Active Timeout

- グローバル・トランザクションがこの時間内にコミットされない場合、トランザクション・マネージャーがトランザクションをロールバックします。(デフォルト値: 600000(10分)、単位: ms)
- クライアント・コンテナでこの値を設定するには、クライアントアプリケーションのスクリプトに以下を追加します。

```
-Djeus.tm.activeto=<time in milliseconds>
```

● Prepare Timeout

- ルート・コーディネーターがこの時間内にサブ・コーディネーターとリソース・マネージャーから「prepare」信号を受信する必要があります。(デフォルト値: 120000(2分)、単位: ms)
- 受信できない場合、ルート・コーディネーターはグローバル・トランザクションをロールバックします。
- クライアント・コンテナでこの値を設定するには、クライアントアプリケーションのスクリプトに以下を追加します。

```
-Djeus.tm.preparedto=<time in milliseconds>
```

● Prepared Timeout

- サブ・コーディネーターがこの時間内に自身のルート・コーディネーターからコミットかロールバックのグローバル決定を受ける必要があります。(デフォルト値: 60000(1分)、単位: ms)
- 設定時間内に受信できない場合は、ルート・コーディネーターに再び「prepare」に対する応答メッセージを送ります。それでも時間内にグローバル決定が送られない場合は、グローバル・トランザクションをロールバックします。
- クライアント・コンテナでこの値を設定するには、クライアントアプリケーションのスクリプトに以下を追加します。

```
-Djeus.tm.preparedto=<time in milliseconds>
```

● Commit Timeout

- ルート・コーディネーターがこの時間内にサブ・コーディネーターとリソース・マネージャーからコミットまたはロールバックの結果を受ける必要があります。(デフォルト値: 240000(4分)、単位: ms)

- 結果を受けられなかった場合、ルート・コーディネーターはグローバル・トランザクションを「Incomplete List」に記録し、トランザクションが完全に終了しなかったことをログに残します。
- クライアント・コンテナでこの値を設定するには、クライアント・アプリケーションのスクリプトに以下を追加します。

```
-Djeus.tm.committo=<time in milliseconds>
```

● Recovery Timeout

- トランザクションを回復するときに使用されます。トランザクション・マネージャーが回復するトランザクションの情報を取得するまで待機する時間です。(デフォルト値: 120000(2分)、単位: ms)
- 他のトランザクション・マネージャーがこの時間内に回復情報を通知しない場合は、当該トランザクションをロールバックします。
- クライアント・コンテナでこの値を設定するには、クライアント・アプリケーションのスクリプトに以下を追加します。

```
-Djeus.tm.recoveryto=<time in milliseconds>
```

● Incomplete Timeout

- トランザクション・マネージャーは全体トランザクション処理を完了するために、失敗したグローバル・トランザクションのリストを保管します。完了していないグローバル・トランザクションの情報は回復の際に使用されるため、このタイムアウト時間まで保管されます。したがって、この時間が短すぎると、回復情報が早く削除されてしまい、トランザクション・マネージャーがグローバル・トランザクションの整合性を回復できなくなります。その結果、グローバル・トランザクションの回復のためにシステム管理者が多くの処理を直接処理しなければなりません。(デフォルト値: 86400000(1日)、単位: ms)
- クライアント・コンテナでこの値を設定するには、クライアント・アプリケーションのスクリプトに以下を追加します。

```
-Djeus.tm.incomplete.to=<time in milliseconds>
```

7.4. トランザクション・アプリケーションの作成

本節では、JEUSTランザクション・プログラミングの例について説明します。。

トランザクション・プログラミングのパターンには次の4つがあります。

- ローカル・トランザクション(Local Transaction)

- クライアント管理トランザクション(Client Managed Transaction)
- Bean管理トランザクション(Bean Managed Transaction)
- コンテナ管理トランザクション(Container Managed Transaction)

ユーザーやアプリケーション・プログラマーは、アプリケーションの運用方式を明確にした後プログラミングする必要があります。そうすることで、予測された結果が得られ、問題が発生したときに簡単に解決することができます。

アプリケーションはトランザクション作業を1つのトランザクションとして管理することができます。作業が1つのリソース・マネージャーによって処理される場合は、ローカル・トランザクションを構成します。そうでない場合は、トランザクションを管理するためにグローバル・トランザクションを使用します。

7.4.1. ローカル・トランザクション(Local Transaction)

ローカル・トランザクションは1つのリソース・マネージャーのトランザクション作業を管理するのに効果的な方法です。軽くて速いので、ローカル・トランザクションを使用することをお勧めします。ローカル・トランザクションはJEUSTランザクション・マネージャーとは関係がなく、すべてのコンテナ・タイプで使用できます。

以下は、ローカル・トランザクションを使用する簡単な例です。Javaアプリケーションですが、一部のコードはサーブレットやEJBなどの他のJava EEプログラミングでも使用できます。

[例 7.1] <<Client.java>>

```
import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import java.util.*;
import javax.transaction.UserTransaction;
import java.sql.*;
import javax.sql.*;

public class Client {
    private static Connection con;

    private static Connection getConnection() throws
        NamingException, SQLException {
        // get a JDBC connection
    }

    private static String insertCustomer(String id, String name,
        String phone) throws NamingException, SQLException {
        // insert a product entity given by the arguments to DB
    }

    private static void deleteCustomer(String id) throws
        NamingException, SQLException {
```

```

        // delete a product entity given by the arguments from DB

    }

    public static void main(String[] args) {
        try {
            // get a JDBC connection
            con = getConnection();

            // set the autocommit attribute as false
            con.setAutoCommit(false);

            // insert customers
            for (int i=0 ; i<number/2 ; i++) {
                System.out.println("inserting customer id="+i+"c... from Client");
                customers[i] =
                    insertCustomer(i+"c", "Hong Kil Dong "+i, "000-123-1234-"+i);
            }
            System.out.println("completed inserting customers!!");
            con.commit();

            // delete customers
            for (int i=0 ; i<number/2 ; i++) {
                System.out.println(
                    "deleting customerid="+customers[i]+" ... from Client");
                deleteCustomer(customers[i]);
            }
            System.out.println("completed deleting customers!!");
            con.commit();

        } catch (NamingException ne) {
            System.out.println("Naming Exception caught : " + ne.getMessage());
        } catch (SQLException sqle) {
            System.out.println("SQL Exception caught : " + sqle.getMessage());
        } catch (Exception e) {
            try {
                con.rollback();
            } catch (Exception ee) {
                System.out.println("Transaction Rollback error : " + ee.getMessage());
            }
            System.out.println("Error caught : " + e.getMessage());
            e.printStackTrace();
        } finally {
            try {
                if (con!=null) con.close();
            } catch (SQLException se) {}
        }
    }
}

```

7.4.2. クライアント管理トランザクション(Client Managed Transaction)

クライアント・コンテナのアプリケーションでグローバル・トランザクションを使用できます。クライアントはユーザー・トランザクションを利用して直接トランザクションを管理し、実際に作業を実行するEJBを呼び出します。

次は、簡単な例です。

クライアント

グローバル・トランザクションを開始する前に「java:comp/UserTransaction」をルックアップして javax.transaction.UserTransaction のインスタンスを取得します。このインスタンスの begin() を呼び出してグローバル・トランザクションを開始した後、EJB Bean を数回呼び出します。EJB Bean のトランザクション作業をコミットするには、ユーザー・トランザクション・インスタンスの commit() メソッドを実行してトランザクションの終了を通知します。

コミットが正常に実行されると、メソッドは終了します。障害が発生してグローバル・トランザクションがコミットされなかった場合は、メソッドが例外を投げます。javax.transaction.RollbackException は、グローバル・トランザクションがロールバックされたことを JEUS トランザクション・マネージャーが保証する例外です。その他の例外は、予測できない例外が発生して JEUS トランザクション・マネージャーがグローバル・トランザクションをロールバックしようとすることを示します。ただし、グローバル・トランザクションに含まれたすべてのトランザクション作業が完全にロールバックされるわけではありません。この場合には、catch 文の中でもう一度ユーザー・トランザクションの rollback() メソッドを直接呼び出してグローバル・トランザクションをロールバックします。

参考

スタンドアロン・クライアントはコンポーネントの概念がないので、java:comp/UserTransaction をスタンドアロン・クライアントで他の名前でもルックアップできるように java:/UserTransaction をサポートします。実際にクライアントでは、java:comp/UserTransaction と java:/UserTransaction の両方をトランザクションのルックアップのために使用することができ、どれを使用しても構いません。

[例 7.2] <<Client.java>>

```
package umt;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.rmi.PortableRemoteObject;
import java.util.*;
import javax.transaction.UserTransaction;

public class Client {
    public static void main(String[] args) {
        UserTransaction tx = null;
        try {
            InitialContext initial = new InitialContext();
```

```

ProductManager productManager =
    (ProductManager)initial.lookup("productmanager");
System.out.println("");
System.out.println("< Testing ProductManager EJBBean " +
    "Using User Managed Transaction >>");
System.out.println("");
int number = 10;
String products[] = new String[number];
tx = (UserTransaction)initial.lookup("java:comp/UserTransaction");
tx.begin();
// insert products
for (int i=0 ; i<number/2 ; i++) {
    System.out.println("inserting product id="+i+"b...");
    // bean call
    products[i] = productManager.insertProduct(i+"b","ball pen", i*10);
}
for (int i=number/2 ; i<number ; i++) {
    System.out.println("inserting product id="+i+"f...");
    products[i] = productManager.insertProduct(i+"f", "fountain pen", (i-number/3)*50);
}
System.out.println("completed inserting products!!");
// delete products
for (int i=0 ; i<number ; i++) {
    System.out.println("deleting productid="+products[i]+" ...");
    productManager.deleteProduct(products[i]); // bean call
}
System.out.println("completed deleting products!!");
tx.commit();
} catch (javax.transaction.SystemException se) {
    System.out.println("Transaction System Error caught : " + se.getMessage());
} catch (javax.transaction.RollbackException re) {
    System.out.println("Transaction Rollback Errorcaught : " + re.getMessage());
} catch (Exception e) {
    try {
        tx.rollback();
    } catch (Exception ee) {
        System.out.println("Transaction Rollback error : " + ee.getMessage());
    }
    System.out.println("Error caught : " + e.getMessage());
    e.printStackTrace();
}
}
}
}

```

EJB

以下のEJB Beanはトランザクション処理を行うメソッドを有します。クライアントでグローバル・トランザクションを管理するには、EJBのトランザクション・タイプがCMT(コンテナ管理によるトランザクション)である必要が

あります。デフォルト値がCMTなので、特に指定する必要はありませんが、明示的に指定したい場合は、下記のようにTransactionManagerTypeの値を指定します。

[例 7.3] <<ProductManagerEJB.java>>

```
package umt;

import javax.ejb.*;
import javax.annotation.*;

@Stateless
@TransactionManagement(TransactionManagementType.CONTAINER)
public class ProductManagerEJB implements ProductManager {
    ....

    public String insertProduct(String id, String name, double price) {
        // insert a product entity given by the arguments to DB
    }

    public void deleteProduct(String id) {
        // delete a product entity indicated by the argument from
        // DB
    }
    .....
}
```

7.4.3. Bean管理トランザクション(Been Managed Transaction)

WebコンテナとEJBコンテナで、アプリケーションはJTAを使ってグローバル・トランザクションの境界を定義することができます。アプリケーションがグローバル・トランザクションを精密に調整する必要がある場合に有用です。実行手順に従ってクライアントの要求が処理できるので、アプリケーションは自身の判断どおりにコミットとロールバックを実行することができます。

以下は、EJBコンテナでアプリケーションがグローバル・トランザクションを実行する例です。

クライアント

Bean管理トランザクション(BMT)では、EJB Beanがグローバル・トランザクションを処理します。そのため、すべてのクライアントはトランザクション処理を行うメソッドだけを呼び出せばいいです。

[例 7.4] <<Client.java>>

```
package bmt;

import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import java.util.*;

public class Client {
```



```

public static void main(String[] args) {
    try {
        ProductManager productManager;
        .....
        // Getting a reference to an instance of
        // ProductManager EJB bean.
        .....
        productManager.transactionTest();
    } catch(Exception e) {
        e.printStackTrace();
    }
}
}

```

EJB

EJB Beanはグローバル・トランザクションを開始するために、`javax.transaction.UserTransaction`を使用します。`javax.ejb.EJBContext`(本例では、EJBがセッションBeanなので`javax.ejb.SessionContext`)を使ってユーザー・トランザクションのインスタンスを取得します。グローバル・トランザクションはメソッドの実行によって開始およびコミットされます。

EJBがBean管理トランザクションで動作するには、アノテーションによって`TransactionManagement`値を`TransactionManagementType.BEAN`に指定します。

[例 7.5] <<ProductManagerEJB.java>>

```

package bmt;

import javax.ejb.*;
import javax.naming.*;
import java.sql.*;
import java.util.*;
import javax.annotation.*;
import javax.transaction.UserTransaction;

@Stateless
@TransactionManagement(TransactionManagementType.BEAN)
public class ProductManagerEJB implements ProductManager {
    @Resource SessionContext ejbContext;

    public void transactionTest() {
        UserTransaction tx = null;
        try {
            int number = 20;
            String products[] = new String[number];
            tx = ejbContext.getUserTransaction();
            tx.begin();
            // insert products
            for (int i=0 ; i<number/2 ; i++) {
                System.out.println("inserting product id="+i+"b ...");
            }
        } catch (Exception e) {
            tx.rollback();
            e.printStackTrace();
        }
    }
}

```

```

        products[i] = insertProduct(i+"b", "ball pen", i*10);
    }
    for (int i=number/2 ; i<number ; i++) {
        System.out.println("inserting product id="+i+"f...");
        products[i] = insertProduct(i+"f", "fountain pen",
            (i-number/3)*50);
    }
    System.out.println("completed inserting products!!");
    // delete products
    for (int i=0 ; i<number ; i++) {
        System.out.println("deleting product id="+products[i]+" ...");
        deleteProduct(products[i]);
    }
    System.out.println("completed deleting products!!");
    tx.commit();
} catch (javax.transaction.SystemException se) {
    throw new EJBException(
        "Transaction System Error caught : " + se.getMessage());
} catch (javax.transaction.RollbackException re) {
    throw new EJBException(
        "Transaction Rollback Error caught : " + re.getMessage());
} catch (Exception e) {
    try {
        tx.rollback();
    } catch (Exception ee) {
        throw new EJBException("Transaction Rollback error : " + ee.getMessage());
    }
    throw new EJBException("Error caught : " + e.getMessage());
}
}

private String insertProduct(String id, String name, double price)
    throws NamingException, SQLException {
    // insert a product entity given by the arguments to DB
}

private void deleteProduct(String id) throws NamingException, SQLException {
    // delete a product entity indicated by the argument from
    // DB
}

// some EJB callback methods
}

```

7.4.4. コンテナ管理トランザクション(Container Managed Transaction)

Java EE仕様によると、アプリケーションはグローバル・トランザクションの境界(demarcation)の指定をコンテナに委任することができます。WebコンテナやEJBコンテナはメソッドと関連するグローバル・トランザクションを管理します。アプリケーションは設定ファイルにトランザクションの属性をメソッド別に設定できます。これがグローバル・トランザクションを処理する最も簡単な方法です。

以下は、EJBコンテナが管理するグローバル・トランザクションの例です。

クライアント

以下は、コンテナ管理トランザクションでサーバー側のEJBコンテナがEJBのグローバル・トランザクション処理を管理する例です。

[例 7.6] <<Client.java>>

```
package bmt;

import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import java.util.*;

public class Client {
    public static void main(String[] args) {
        try {
            ProductManager productManager;
            // getting a reference to an instance of
            // ProductManager EJB bean.
            productManager.transactionTest();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

EJB

コンテナ管理トランザクションのメリットは、EJB Beanの開発者がビジネス・ロジック・コードでトランザクション関連コードを作成する必要がなくなったことです。

EJB Beanのメソッドに適切なトランザクションの属性さえ指定すれば、グローバル・トランザクション関連作業はすべてEJBコンテナに委任されます。下記例のtransactionTest()は、トランザクション関連コードがありません。EJB Beanがコンテナ管理トランザクションで動作するようにEJBコンテナに通知すると、TransactionManagementアノテーションを省略するか、TransactionManagerType.CONTAINER値に指定します。

[例 7.7] <<ProductManagerEJB.java>>

```
package cmt;

import javax.ejb.*;
import javax.naming.*;
import java.sql.*;
import java.util.*;
import javax.annotation.*;
```

```

@Stateless
@TransactionManagement(TransactionManagementType.CONTAINER)
public class ProductManagerEJB implements ProductManager {
    public void transactionTest() {
        try {
            int number = 20;
            String products[] = new String[number];
            // insert products
            for (int i=0 ; i<number/2 ; i++) {
                System.out.println("inserting product id="+i+"b...");
                products[i] = insertProduct(i+"b", "ball pen", i*10);
            }
            for (int i=number/2 ; i<number ; i++) {
                System.out.println("inserting product id="+i+"f...");
                products[i] = insertProduct(i+"f", "fountain pen", (i-number/3)*50);
            }
            System.out.println("completed inserting products!!");
            // delete products
            for (int i=0 ; i<number ; i++) {
                System.out.println("deleting product id="+products[i]+" ...");
                deleteProduct(products[i]);
            }
            System.out.println("completed deleting products!!");
        } catch (Exception e) {
            throw new EJBException("Error caught : " + e.getMessage());
        }
    }

    private String insertProduct(String id, String name, double price)
        throws NamingException, SQLException {
        // insert a product entity given by the arguments to DB
    }

    private void deleteProduct(String id) throws NamingException, SQLException {
        // delete a product entity indicated by the argument from
        // DB
    }

    // some EJB callback methods
}

```

7.4.5. トランザクション・マネージャーの使用

トランザクション・マネージャーを直接使用する場合は、JNDIコンテキストで `java:appserver/TransactionManager` の名前でルックアップすると、トランザクション・マネージャーが取得できます。トランザクション・マネージャーはトランザクションを開始・終了し、トランザクション・オブジェクトの取得、XAリソース・オブジェクトの登録機能を提供します。詳しい説明は「Java Transaction API(JTA)」仕様を参照してください。

参考

JEUSでトランザクション・マネージャーまたはユーザー・トランザクションを取得する方法は、GlassfishまたはSunOneアプリケーション・サーバーと互換されます。したがって、Hibernateのようなサード・パーティー・ライブラリーを使用する時に、互換サーバーの設定をそのまま使用することができます。

7.5. トランザクションのリカバリー

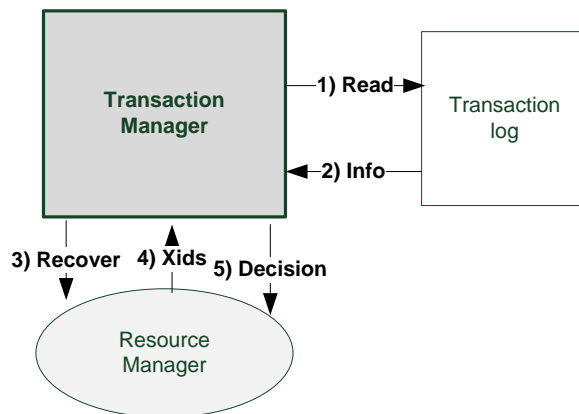
本節では、トランザクションのリカバリーについて説明します。トランザクションのリカバリーは予想外の障害発生においてトランザクションの整合性をサポートするための重要な機能です。JEUSのユーザーは、前述した関連設定とトランザクションのリカバリー手順について熟知する必要があります。

7.5.1. トランザクションのリカバリー手順

トランザクション・マネージャーはトランザクションの使用時に起こり得る障害の対策を持っている必要があります。障害対策の最も重要な目的は、実行中のトランザクションの整合性を保証することです。トランザクション・マネージャーがルート・コーディネーターなのかサブ・コーディネーターなのか、あるいはJEUS以外の外部トランザクション・マネージャーとの作業中なのかどうかによってトランザクション・マネージャーのリカバリー方式は変わってきます。

下記の図は、トランザクション・マネージャーのリカバリー手順を簡単に示したものです。

[図 7.5] 単純化されたトランザクションのリカバリー手順



- 1)、2)トランザクション・マネージャーが障害から回復される場合、マネージャーはトランザクション・ログからリカバリーするトランザクションの情報を取得します。
- 3) その後ログの内容に従って該当のリソース・マネージャーにRecoverコマンドを送信します。
- 4) リソース・マネージャーは応答として自身が処理できなかったトランザクションのXIDをマネージャーに渡します。

- 5) トランザクション・マネージャーはこのXIDとログから取得した情報をもとに、リソース・マネージャーにコミットまたはロールバックのいずれかを送信します。

トランザクション・マネージャーの役割によってリカバリー方式は多少異なりますが、大筋は前述の通りです。以下ではリカバリーに参加する各要素について簡単に説明し、関連設定を行う方法について説明します。

トランザクション・マネージャーごとのリカバリー手順

トランザクション・マネージャーはトランザクションをリカバリーするための中心的な役割を果たします。以下では、トランザクション・マネージャーの役割によるリカバリー方式の違いについて説明します。

- ルート・コーディネーター

トランザクション・マネージャーがルート・コーディネーターの場合は、[\[図 7.5\]](#)で説明したように動作します。

- サブ・コーディネーター

サブ・コーディネーターは、ルート・コーディネーターに1つのリソース・マネージャーのように登録されます。

サブ・コーディネーターはすべての決定をルート・コーディネーターから受けるので、自身に登録されたリソース・マネージャーとログファイルのXIDを取得する作業のみを行います。その後、XIDをルート・コーディネーターに送って決定を待ちます。ルート・コーディネーターから決定が渡されると、その内容をリソース・マネージャーに渡してリカバリーを完了します。

- 外部トランザクション・マネージャーとの作業

JEUSトランザクション・マネージャーはルート・コーディネーターとして動作し、外部トランザクション・マネージャーはJEUSを1つのリソース・マネージャーとして認識します。したがって、トランザクション決定以外のすべての作業を行います。外部トランザクション・マネージャーがJEUSにRecoverコマンドと決定を出すと、それに応じてリカバリーを完了します。

自動トランザクション・リカバリー

自動トランザクション・リカバリー機能は、クラスター環境で現在のトランザクション・マネージャーが異常終了すると、他のサーバーのトランザクション・マネージャーが自動的にインダウト・トランザクションをリカバリーする機能です。この機能が正常に動作するには、サーバー間でクラスターが設定されている必要があり、リカバリーするサーバーが異常終了したサーバーのトランザクション・ログにアクセスできる必要があります。




以下では、WebAdminとコンソール・ツールを使って自動トランザクション・リカバリー機能を設定する手順について説明します。この設定はサーバーを再起動せずに変更内容を適用できます。

- WebAdminの使用

WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバーを選択すると、サーバー設定画面に移動します。

サーバー設定画面で**[Tm Config]**メニューを選択して、トランザクション設定画面に移動します。設定画面で「**Automatic Recovery**」項目にチェックして**[確認]**ボタンをクリックします。

[図 7.6] WebAdminによる自動トランザクション・リカバリーの設定

| | | |
|---|---|--------------------------------|
|  動的設定  必須項目 | | 確認 再設定 削除 |
| Active Timeout | <input type="text" value="600000"/> [デフォルト: 600000] グローバルトランザクションが開始されたら、設定した時間内にコミットが実行される必要があります。そうでない場合は、トランザクションマネージャがロールバックさせます。 | |
| Automatic Recovery  | <input checked="" type="checkbox"/> [デフォルト: false] クラスタリング環境において、現在のトランザクションマネージャに障害が生じた場合、別の場所で自動的にndoubtトランザクションを復旧する機能を使用するかどうかを設定します。この機能が正常に動作するためには、クラスタリングが設定された上で、Failed TMのログディレクトリを別の場所でアクセスできるように設定する必要があります。 | |

● コンソール・ツールの使用

コンソール・ツールで**modify-transaction-manager**コマンドを使って自動トランザクション回復機能を有効にできます。このコマンドについての詳しい説明は、『*JEUS リファレンスガイド*』の「4.2.13.1. modify-transaction-manager」を参照してください。

```
[DAS]domain1.adminServer>modify-transaction-manager server1 -automaticRecovery true
Successfully performed the MODIFY operation for transaction of server (server1), but all
changes were non-dynamic. They will be applied after restarting.
Check the results using "show-transaction-manager server1 or modify-transaction-manager
server1"
```

7.5.2. リカバリー関連のログ・ファイル

トランザクションのリカバリーに使用されるログ・ファイルは次の2つがあります。

- トランザクションの実行状況を記録したログ
- 使用したXAリソースに関するログ

ログ・ファイルは基本的にSERVER_HOME/.workspace/tmlogの下位ディレクトリーに生成されます。SERVER_HOMEの場所は、「[1.5. サーバー・ディレクトリー構造](#)」を参照してください。

ログ・ファイルの場所は、上で説明した自動トランザクション回復のために複数のサーバーがアクセスできる場所にログを残したい場合、共用ディレクトリーを指定します。回復の不要なトランザクションが原因で起動中に回復に関する問題が発生したら、このログをバックアップした後 JEUSを再実行します。

以下では、WebAdminとコンソール・ツールを使って回復関連ログ・ファイルを設定する手順について説明します。

● WebAdminの使用

WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。サーバーを選択すると、サーバー設定画面に移動します。

サーバー設定画面で**[Tm Config]**メニューを選択して、トランザクション設定画面に移動します。設定画面の**詳細設定**で「Tx Log Dir」項目を設定し、**[確認]**ボタンをクリックします。

[図 7.7] WebAdminによるトランザクション・ログ・ディレクトリーの変更

動的設定

必須項目

確認

再設定

削除

| | | |
|--|-------------------------------------|--|
| Active Timeout | <input type="text" value="600000"/> | |
| <small>[デフォルト: 600000] グローバルトランザクションが開始されたら、設定した時間内にコミットが実行される必要があります。そうでない場合は、トランザクションマネージャがロールバックさせます。</small> | | |
| Automatic Recovery | <input type="checkbox"/> | |
| <small>[デフォルト: false] クラスタリング環境において、現在のトランザクションマネージャに障害が生じた場合、別の場所で自動的にndoubtトランザクションを復旧する機能を使用するか否かを設定します。この機能が正常に動作するためには、クラスタリングが設定された上で、Failed TMのログディレクトリを別の場所でアクセスできるように設定する必要があります。</small> | | |

詳細設定

すべてを開く

| | | |
|--------------------|---|----|
| Prepare Timeout | <input type="text" value="120000"/> | ms |
| Prepared Timeout | <input type="text" value="60000"/> | ms |
| Commit Timeout | <input type="text" value="240000"/> | ms |
| Recovery Timeout | <input type="text" value="120000"/> | ms |
| Incomplete Timeout | <input type="text" value="86400000"/> | ms |
| Tx Log Dir | <input type="text" value="Users/user1/shared/txlog"/> | |

● コンソール・ツールの使用

コンソール・ツールで**modify-transaction-manager**コマンドを使ってトランザクション・ログ・ディレクトリーを変更できます。このコマンドについての詳しい説明は、『*JEUS リファレンスガイド*』の「4.2.13.1. modify-transaction-manager」を参照してください。

```
[DAS]domain1.adminServer>modify-transaction-manager server1 -txLogDir
/Users/user1/jeus/domain1/tmlogs
Successfully performed the MODIFY operation for transaction of server (server1) ,but ALL
changes were non-dynamic. They will be applied after restarting.
Check the results using "show-transaction-manager server1 or modify-transaction-manager
server1"
```


7.5.3. リカバリー関連の設定

リカバリーに関連する設定は以下のとおりです。システム・プロパティで下記の内容をサーバーのJVM設定に追加します。追加する方法は、『JEUS ドメインガイド』の「3.6.2. サーバーのJVM設定の変更」を参照してください。

システム管理者が直接リカバリーする場合や性能の向上が必要な場合、リカバリーのためのロギングを防ぐこともできます。

```
-Djeus.tm.noLogging=true
```

リカバリーの途中、何らかの問題によって失敗すると、トランザクション・マネージャーはそのリソースに対して再びリカバリーを試みます。2分ごとにリカバリーが試みられ、ユーザーはトライ回数を指定することができます。

トライ回数を指定するために、次のプロパティをスクリプトに追加します。(デフォルト値: 30)

```
-Djeus.tm.recoveryTrial=<number of trial>
```

サーバーを起動するときにリカバリーするトランザクションがあれば、STANDBYになる前にリカバリーを始めます。リカバリーの途中で失敗したリソースがあれば、バググランドでリトライします。

このとき、リトライする間隔を指定できます。(デフォルト値: 120000(2分)、単位: ms)

```
-Djeus.tm.recoveryInterval=<time in milliseconds>
```

TMログ・ファイルがなくなってリカバリーが不可能な状況の場合、JEUSの起動は失敗します。リカバリーが不要な状況の場合、ファイルが破損したことをJEUSが検知した後、TMログ・ファイルをすべて削除するように初期化させた後に正常に起動します。(デフォルト値 : false)

```
-Djeus.tm.ignore.broken.log.file=<true or false>
```

7.5.4. リソース・マネージャーの障害

リソース・マネージャーで障害が発生する場合、システム・マネージャーはコンソール・ツールを使って手動でリカバリーします。その理由は、JEUSトランザクション・マネージャーは、リソース・マネージャーがトランザクション処理が実行できる状態なのか否かを把握できないためです。

リソース・マネージャーの問題を解決した後、コンソール・ツールの**recover-transactions**コマンドを実行してリカバリーを進めます。コンソール・ツールの詳しい使用方法については、『JEUS リファレンスガイド』の「4.2.13.2. recover-transactions」を参照してください。

7.6. トランザクション・プロファイル機能

JEUSTランザクション・マネージャーはそれぞれのトランザクションの重要な時点で、ユーザーのコールバックを実行できる機能を提供します。クライアントでは使用できず、サーバーでのみ使用できます。

以下で定義されたインターフェースを継承したユーザー用のプロファイル・リスナーを実装します。

[例 7.8] <<CoordinatorProfileListener.java>>

```
package jeus.transaction.profile;

import jeus.transaction.info.TransactionInfo;

public interface CoordinatorProfileListener extends ProfileListener {

    public void beforePrepare( TransactionInfo info );

    public void afterPrepare( jeus.transaction.info.TransactionInfo info );

    public void beforeSetDecision( TransactionInfo info );

    public void afterSetDecision( TransactionInfo info );

    public void beforeCommit( TransactionInfo info );

    public void afterCommit( TransactionInfo info );

    public void beforeOnePhaseCommit( TransactionInfo info );

    public void afterOnePhaseCommit( TransactionInfo info );

    public void beforeRollback( TransactionInfo info );

    public void afterRollback( jeus.transaction.info.TransactionInfo info );

    public void beforeDestroy( TransactionInfo info );

    public void afterDestroy( TransactionInfo info );

    public void beforeActiveTimeout( TransactionInfo info );

    public void afterActiveTimeout( TransactionInfo info );

}
```

[例 7.9] <<TransactionInfo.java>>

```
package jeus.transaction.info;

import javax.transaction.xa.Xid;

public interface TransactionInfo {
```

```

public static final int JEUS_SPECIFIC_CURRENT_FORMAT_XID = 303077;
public static final int UNSPECIFIED_TIME = -1;

/**
 * このトランザクションのXIDを返します。返されたXIDはJEUS内部実装のXIDであり、
 * serializableです。
 * 以降、このXIDをstringで表現したい場合、XidToString.getXidHexString()を
 * 利用します。
 *
 * @return 該当するtxのXID実装
 */
public Xid getXid();

/**
 * トランザクション・マネージャーの情報をstring形式で返します。この情報でTXの
 * TMサーバー情報(IP、ポートなど)を確認できます。
 *
 * @return 該当するtxのトランザクション・マネージャーの情報
 */
public String getCoordinator();

/**
 * 外部で開始されたTXの場合、外部のXIDを返します。返されたXIDはJEUS内部実装のXIDに
 * 交替され、serializableです。
 *
 * @return 該当するtxのXID実装
 */
public Xid getExternalXid();

/**
 * active timeout設定を返します。単位はmsです。
 *
 * @return 該当するtxのアクティブ・タイムアウト
 */
public long getTimeout();

/**
 * TXが開始されてから経過した時間を返します。単位はmsです。
 *
 * @return 該当するtxの経過時間
 */
public long getElapseSinceBegin();

/**
 * TXの状態をstring形式で返します。
 *
 * @return 該当するtxのstring status値
 */
public String getStatus();

/**
 * TXにenlistされたXAResourceをString形式で返します。
 *

```

```

    * @return 該当するtxにenlistされたXAResourceの情報
    */
    public String[] getXaResources();

    /**
     * TXを開始したスレッドを返します。
     * TXを伝播された場所ではこの情報を取得できません。
     *
     * @return 該当するtxを開始したスレッド
     */

    public Thread getBeginThread();
}

```

実装したリスナーをシステム・プロパティで指定します。複数のリスナー・クラスを空白、セミコロン(;)、またはコンマ(,)をセパレータとして指定できます。

```
-Djeus.tm.profile.classes=test.profile.MyCoordinatorListener1,test.profile.MyCoordinatorListener2"
```

7.7. IP帯域が異なるサーバー間のトランザクション通信問題

JEUSTランザクション・マネージャー間で通信を行なうときは、IPアドレスを利用して接続します。しかしこの場合、NAT環境内にあるJEUSTランザクション・マネージャーと外部環境にあるJEUSTランザクション・マネージャーの相互のIP帯域が合わず、通信できない場合が発生します。

この問題は、リアルIPを仮想IPにマッピングするプロパティを指定することで解決できます。JEUS_HOME/templates/nat.properties.templateを参照してIPマッピング・プロパティ・ファイルを作成した後、以下のようにシステム・プロパティとして指定します。

```
-Djeus.tm.net.address-mapping-properties=<full path of properties file>
```

第8章 ロギング

本章では、JEUSロギング・システムのJEUSロガー構造と各ロガーおよびハンドラーの設定方法、ログ・メッセージの内容について説明します。

8.1. 概要

JEUSロギングは、JEUSの実行中にシステムで行われた一連の作業内容を順序どおり保存、記録する機能です。システム管理者はこの機能を利用して、JEUSで実行された様々な作業や発生したエラーについて把握し、それに対応することができます。

JEUSは、システムやアプリケーションで発生する様々な状況をログで出力します。JEUSは、Java SEの標準Logging API(`java.util.logging`)を使用するため、ロギング・システムの構造や設定方法もJava SE Logging APIに準拠し、ロガーやハンドラー、フォーマッターの構造をそのまま反映しています。また、開発者がLogging APIを利用してJEUSのロガーを必要に合わせて使用することもできます。

JEUSのロガー・システムはJava SE Logging APIをベースに実装されたため、開発者はこのAPIを使ってJEUSロギング・システムをカスタマイズできます。

JEUSロガーの種類には、JEUSランチャーやJEUSサーバーのロガー、そしてWEBエンジンで使用されるアクセス・ロガーがあります。各ロガーは、「jeus」ロガー(「jeus」はロガーの名前、以下、jeusロガー)を基準に生成されます。JEUSサーバーはjeusロガーを基本的に生成します。下位に生成される様々なモジュールは、jeus.ejbのようにjeus下位のロガーを作成して使用します。

参考

jeusロガーは、JEUSで使用される最上位ロガーです。このロガーは、ユーザーが設定しなくてもデフォルトで生成されます。

JEUSの設定ファイルで設定可能なロガーには、以下のようなものがあります。

- JEUSロガー
- ユーザー・ロガー
- Webエンジンで使用するアクセス・ロガー
- JEUSロガーの下位ロガー

- Javaロガー

ハンドラーは、実際にロガーが出力するログ・メッセージをある対象に記録する役割をします。したがって、ロガーは常にハンドラーを伴います。JEUSではログ・メッセージの出力対象によって、次の4種類のロガーにハンドラーを設定することができます。設定されたハンドラーによってログ・メッセージが記録されます。ハンドラーの中で最も普遍的に使用されるのはファイル・ハンドラーです。

| 区分 | 説明 |
|------------|----------------------------------|
| ファイル・ハンドラー | ロガーによりロギングされるログ・メッセージをファイルに記録します |
| SMTPハンドラー | ログ・メッセージをメールに記録します |
| ソケット・ハンドラー | ログ・メッセージを指定したIPに記録します |
| ユーザー・ハンドラー | ユーザーが作成したハンドラーでログ・メッセージを記録します |

その他のハンドラーの説明と各ロガーのハンドラーをJEUS設定ファイルに設定する方法については、「[8.3.1. ロガー情報の確認](#)」で説明します。ロギング・システムに関する基本的な内容についてはJava SEのLogging APIを参照してください。

参考

1. JEUS 7からはコンソール・ハンドラーは設定できません。
2. JEUS 7からはサーバー・プロセスのコンソールは存在しません。ランチャーでサーバーを起動するため、ランチャー・プロセスのコンソールはありますが、サーバー・プロセスのコンソールはありません。
3. サーバーで発生するログ・メッセージをコンソールに出力するには、ランチャー・プロセスがサーバーがシャットダウンされるまで終了されない必要があります。サーバーを起動時に-verboseオプションを与えて起動すると、ランチャー・プロセスはサーバーがダウンするまで終了されずに、サーバーで発生するログ・メッセージをパイプから読み込み、コンソールに出力します。
4. JEUS 7 Fix#4バージョンからは非同期ロギング(Asynchronous logging)をサポートします。ログを呼び出したワーカー・スレッドでロギングを処理するのではなく、専用のロガー・スレッドで処理することにより、JEUSの全般的な性能向上を図ります。この方式では、ファイル・ハンドラーのほか、ロギングが遅延される可能性のあるハンドラーは提供しません。他のハンドラーを使用したい場合は、jeus.logging.useAsyncオプションをfalseに設定します。また、非同期ロギング方式では、jeus.access.logging.skip.when.busyオプションをtrueにして、高性能を必要とするWebワーカーが過剰なロギングによって待機することを自動的に検出し、アクセス・ロガーが待機することを防ぎます。

8.2. JEUSロガーの基本構造

本節では、ロガーを使用するための基本概念と設定方法について説明します。

8.2.1. 概要

特にサーバーにロガーを設定しなくても、最上位ロガーのjeusロガーはデフォルトで存在します。ファイル・ハンドラーを使用し、毎日ログ・ファイルを切り替えて使用します。ログ・ローテーションを含むファイル・ハンドラーの詳細については、「[8.3.1. ロガー情報の確認](#)」で説明します。

参考

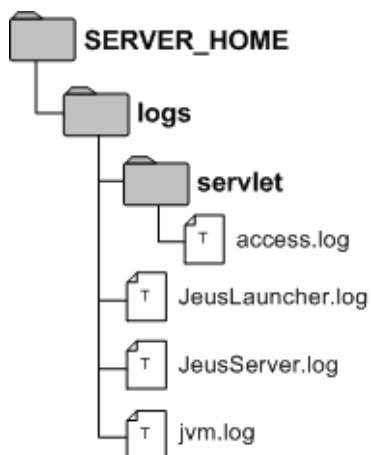
jeusロガーは、ロガー削除コマンドで削除できません。jeusロガーを削除してしまうと、他のロガーを設定していない場合にサーバーにログ・メッセージがロギングされないため、運用上問題になり得ます。サーバーでログを残したくない場合は、jeusロガーのレベルをOFFに設定します。

ドメインに登録されているサーバーの削除も、該当サーバーのlogsディレクトリーの削除を意味しません。ロガーは管理者がいつでも閲覧できる重要な情報であるため、不要と判断して削除するのは管理者の役割です。

ログ・ディレクトリーの構造

ロガーでファイルの名前を別途指定していない場合、JEUSでは決まった場所にログ・ファイルを作成します。ログ・ディレクトリーの構造は以下のとおりです。

[図 8.1] ログ・ディレクトリー



以下は、ログ・ディレクトリーに基本的に生成されるログ・ファイルについての説明です。

servlet/access.log

Webアプリケーションの要求に対するアクセス・ログ・ファイルです。基本的にサーバーがすべてのWebアプリケーションに要求した内容が記録されます。Webエンジンにバーチャル・ホストが設定された場合は、servletディレクトリーの下位にバーチャル・ホストの名前でディレクトリーが生成され、その下位にaccess.logファイルを作成してバーチャル・ホストに要求された情報がロギングされます。

JeusLauncher.log

ランチャーでサーバーの起動のために残す情報とサーバーを起動時に発生するログ・メッセージをロギングします。

JeusServer.log

サーバーでロギングする基本ログ・ファイルです。サーバーにロガー関連設定を行わなかった場合、jeusロガーを含むその下位のすべてのロガーに対する基本ログ・ファイルとなります。

jvm.log

サーバーJVMで発生するgcログやスレッド・ダンプなどが記録されます。このファイルは、ランチャーでサーバーを起動するとき、特定のJVMオプションを与えると生成されます。

ランチャーでサーバーを起動するとき、JVMオプションに下記を追加します。

```
-XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput -XX:LogFile=SERVER_HOME/logs/jvm.log
```

JVMログを別途のファイルに記録するように設定する理由は、サーバーがバックグラウンド・プロセスで動作し、サーバーのスレッド・ダンプがそれ以上コンソールに残らないためです。このログの場所を変更するには、-XX:LogFileオプションをサーバーのJVMオプションに追加すると、基本設定が無視され、ユーザーの設定に従います。

ユーザー定義GCログファイル

Oracleの場合は「-Xloggc:」、IBMの場合は「-Xverbosegclog:」オプションを使用してGCログファイルを指定することができます。そのとき、JEUSでは指定されたファイルの後ろに自動でタイム・スタンプを付けて、JEUSを起動するたびに作成されるGCログ・ファイルと区別できるようにします。

特定のログ・ディレクトリーの設定

サーバーで作成するすべてのログ(ローテーション・バックアップ・ログを含む)を特定のディレクトリーに保存したい場合、WebAdminを使って以下のように設定します。

WebAdminの左のメニューで**[Servers]**を選択して照会されたサーバー・リストで特定のサーバーをクリックすると、サーバー設定画面に移動します。**[Basic] > [Basic Info]**メニューの「**Log Home**」項目にディレクトリー・パスを設定します。ただし、個別ログのファイル生成パスが絶対パスで設定されている場合は、そのパスが優先されます。

[図 8.2] Log Homeパスの設定

The screenshot shows the 'Server' configuration window. At the top, there's a 'HISTORY' button and a search bar. Below that, a message says 'ドメイン内で使用するJEUSサーバの詳細設定を定義します。' (Define detailed settings for the JEUS server used within the domain). There's a 'ヘルプ' (Help) button. The main area has tabs for 'Basic', 'Resource', and 'Engine'. Under 'Basic', there's a sub-menu with 'Basic Info', 'Res Ref', 'Naming Server', 'System Thread Pool', 'System Logging', 'User Logging', and 'Tm Config'. The 'Basic Info' tab is active, showing a table of settings. The 'Log Home' row is highlighted, showing a text input field with a placeholder path and a description: 'JEUSサーバで生成するログのデフォルトパスを指定します。' (Specify the default path for logs generated by the JEUS server). Other rows include 'Auto Generated' (checkbox), 'Name' (text field 'adminServer'), and 'Node Name' (dropdown menu 'node1').

| 動的設定 | 必須項目 | このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 |
|----------------|--------------------------|---|
| Auto Generated | <input type="checkbox"/> | サーバが動的に自動作成されたのかを表示します。ユーザが設定できない項目です。trueの場合、クラスタが削除されると、該当するサーバも自動的に削除されます。 |
| Name | * | adminServer サーバ名です。 |
| Log Home | | JEUSサーバで生成するログのデフォルトパスを指定します。同パスが設定されていても、ロガーのファイルハンドラに設定されているパスが絶対パスの場合にはそのパスが優先されます。デフォルトログのみならず、ローテーションされたバックアップログが生成されるパスにも同様に適用されます。 |
| Node Name | | node1 サーバが属するノード名を指定します。ノードにはマシン情報、ホスト情報などが含まれます。 |

8.2.2. ランチャー・ロガー

ランチャーは、サーバーを起動の際に使用するプロセスです。実行スクリプトまたはコマンドでサーバーを起動すると、ランチャー・プロセスが実行されます。ランチャーは設定ファイルを読み込んでサーバーを起動し、サーバーが起動の際に発生するログ・メッセージをJeusLauncher.logファイルとコンソールに残します。ランチャー・プロセスは一般的にサーバーの起動が完了すると終了され、サーバーの起動ログだけがランチャー・ロガーによって残されます。しかし、サーバーを実行するときに-verboseオプションを与えた場合は、サーバーの起動が完了した後もランチャー・プロセスが終了されず、サーバーが終了するまで残されるすべてのログ・メッセージをランチャーのコンソールとファイルに記録します。

サーバーでサーバー・ロガーによってログ・メッセージをロギングしているのに、ランチャー・プロセスで別途ロギングする理由は、サーバーの起動中にサーバーでロギングできない状況で発生し得るエラーに備えるためです。たとえば、設定ファイルのエラーなどの理由でサーバーを起動できなかったり、サーバーが起動されたものの、ブート失敗によりサーバーのロガーが初期化される前に終了した場合は、ブート失敗の原因を把握できるログ・メッセージがサーバー・ロガーによって残されません。そのような場合、ランチャーが残したサーバーのブート・ログを確認してサーバーのブート時に発生したエラーを解決することができます。ランチャーに関する詳しい内容は、「1.6. ランチャー」を参照してください。

[例 8.1] 設定ファイルのエラーによる起動失敗時にランチャーが残したログ・メッセージ

```
[2016.08.24 12:38:52][0] [launcher-1] [XmlValidationEventHandler] [FATAL_ERROR]
Invalid content was found starting with element <system-clustering-framework> (of parent
element <domain>). One of '{password-validator, admin-server-name}' is expected.

[2016.08.24 12:38:52][0] [launcher-1] [SERVER-0522] An exception occurred while processing
```

```
[domain.xml].
<<__Exception__>>
jeus.xml.binding.JeusJAXBException: Unmarshalling the XML descriptor failed: domain.xml
class org.xml.sax.SAXParseException :
cvc-complex-type.2.4.a: Invalid content was found starting with element
'system-clustering-framework'. One of
'{"http://www.tmaxsoft.com/xml/ns/jeus":password-validator,
"http://www.tmaxsoft.com/xml/ns/jeus":admin-server-name}' is expected..
    at jeus.xml.binding.BindingHelper.getDescriptor(BindingHelper.java:96)
    at jeus.service.descriptor.DescriptorFile.getDeploymentDescriptor(DescriptorFile.java:210)
    at
jeus.service.descriptor.JEUSDomainDescriptorFile.getConfigDescriptor(JEUSDomainDescriptorFile.java:54)
    at jeus.launcher.Launcher.initDomainType(Launcher.java:222)
    at jeus.launcher.Launcher.readDescriptor(Launcher.java:210)
    at jeus.launcher.Launcher.start(Launcher.java:105)
    at jeus.launcher.Launcher.main(Launcher.java:58)
```

8.2.3. サーバー・ロガー

サーバー・ロガーは、サーバーの運用中に発生するログ・メッセージをロギングします。サーバーのログ・メッセージはサーバーの運用中に実行された作業情報を示します。

JEUSの基本ロガーは最上位ロガーのjeusロガーであり、jeusロガーの下位ロガーは、jeusロガーのハンドラーによりロギングされます。サーバーでロギングされるロガーは動的に追加、削除、変更することができます。ハンドラーも動的な追加、削除、変更が可能です。

ログ・メッセージのフォーマットはフォーマッター・クラス(formatter class)を使用して変更できます。何も設定しない場合、JEUSが提供する基本フォーマッターのjeus.util.logging.SimpleFormatterが使用されます。その形式は以下のとおりです。

- [時間] [レベル] [スレッド情報] [ログ・メッセージID] ログ・メッセージ

| 項目 | 説明 |
|-----|--|
| 時間 | 「年.月.日 時間:分:秒」形式で出力します |
| レベル | ログ・レベルをマッピングされる数字で出力します <ul style="list-style-type: none"> – 0: SEVERE – 1: WARNING – 2: INFO – 3: CONFIG – 4: FINE |

| 項目 | 説明 |
|------------|--|
| | <ul style="list-style-type: none"> – 5: FINER – 6: FINEST – 7: ALL |
| スレッド情報 | ログ・メッセージをロギングするスレッド情報です。ロギングするプロセスとスレッド番号で表現され、この2つはハイフン(-)で区分されます。スレッド情報が同一のログ・メッセージは同じスレッドでロギングしたものです |
| ログ・メッセージID | ログを出力するモジュール情報です。モジュール名とメッセージ番号で表現され、この2つはハイフン(-)で区分されます。ログに出力される各モジュールの名前は、 「8.2.7. ログ・メッセージのモジュール名」 を参照します |
| ログ・メッセージ | 運用中に発生した作業内容を示すログ・メッセージです |

参考

ランチャー・プロセスでロギングされるログ・メッセージも、ロギングされるプロセスだけが異なり、フォーマットはサーバーでロギングするログ・フォーマットと同じです。

以下は、実際にJEUSサーバーに出力されるログ・メッセージの例です。

[例 8.2] サーバー・ログ・メッセージの出力例

```
2016.08.24 10:36:17][0] [adminServer-1] [SERVER-0000] Version information - JEUS 8.0 (Fix#0)
(8.0.0.0-b28).
[2016.08.24 10:36:17][0] [adminServer-1] [SERVER-0001] java.specification.version=[1.7],
java.runtime.version=[1.7.0_75-b13], vendor=[Oracle Corporation]
[2016.08.24 10:36:17][2] [adminServer-1] [SERVER-0002] Domain=[domain1], Server=[adminServer],
baseport=[9736], pid=[9316]
[2016.08.24 10:36:17][2] [adminServer-1] [SERVER-0004] The current system time zone : Korea
Standard Time
[2016.08.24 10:36:17][2] [adminServer-1] [WSVC-3805] JAX-RS Implementation libraries are
added to the classpath of RootClassLoader: [library-ref:name=jax-rs, specVersion=2.0.1,
implVersion=0]
[2016.08.24 10:36:17][2] [adminServer-1] [SERVER-0571] All JEUS system properties have been
confirmed.
[2016.08.24 10:36:17][2] [adminServer-1] [SERVER-0568] Service address='0.0.0.0:9736',
hostname='hahehihowin7', representation ip='192.168.15.45'
[2016.08.24 10:36:17][2] [adminServer-1] [SERVER-0561] The default RMI export port = 9743
and bind address = /0.0.0.0.
```

まず最初のログ・メッセージは、2012年5月9日午後15時2分37秒に出力されたレベル2(INFOレベル)のメッセージであることが分かります。このログ・メッセージはadminServerというサーバーの1番スレッドによりロギングされ、SERVERモジュールの0000番メッセージが出力されたことが分かります。このログ・メッセージは

JEUSサーバーが起動を開始したことを知らせる、サーバーの最初のメッセージです。JEUSのバージョン情報と一緒に出力されます。

他のログ・メッセージも最初のメッセージと同じプロセスでロギングされ、SERVERモジュールのメッセージが出力されたことが分かります。上記のログ・メッセージはサーバーの起動時に発生したもので、Javaバージョン、プロセスID、タイムゾーン、ネットワーク情報といった環境情報を出力しています。

JEUSではサーバー・ロガーのフォーマッターとして基本フォーマッターの`jeus.util.logging.SimpleFormatter`の他に、`jeus.util.logging.SimpleMillisFormatter`を追加で提供します。このフォーマッターは基本フォーマッターとほぼ同じフォーマットで、ミリ秒まで含む時間を出力します。

以下は、`SimpleMillisFormatter`を設定時に発生するログ・メッセージの例です。フォーマッターの設定に関する詳細は、「[8.3. ロギングの設定](#)」を参照してください。

[例 8.3] ログ・フォーマッターを使用したサーバー・ログ・メッセージの出力例

```
[2016.08.24 10:42:51.385][0] [adminServer-1] [SERVER-0000] Version information - JEUS 8.0
(Fix#0) (8.0.0.0-b28).
[2016.08.24 10:42:51.385][0] [adminServer-1] [SERVER-0001] java.specification.version=[1.7],
java.runtime.version=[1.7.0_75-b13], vendor=[Oracle Corporation]
[2016.08.24 10:42:51.387][2] [adminServer-1] [SERVER-0002] Domain=[domain1],
Server=[adminServer], baseport=[9736], pid=[13364]
[2016.08.24 10:42:51.387][2] [adminServer-1] [SERVER-0004] The current system time zone :
Korea Standard Time
[2016.08.24 10:42:51.560][2] [adminServer-1] [WSVC-3805] JAX-RS Implementation libraries are
added to the classpath of RootClassLoader: [library-ref:name=jax-rs, specVersion=2.0.1,
implVersion=0]
[2016.08.24 10:42:51.560][2] [adminServer-1] [SERVER-0571] All JEUS system properties have
been confirmed.
[2016.08.24 10:42:51.568][2] [adminServer-1] [SERVER-0568] Service address='0.0.0.0:9736',
hostname='hahehihowin7', representation ip='192.168.15.45'
[2016.08.24 10:42:51.580][2] [adminServer-1] [SERVER-0561] The default RMI export port = 9743
and bind address = /0.0.0.0.
```

8.2.4. アクセス・ロガー

アクセス・ロガーは、ユーザーのアプリケーション要求がどのように処理されたかを残すために使用されます。Webアプリケーションへの要求情報を記録します。本節では、Webエンジンのアクセス・ロガーについて説明します。

Webエンジンのアクセス・ロガー

Webエンジンのアクセス・ロガーは、Webエンジンが処理したすべての要求を記録します。アクセス・ロガーによるログは、Webアプリケーションにアクセスした情報と記録する内容を指定することで、管理者が必要な情報を取得できます。Webエンジンに要求が送られ、すべての要求を処理して応答を返した後、アクセス・ロガーは設定された情報を記録します。

Webエンジンのアクセス・ロガーによって記録されるログは、基本的に共通ログ形式(Common Log Format)をサポートします。この形式はApacheなどの多くのところで共通して使用しているので、ログを分析するとき多くの情報を得ることができます。特に、ログ分析ツールでは共通ログ形式を分析するツールがたくさんあるので、これらのツールを利用すると、Webエンジンのアクセス・ログを分析するのに大いに役立ちます。

注意

JEUS 6バージョンまでは共通ログ形式はサポートされていません。

また、Webエンジンではバーチャル・ホストを設定することができますが、バーチャル・ホストを設定した場合、バーチャル・ホスト別にアクセス・ロガーを設定して別途のアクセス・ログを収集することができます。このとき、バーチャル・ホストのアクセス・ログは以下のファイル名で保存されます。

```
SERVER_HOME/logs/servlet/バーチャル・ホスト名/access.log
```

Webエンジンのアクセス・ログの基本形式でアクセス・ログを残した場合、以下のようなログがaccess.logに残ります。

[例 8.4] access.logの記録例

```
192.168.15.57 [29/Aug/2016:17:37:02 +0900] "GET /example/test1.jsp HTTP/1.1" 200 5 38
```

上記のログは、/example/test1.jsp要求の結果、192.168.15.57 IPから38バイトの応答を「29/Aug/2016:17:37:02 +0900」時間に正常に送信したことを示しています。より詳しいログ形式は共通ログ形式の文書を参照してください。

ログ形式の変更やその他のWebエンジンのアクセス・ロガーの設定については、『*JEUS Webエンジンガイド*』の1.6.10節「アクセス・ログの基本設定」を参照してください。

8.2.5. ユーザー・ロガー

JEUSの各サーバーごとに提供されるユーザー・ロガー(user logger)は、開発者が別途のロガーを使用する必要なく、JEUSが提供するロガーを使用できるようにします。Java SE Logging APIのjava.util.logging.logger APIを使ってユーザー・ロガーを使用できます。

8.2.6. ロガー・リスト

- EJB関連

| ロガー | 説明 |
|------------------|------------------------|
| jeus.ejb.bean | EJBホーム/オブジェクト・スタブ関連ロガー |
| jeus.ejb.cluster | EJBクラスタリング関連ロガー |

| ロガー | 説明 |
|----------------------|------------------------|
| jeus.ejb.compiler | EJBスタブ・コンパイラー関連ロガー |
| jeus.ejb.connector | MDBとリソース・アダプター関連ロガー |
| jeus.ejb.container | EJBコンテナ関連ロガー |
| jeus.ejb.ejbserver | EJBエンジン関連ロガー |
| jeus.ejb.interop | EJB CORBA連動関連ロガー |
| jeus.ejb.persistence | CMP関連ロガー |
| jeus.ejb.schema | EJB QL関連ロガー |
| jeus.ejb.timer | EJB Timer関連ロガー |
| jeus.ejb.transaction | EJBトランザクションおよび同期化関連ロガー |
| jeus.ejb.webserver | EJB Class FTP関連ロガー |
| jeus.ejb.util | その他のロガー |

- JPA関連

| ロガー | 説明 |
|----------------------------|--|
| jeus.persistence | JPAモジュールのルート・ロガー |
| jeus.persistence.provider | TopLink Essentials(デフォルト・プロバイダー)のルート・ロガー |
| jeus.persistence.container | JPAのコンテナ・ロガー |

- サーブレット関連

| ロガー | 説明 |
|-------------------------|----------------------|
| jeus.servlet.common | サーブレット共通モジュール関連ロガー |
| jeus.servlet.connection | コネクタ関連ロガー |
| jeus.servlet.connector | 非同期I/Oコネクタ関連ロガー |
| jeus.servlet.deployment | デプロイメント関連ロガー |
| jeus.servlet.engine | サーブレット主要処理手順関連ロガー |
| jeus.servlet.filter | フィルター関連ロガー |
| jeus.servlet.jsp | JSP関連ロガー |
| jeus.servlet.listener | サーブレット・リスナー関連ロガー |
| jeus.servlet.loader | クラス・ローダー関連ロガー |
| jeus.servlet.property | プロパティ関連ロガー |
| jeus.servlet.servlets | JEUSで提供するサーブレット関連ロガー |
| jeus.servlet.util | ユーティリティで使用するロガー |

| ロガー | 説明 |
|----------------|-------------------------|
| jeus.websocket | WebSocketサーバー関連のすべてのロガー |
| jeus.webserver | Class FTPサービスで使用するロガー |

- セッション・マネージャー関連

| ロガー | 説明 |
|--------------------------|-----------------------------------|
| jeus.session | セッション・マネージャーのルート・ロガー、共通して使用されるロガー |
| jeus.session.distributed | 分散セッション・マネージャーのロガー |

- Webサービス関連

- JAX-RPC/SAAJロガー

| ロガー | 説明 |
|---------------------------|--|
| jeus.webservices.client | jeus.webservices.clientパッケージ(クライアント呼び出しフレームワーク)関連ロガー |
| jeus.webservices.encoding | SOAPのシリアライズ/デシリアライズ関連ロガー |
| jeus.webservices.message | SAAJおよびSOAPメッセージ関連ロガー |
| jeus.webservices.wsdl | WSDL処理関連ロガー |
| jeus.webservices | その他のWebサービス関連ロガー |

- UDDIロガー

| ロガー | 説明 |
|---------------------|---------------------------------------|
| jeus.uddi.datastore | データベース処理関連ロガー |
| jeus.uddi.function | UDDI APIメッセージ処理関連ロガー |
| jeus.uddi.judy | レジストリー・サーバー・エンジン関連ロガー |
| jeus.uddi.registry | トランスポート・レイヤー、要求/応答処理、レジストリー・エンジン関連ロガー |

- WS-*ロガー(JAX-RPCベース)

| ロガー | 説明 |
|----------------------|------------------|
| jeus.webservices.wss | ws-security関連ロガー |

– その他

| ロガー | 説明 |
|--|---------------------------------|
| jeus.xml.binding.webservices Helper | EWS(JSR109)で使用されるDDバインディング関連ロガー |

● トランザクション関連

| ロガー | 説明 |
|---------------------------|-----------------------------|
| jeus.transaction | トランザクション・マネージャーで全般的に使用するロガー |
| jeus.transaction.logging | 回復に使用されるリソース・ファクトリー関連ロガー |
| jeus.transaction.ots | OTS関連ロガー |
| jeus.transaction.recovery | トランザクション回復作業内容を記録するロガー |

● セキュリティー関連

| ロガー | 説明 |
|--------------------------|---------------------------|
| jeus.security | JEUSセキュリティ関連ロガー |
| jeus.security.impl.login | JEUSセキュリティ・ログイン・サービス関連ロガー |
| jeus.security.util | JEUSセキュリティ・ユーティリティ関連ロガー |

● その他の主要ロガー

| ロガー | 説明 |
|-------------------|--------------------------------------|
| jeus.classloader | JEUSクラス・ローディング関連ロガー |
| jeus.clustering | JEUSサーバー・クラスタリング関連ロガー |
| jeus.config | JEUS動的な設定変更関連ロガー |
| jeus.connector | J2EEコネクタ関連ロガー |
| jeus.converter | XMLコンバーター関連ロガー |
| jeus.ddinit | デプロイメント記述子(DD)初期化子(Initializer)関連ロガー |
| jeus.deploy | アプリケーション・デプロイメント関連ロガー |
| jeus.domain | ドメイン関連ロガー |
| jeus.filetransfer | 設定ファイル/アプリケーション・ファイルの送信関連ロガー |
| jeus.io | JEUSネットワークI/Oライブラリー関連ロガー |
| jeus.jdbc | JDBCコネクション・プール関連ロガー |
| jeus.jmx | JMX関連ロガー |

| ロガー | 説明 |
|------------------|-----------------------|
| jeus.jndi | JNDI関連ロガー |
| jeus.jnlp | JNLP関連ロガー |
| jeus.jtmax | JTmax関連ロガー |
| jeus.management | JMX MBeanフレームワーク関連ロガー |
| jeus.net | JEUS Network API関連ロガー |
| jeus.node | SSHノード・マネージャー関連ロガー |
| jeus.nodemanager | Javaノード・マネージャー関連ロガー |
| jeus.rmi | JEUS RMI関連ロガー |
| jeus.scheduler | JEUSスケジューラー関連ロガー |
| jeus.service | JEUSサービスMBean関連ロガー |
| jeus.weld | JEUS CDI関連ロガー |

8.2.7. ログ・メッセージのモジュール名

JEUSで使用するログ・メッセージは様々な情報を提供しています。その中でログ・メッセージの形式によって、モジュールの情報を出力するログ・メッセージ情報では各モジュールの名前を確認することができます。本節では、ログに出力される各モジュール名とそれに該当する実際のモジュールを説明します。

| モジュール名 | モジュール情報 |
|-----------|-------------------|
| Connector | コネクタ |
| Console | コンソール・コマンド |
| Config | 設定ファイルの同期化関連モジュール |
| CORBA | CORBA |
| CPOOL | コネクション |
| D_Session | 分散式セッション・サーバー |
| Deploy | アプリケーションのデプロイ |
| Domain | ドメイン |
| EJB | EJBエンジン |
| JDBC | JDBC |
| JMS | JMSエンジン |
| JMSC | JMSクラスタリング |
| JMSF | JMSフェイルオーバー |
| JMX | JMX |

| モジュール名 | モジュール情報 |
|--------------|--------------------------------------|
| JMXR | JMXリモート |
| JNDI.Common | JNDI共通 |
| JNDI.Context | JNDIコンテキスト |
| JNDI.Local | JNDIローカル・クライアント |
| JNDI.Remote | JNDIリモート・クライアント |
| JNSS | JNDIサーバー |
| JPA | JPA |
| JTMAX | JTmax |
| Launcher | JEUS Launcher (ランチャー・プロセス) |
| Network | JEUSネットワーク |
| NodeManager | ノード・マネージャー |
| OTS | OTS |
| SCF | SCf(JEUS System Clustering Framwork) |
| Scheduler | スケジューラー |
| Secutiry | JEUSセキュリティー |
| SERVER | サーバーの起動、終了、モニタリング関連モジュール |
| Session | セッション・サーバー |
| TM | トランザクション・マネージャー |
| TMRecovery | トランザクション・マネージャーの 回復 |
| UDDI | UDDI |
| WEB | サーブレット・エンジン |
| WebT | WebT |
| WebtobLight | WebtoB |
| WSS | Webサービス・セキュリティー |
| WSVC | Webサービス |

8.3. ロギングの設定

本節では、JEUSでのロギングの設定方法とカスタマイズ方法について説明します。

8.3.1. ロガー情報の確認

JEUSのロガーのうち、jeusロガーは設定がなくても基本的に存在します。しかし、jeusロガーを除く他のロガーは設定しない場合、上位ロガー(jeusロガー)のハンドラーを使用してログ・メッセージを出力します。jeus

ロガーは特に設定しない場合、ファイル・ハンドラーを使用します。したがって、サーバーにロガーを設定しなかった場合は、サーバーの運用中に発生するログ・メッセージはデフォルト・ログ・ファイルに残されます。デフォルトのログ・レベルはINFOです。jeusロガーは上位ロガーのハンドラーを使用しません。

参考

サーバーのログ・メッセージをコンソールで確認したい場合は、サーバーを起動するときに-verboseオプションを設定します。

-verboseオプションを設定すると、ランチャー・プロセスがサーバーを起動した後も終了されずに、サーバーが終了するまで動作しながら、サーバーのログ・メッセージをパイプから読み込んでコンソールに出力します。

以下は、コンソール・ツールとWebAdminを使ってサーバーに基本的に存在するjeusロガー情報を確認する例です。

● WebAdminの使用

WebAdminの左のメニューで[Servers]を選択して照会されるサーバー・リストで目的のサーバーをクリックします。[Basic] > [System Logging]メニューでjeusロガー情報を確認できます。

[図 8.3] WebAdminによるjeusロガー情報の確認

System Logging

HISTORY

サーバで使用するロガーについての設定です。

ヘルプ

BasicResourceEngine

Basic InfoRes RefNaming ServerSystem Thread PoolSystem LoggingUser LoggingTm Config

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

Name

jeus

ロガーに対して設定を適用するとき、該当するロガー名を指定します。ロガー名を確認したい場合は、ロガーページを参照してください。

Level

INFO

[デフォルト: INFO] ロガーのレベルを設定します。各レベルの意味については、Java SE logging APIのLevel Class Documentationを参照してください。

詳細設定

すべてを閉く

Use Parent Handlers

Filter Class

com.tmax.logging.filter.MyFilter

Formatter Class

jeus.util.logging.SimpleFormatter

com.tmax.logging.handler.MyHandler

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

Handlers

| Name | Type | Level | |
|-------------|------|--------|--------|
| fileHandler | file | FINEST | Delete |

第8章 ロギング 299

- コンソール・ツールの使用

コンソール・ツールで以下のように**log-level**、**list-log-handlers**コマンドを実行すると、jeusロガーとそれに登録されたハンドラーの情報を確認できます。コマンドのより詳しい使用方法は、『*JEUS リファレンスガイド*』の「4.2.3. サーバー管理関連コマンド」を参照してください。

[例 8.5] コンソール・ツールによるjeusロガー情報の確認

```
[DAS]domain1.adminServer>log-level -server server1 jeus
The logger[jeus] information for the server [server1]
Information about the logger[jeus].
=====
Logger Name : jeus
Level : INFO
Use Parent Handlers : false

+-----+-----+-----+
|           Handler Name           | Handler Type | Handler Level |
+-----+-----+-----+
| jeus.util.logging.ConsoleHandler@2569862 | ConsoleHandler | ALL           |
| FileHandler                         | FileHandler   | FINEST        |
+-----+-----+-----+
=====

[DAS]domain1.adminServer>modify-logger -server server1 jeus
Show the current configuration.
The logger[jeus] information for the server [server1]
=====
+-----+-----+-----+
|           Name           |           Value           |
+-----+-----+-----+
| Level                    | INFO                      |
| Use Parent Handlers      | false                     |
| Formatter                | jeus.util.logging.SimpleFormatter |
+-----+-----+-----+
=====

[DAS]domain1.adminServer>list-log-handlers -server server1 jeus
List of Loggers
=====
+-----+-----+-----+
| Handler Name           | FileHandler              |
| Handler Type           | FileHandlerType          |
| Handler Level          | FINEST                   |
| Filename                | JeusServer.log           |
| Enable Rotation         | true                      |
| Rotation Directory      | ${SERVER_HOME}/logs      |
| Valid Day               | 1                         |
| Buffer Size             | 1024                     |
+-----+-----+-----+
```

| | | |
|---------------|------|--|
| Append Logs | true | |
| +-----+-----+ | | |
| ===== | | |

コンソール・ツールを使ってjeusロガー情報を確認するとき、コンソール・ハンドラーが確認される場合は2つのケースがあります。

- サーバーを起動時に、実行スクリプトに-verboseオプションを設定する場合
- サーバーが起動中の場合

サーバーを起動時に-verboseオプションを設定した場合は、ランチャー・プロセスでコンソール・ログが出力されます。サーバーが起動中の場合も、ランチャー・プロセスによってログ・メッセージがコンソール画面に出力されることがあります。

8.3.2. ロガーの動的設定

コンソール・ツールやWebAdminを使ってランタイムにロガーやハンドラーを追加、削除、変更することができます。

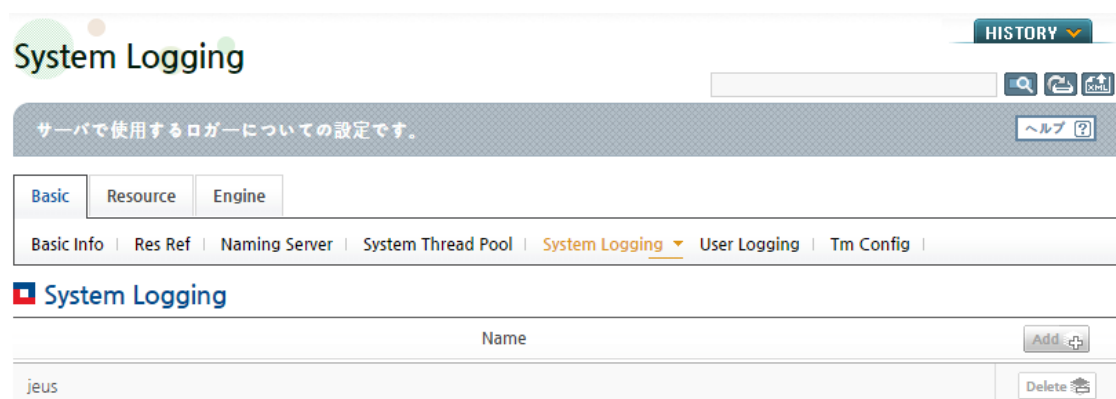
WebAdminの使用

以下では、WebAdminを使ってサーバーにロガーを動的に追加する手順について説明します。

1. WebAdminの左のメニューで**[Servers]**を選択すると、サーバー・リスト画面が表示されます。リストから設定するサーバー(server1)を選択すると、サーバー設定画面に移動します。

サーバー設定画面で**[Basic Info] > [System Logging]**メニューを選択します。

【図 8.4】 WebAdminによるロガーの動的設定(1)



2. サーバーにロガーを追加するには、先に設定変更のためのロックを設定する作業が必要であり、左のメニューの下部にある**[Lock & Edit]**ボタンをクリックしてロックを設定します。

次に、ロガー・リストの上部にある[Add]ボタンをクリックします。

[図 8.5] WebAdminによるロガーの動的設定(2)

System Logging

サーバで使用するロガーについての設定です。

Basic Resource Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

| Name |
|------|
| jeus |

Add Delete

3. ロガーの名前(「Name」)とレベル(「Level」)を設定して[確認]ボタンをクリックします。

[図 8.6] WebAdminによるロガーの動的設定(3)

System Logging

サーバで使用するロガーについての設定です。

Basic Resource Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

動的設定 * 必須項目

確認 再設定

Name * jeus.config
ロガーに対して設定を適用するとき、該当するロガー名を指定します。ロガー名を確認したい場合は、ロガーページを参照してください。

Level FINE
【デフォルト: INFO】 ロガーのレベルを設定します。各レベルの意味については、Java SE logging APIの「Level Class Documentation」を参照してください。

詳細設定 すべてを開く

Use Parent Handlers ☒

Filter Class com.tmax.logging.filter.MyFilter

Formatter Class com.tmax.logging.handler.MyHandler

確認 再設定

Handlers

| Name | Type | Level |
|------|------|-------|
|------|------|-------|

該当する内容が存在しません。

4. 画面上部で設定の一時保存結果メッセージが表示され、ロガー・リストで追加したロガー情報が確認できます。追加した「jeus.config」ロガーをクリックします。

【図 8.7】 WebAdminによるロガーの動的設定(4)

System Logging

サーバで使用するロガーについての設定です。

追加されました。

Basic Resource Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

| Name | |
|-------------|--------|
| jeus | Delete |
| jeus.config | Delete |

5. ファイル・ハンドラーを追加するために、画面下部にある[FILE HANDLER]ボタンをクリックします。

【図 8.8】 WebAdminによるロガーの動的設定(5)

System Logging

サーバで使用するロガーについての設定です。

Basic Resource Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

動的設定 必須項目

確認 再設定

| | |
|-------|--|
| Name | jeus.config ロガーに対して設定を適用するとき、該当するロガー名を指定します。ロガー名を確認したい場合は、ロガーページを参照してください。 |
| Level | FINE [デフォルト: INFO] ロガーのレベルを設定します。各レベルの意味については、Java SE logging APIの「Level Class Documentation」を参照してください。 |

詳細設定

すべてを開く

Use Parent Handlers ☒

Filter Class

Formatter Class

確認 再設定

Handlers

| Name | Type | Level |
|------|------|-------|
|------|------|-------|

該当する内容が存在しません。

File Handler SMTP Handler Socket Handler User Handler

6. 追加するファイル・ハンドラーの名前を設定し、ログ・ファイルのローテーション方式を毎日更新するように1日に設定して[確認]ボタンをクリックします。

[図 8.9] WebAdminによるロガーの動的設定(6)

File Handler

HISTORY

ログメッセージをファイルに出力する場合に使用するハンドラです。

ヘルプ

BasicResourceEngine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

動的設定 必須項目

確認再設定

| | |
|--|--|
| Name * | <input type="text" value="fileHandler"/> EX handler1 ハンドラの名前を設定します。この名前は1つのロガー内で一意である必要があります。設定された名前は管理ツール(WebAdminのツール)などでハンドラを指すときに使用します。 |
| File Name | <input type="text" value="/home/jeus/logs/mylog.log"/> EX /home/jeus/logs/mylog.log ハンドラが使用するファイル名を設定します。設定していない場合は、各ロガーのデフォルトファイル名が使用されます。各デフォルトファイル名については、「JEUS サーバガイド」を参照してください。 |
| Level | <div></div> [デフォルト: FINEST] ハンドラのレベルを設定します。ロガーが送信したメッセージのレベルがハンドラに指定されているレベルに該当する場合のみ出力されます。 |
| Enable Rotation | <input checked="" type="checkbox"/> EX true [デフォルト: true] ハンドラが使用するファイルがログファイルローテーション機能を使用するか否かを設定します。特に設定していない場合はtrueに設定され、ファイルにロギングする際にローテーション機能を使用します。 |
| Rotation Count | <input type="text" value="10"/> EX 10 ハンドラが使用するファイルがログファイルローテーション機能を使用するとき、バックアップするファイル数を設定します。設定せずにファイルサイズでローテーションする場合、99999個まで蓄積されます。日付または時間でローテーションする場合は、ローテーションされたファイルは継続して蓄積されます。 |
| <input checked="" type="radio"/> Valid Day | <input type="text" value="d"/> EX 1 [デフォルト: 1] ハンドラが使用するファイルをValid Dayに設定した期間のみ使用し、継続して更新されます。この設定は日数単位でファイルを更新するときに使います。この場合、ハンドラが使用するファイル名の最後にファイルが使用された日付が自動で付与されます。 |
| <input type="radio"/> Valid Hour | <input type="text" value="h"/> EX 3 ハンドラが使用するファイルをValid Hourに設定した時間のみ使用し、継続して更新されます。この設定は時間単位でファイルを更新するときに使います。この場合、ハンドラが使用するファイル名の最後にファイルが使用された日付と時間が自動で付与されます。 |
| <input type="radio"/> Valid Size | <input type="text" value="kbyte"/> EX 1024 ハンドラが使用するファイルがValid Sizeに設定したサイズより小さい場合のみ使用し、継続して更新されます。この設定はサイズ単位でファイルを更新するときに使います。この場合、ハンドラが使用するファイル名の最後に順次的にインデックスが付与されます。 |

必要な場合、詳細設定を設定します。

詳細設定

すべてを開く

| | |
|--------------|--|
| Encoding | <input type="text"/> |
| Filter Class | <input type="text" value="com.tmax.logging.filter.MyFilter"/> EX com.tmax.logging.filter.MyFilter |
| Append | <input checked="" type="checkbox"/> |
| Rotation Dir | <input type="text" value="/home/jeus/backup_logs"/> EX /home/jeus/backup_logs |
| Buffer Size | <input type="text" value="byte"/> EX byte |

確認再設定

7. 以下は、ログ・ハンドラーを追加した結果画面です。画面上部に設定の一時保存結果メッセージが表示されます。また、ハンドラー・テーブルに追加したハンドラーが表示されることが確認できます。

【図 8.10】 WebAdminによるログーの動的設定(7)

System Logging

HISTORY

サーバで使用するログーについての設定です。

追加されました。

ヘルプ

Basic Resource Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

動的設定 必須項目 確認 再設定

Name * jeus.config
ログーに対して設定を適用するとき、該当するログー名を指定します。ログー名を確認したい場合は、ログーページを参照してください。

Level FINE
[デフォルト: INFO] ログーのレベルを設定します。各レベルの意味については、Java SE logging APIの「Level Class Documentation」を参照してください。

詳細設定 すべてを開く

Use Parent Handlers ☒

Filter Class

Formatter Class

確認 再設定

Handlers

| Name | Type | Level |
|-------------|------|-------|
| fileHandler | file | |

Delete

File Handler SMTP Handler Socket Handler User Handler

8. **[Activate Changes]**ボタンをクリックして変更した設定をサーバーに反映します。
9. 画面上部には、ログーとログーのハンドラーを追加してサーバーに反映した結果がメッセージで表示されます。ログーとハンドラーの追加は、運用中のサーバーに動的に反映できます。

参考

WebAdminを使用したログーの変更と削除方法は、本ガイドでは説明を省略します。同内容については、上述したログーの追加方法を参照してください。

コンソール・ツールの使用

コンソール・ツールで**add-logger**、**modify-logger**、**remove-logger**コマンドを使ってロギングの設定を動的に変更することができます。

list-loggersコマンドを使用すると、現在設定されているロガー情報を確認できます。各コマンドについての詳しい説明は、『*JEUS リファレンスガイド*』の「4.2.3. サーバー管理関連コマンド」を参照してください。

[例 8.6] コンソール・ツールによるロガーの追加/変更/削除

```
[DAS]domain1.adminServer>list-loggers server1
List of Loggers
=====
+-----+-----+-----+-----+-----+
| Logger Name | Level | Use Parent | Filter | Formatter |
|             |       | Handlers   |        |            |
+-----+-----+-----+-----+-----+
| jeus        | INFO  | false      |        | jeus.util.logging.SimpleFor|
|             |       |            |        | macter      |
+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>add-logger -server server1 jeus.ejb -level FINEST
Successfully performed the ADD operation for The logger for the server(server1)..
Check the results using "list-loggers or add-logger"

[DAS]domain1.adminServer>add-logger -server server1 jeus.ejb.clustering -level FINEST
Successfully performed the ADD operation for The logger for the server(server1)..
Check the results using "list-loggers or add-logger"

[DAS]domain1.adminServer>list-loggers server1
List of Loggers
=====
+-----+-----+-----+-----+-----+
| Logger Name | Level | Use Parent | Filter | Formatter |
|             |       | Handlers   |        |            |
+-----+-----+-----+-----+-----+
| jeus        | INFO  | false      |        | jeus.util.logging.SimpleF|
|             |       |            |        | ormatter   |
+-----+-----+-----+-----+-----+
| jeus.ejb    | FINEST | true       |        | jeus.util.logging.SimpleF|
|             |       |            |        | ormatter   |
+-----+-----+-----+-----+-----+
| jeus.ejb.cluste| FINEST | true       |        | jeus.util.logging.SimpleF|
| ring        |       |            |        | ormatter   |
+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>modify-logger -server server1 jeus.ejb.clustering -level FINE
Successfully performed the MODIFY operation for The logger[jeus.ejb.clustering]
```

```

information for the server [server1].
Check the results using "modify-logger"

[DAS]domain1.adminServer>list-loggers server1
List of Loggers
=====
+-----+-----+-----+-----+-----+
| Logger Name | Level | Use Parent | Filter | Formatter |
|             |       | Handlers   |        |            |
+-----+-----+-----+-----+-----+
| jeus        | INFO  | false      |        | jeus.util.logging.SimpleF|
|             |       |            |        | ormatter   |
+-----+-----+-----+-----+-----+
| jeus.ejb    | FINEST| true       |        | jeus.util.logging.SimpleF|
|             |       |            |        | ormatter   |
+-----+-----+-----+-----+-----+
| jeus.ejb.cluste| FINE  | true       |        | jeus.util.logging.SimpleF|
| ring        |       |            |        | ormatter   |
+-----+-----+-----+-----+-----+
=====

[DAS]domain1.adminServer>remove-logger -server server1 jeus.ejb
Successfully performed the REMOVE operation for The logger for the server(server1)..
Check the results using "list-loggers or remove-logger"

[DAS]domain1.adminServer>list-loggers server1
List of Loggers
=====
+-----+-----+-----+-----+-----+
| Logger Name | Level | Use Parent | Filter | Formatter |
|             |       | Handlers   |        |            |
+-----+-----+-----+-----+-----+
| jeus        | INFO  | false      |        | jeus.util.logging.SimpleFo|
|             |       |            |        | rmatter    |
+-----+-----+-----+-----+-----+
| jeus.ejb.cluste| FINE  | true       |        | jeus.util.logging.SimpleFo|
| ring        |       |            |        | rmatter    |
+-----+-----+-----+-----+-----+
=====

```

8.3.3. 標準出力と標準エラーのログ形式の出力

JEUSでは標準出力と標準エラーをJEUSのログ形式で出力する機能を提供します。JEUSでデフォルトで使用するフォーマットを使用して、JEUSロガーと類似した形式で標準出力と標準エラーを出力できます。

以下では、WebAdminまたはコンソール・ツールを使って標準出力と標準エラーをJEUSのログ形式で出力するための設定方法について説明します。

WebAdminの使用

WebAdminの左のメニューで[**Servers**]を選択してサーバー・リスト画面に移動します。設定するサーバー(server1)を選択すると、サーバー設定画面に移動します。サーバー設定画面で[**Basic Info**]メニューを選択すると、下位の**詳細設定**で「**Log Stdout To Raw Format**」項目を設定できます。

[図 8.11] WebAdminで標準出力と標準エラーをJEUSのログ形式で出力する設定

詳細設定 すべてを開く

Server
ドメイン内で使用するJEUSサーバの詳細設定を定義します。

| | | |
|--------------------------|-------------------------------------|--|
| Use MEJB | <input type="checkbox"/> | クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。 [デフォルト: false] J2EE管理仕様で提示するMEJBを使用するか否かを設定します。使用しない場合は、MEJBをデプロイしません。 |
| Class Ftp | <input type="checkbox"/> | クラスタに含まれるサーバの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。 [デフォルト: false] EJB 2.Xクライアントが動的プロキシ方式ではなく、スタブ方式の呼び出しを希望する場合は、クライアントにスタブが存在する必要があります。クラスFTPサービスが有効になっていれば、FTPを使ってEJBスタブファイルをクライアントに転送します。このサービスが無効の場合は、直接EJBスタブファイルをクライアントにコピーする必要があります。基本的に動的プロキシを使用するため、通常、このサービスは無効になっています。 |
| Log Stdout To Raw Format | <input checked="" type="checkbox"/> | [デフォルト: true] 標準出力(stdout)と標準エラー出力(stderr)を行うとき、RAWフォーマットにするか、ログフォーマットにするかを指定します。フォーマッタを使用するときのみ適用され、ログで出力する場合は[STDOUT]プレフィックスが付けられます。デフォルト値はtrueであり、標準出力と標準エラー出力をRAWフォーマットで表示します。 |

参考

「**Log Stdout To Raw Format**」項目のデフォルト値はtrueであり、標準出力と標準エラーをそのまま出力します。標準出力と標準エラーをJEUSのログ形式で出力するには、この値をfalseに設定します。WebAdminでは、項目のチェックを解除すると同じ結果になります。

コンソール・ツールの使用

以下は、コンソール・ツールで標準出力と標準エラーをJEUSのログ形式で出力する設定の例です。

[例 8.7] コンソール・ツールで標準出力と標準エラーをJEUSのログ形式で出力する設定

```
[DAS]domain1.adminServer>modify-server server1
Show the current configuration.
server (server1)
=====
+-----+-----+
| Node           | node1                               |
| JVM Configs    | -Xmx256m -XX:MaxPermSize=128m     |
| Action On Resource Leak | WARNING                           |
| Stdout to Raw Format | true                               |
| MEJB           | false                              |
+-----+-----+
```

```
| Class FTP | false |
| Server Log Home Directory | none |
+-----+-----+
=====

[DAS]domain1.adminServer>modify-server server1 -logStdoutToRawFormat false
Successfully performed the MODIFY operation for server (server1).
Check the results using "list-servers server1 or modify-server server1"

[DAS]domain1.adminServer>list-servers server1
List of Servers
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Ser | Base | Base | Node | JVM | Action | Stdout | MEJB | Class | Server | Type |
| ver | Listen | Listen | | | On | to Raw | | FTP | Log | |
| | Addre | Port | | | Resour | Format | | Home | Direct |
| | ss | | | | ce Leak | | | | ory |
| | | | | | | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ser | | 9836 | nod | -Xmx2 | Warni | false | fal | false | none | ser |
| ver1 | | | el | 56m | ng | | se | | | ver |
| | | | | -XX:Ma | | | | | | |
| | | | | xPermS | | | | | | |
| | | | | ize=12 | | | | | | |
| | | | | 8m | | | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
=====
```

この機能を使用すると、標準出力は以下のようなフォーマットで出力されます。

| [時間] [レベル] [ログGINGされるスレッド情報] [STDOUT/STDERR] メッセージ | |
|--|---|
| 項目 | 説明 |
| [時間] | 「年.月.日 時間:分:秒」の形式で出力します |
| [レベル] | ログ・レベルをマッピングされる数字で出力します。すべてSEVEREレベルであり、マッピングされる数字は0です |
| [ログGINGされるスレッド情報] | ログGINGするプロセス(サーバーまたはランチャー)とスレッド番号で表現され、この2つはハイフン(-)で区分されます。スレッド情報が同一のログ・メッセージは同じスレッドでログGINGしたものです |
| [STDOUT/STDERR] | 標準出力なのか標準エラーなのかを示します – STDOUT : ログGINGするメッセージが標準出力の場合 – STDERR : 例外などの標準エラーの場合 – UNKNOWN : 分からない場合 |
| メッセージ | System.outまたはSystem.errで出力するメッセージです |

8.3.4. ロガーの設定

WebAdminまたはコンソール・ツールを使ってロガーやハンドラーを追加、削除、変更することができます。これらの操作はすべて動的に反映できます。つまり、運用中のサーバーを再起動しなくても、変更した設定内容を適用できます。

参考

動的に反映できるロガーの設定は、レベル(level)と上位ハンドラーの使用の可否(use-parent-handlers)です。ハンドラーの設定では、レベル(level)のみ動的に反映できます。コンソール・ツールのコマンドでは、ファイル・ハンドラーだけ追加、変更することができます。

WebAdminの使用

WebAdminの左のメニューで[**Servers**]を選択して、照会されるサーバー・リストで目的のサーバーをクリックします。[**Basic**] > [**System Logging**]メニューを選択します。ロガーを設定するために[**Add**]ボタンをクリックすると、ロガー設定画面が表示されます。

[図 8.12] WebAdminのロガー設定画面

System Logging HISTORY

サーバで使用するロガーについての設定です。 ヘルプ

Basic Resource Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | **System Logging** | User Logging | Tm Config

動的設定 必須項目 確認 再設定

Name ロガーに対して設定を適用するとき、該当するロガー名を指定します。ロガー名を確認したい場合は、ロガーページを参照してください。

Level [デフォルト: INFO] ロガーのレベルを設定します。各レベルの意味については、Java SE logging APIの「Level Class Document ation」を参照してください。

詳細設定 すべてを閉く

Use Parent Handlers ☒

Filter Class

Formatter Class

確認 再設定

Handlers

| Name | Type | Level |
|----------------|------|-------|
| 該当する内容が存在しません。 | | |

以下は、設定項目についての説明です。

| 項目 | 説明 |
|---------------------|--|
| Name | ロガーの名前を設定します |
| Level | ロガーのレベルを設定します。このレベル以下のメッセージだけがロガーによって出力されます。使用可能な値としては、Logging APIのSEVERE、WARNING、INFO、CONFIG、FINE、FINER、FINEST、ALLレベルがあります(デフォルト値: INFO) |
| Use Parent Handlers | ロガーが自身のハンドラー以外に上位ロガーのハンドラーを使用するかどうかを設定します。 デフォルト値はtrueです。jeusロガーの場合は、JEUSの最上位ロガーであるため、falseに設定します。ハンドラーを設定せずに、この値をfalseに設定した場合は、上位ハンドラーを使用できるように、この値がtrueに変更されます |
| Filter Class | ロガーがログ・メッセージをハンドラーに送信する前に実行するフィルタリングで使用するクラスを指定します。指定したクラスは、lib/applicationディレクトリーのJARファイル内に含まれている必要があります |
| Formatter Class | ロガーがログ・メッセージをハンドラーに送信する前に実行するログ・メッセージのフォーマットिंगで使用するクラスを指定します。指定したクラスは、lib/applicationディレクトリーのJARファイル内に含まれている必要があります。 ログ・フォーマッターを指定するには、ロガーにハンドラーが設定されている必要があります。指定したフォーマッターは設定されたハンドラーによって出力されるログ・メッセージにのみ適用されます |
| Handlers | ロガーが使用するハンドラーを指定します。この項目を設定していない場合、jeusロガーはファイル・ハンドラーを基本的に使用し、その他の下位ロガーはjeusロガーのハンドラーを使用します |

参考

JEUS 4.xで使用されたFATAL、NOTICE、INFORMATION、DEBUGログ・レベルは、JEUS 7からは使用できません。

Handlerにはファイル・ハンドラー、SMTPハンドラー、ソケット・ハンドラー、ユーザー・ハンドラーがあり、各ハンドラーには以下のような下位項目があります。

• ファイル・ハンドラー

ファイル・ハンドラーは、ログ・メッセージをファイルに出力するハンドラーです。

以下は、設定項目についての説明です。

| 項目 | 説明 |
|-----------------|---|
| Name | ハンドラーの名前を指定します。この名前は1つのロガー内で一意である必要があります。指定しない場合、クラス名とハッシュ・コードを組み合わせた名前が使用されます |
| File Name | <p>ハンドラーがログ・メッセージを出力するファイルの名前を指定します。絶対パスを指定する場合はそのパスにファイルが生成され、相対パスを指定する場合は各ロガーのデフォルト・パスを基準とする相対パスとして認識します。</p> <p>設定しない場合、ロガー別に指定されたパスにファイルを生成してログ・メッセージを出力します</p> |
| Level | <p>ハンドラーが出力するメッセージのレベルを指定します。ログ・メッセージを出力するときに、ロガーを通過したログ・メッセージはロガーが使用するそれぞれのハンドラーに送信されますが、そのとき、指定したレベルに適合するログ・メッセージだけがハンドラーによって出力されます。</p> <p>デフォルト値を設定すると、ロガーを通過するすべてのログ・メッセージがハンドラーによって出力されます(デフォルト値: FINEST)</p> |
| Enable Rotation | ハンドラーがログ・ファイルのローテーション機能を使用するかどうかを設定します。デフォルト値を設定すると、ローテーション機能を使用します(デフォルト値: true) |
| Rotation Count | ローテーション回数を設定します。ハンドラーがログ・ファイルのローテーション機能を使用する場合にのみ意味があります(デフォルト値: 10) |
| Valid Day | <p>ハンドラーが出力するファイルを日付別に作成する場合に使用します。</p> <p>ファイル名の形式は、ファイルの末尾に「_YYYYMMDD」が付きます</p> |
| Valid Hour | <p>ハンドラーが出力するファイルを時間別に作成する場合に使用します。</p> <p>24の約数(例:3、6)、または24で割った余りが約数(例:27、30)の値を指定します。ファイル名の形式は、ファイルの文末に「_YYYYMMDD_HH」が付きます</p> |
| Valid Size | ハンドラーが出力するファイルをサイズ別に作成する場合に設定します。ハンドラーがログ・ファイルのローテーション機能を使用する場合にのみ意味があります |
| Encoding | ハンドラーが出力する文字列のエンコーディングを指定します。デフォルトではシステム・エンコーディングが設定されます |
| Filter Class | <p>ハンドラーがログ・メッセージをフィルタリングして出力する場合に使われるクラスです。</p> <p>ロガーのFilter Class設定と同様に、lib/applicationにこのクラスを含むJARファイルが存在する必要があります</p> |

| 項目 | 説明 |
|--------------|--|
| Append | <p>サーバーを起動させた後ログをファイルで出力するときに、すでに同じ名前のファイルが存在する場合、ファイルを上書きするか、ファイルの末尾に追加するかを決定します(デフォルト値: true)</p> <ul style="list-style-type: none"> – true : ファイルの末尾に継続して追加します – false : ローテーション機能を使用するかどうかによって動作が異なります。ローテーションを使用している場合は、サーバーを起動するたびに以前のログ・ファイルがバックアップされ、新しいログ・ファイルが生成されます。しかし、ローテーション機能を使用していない場合は、サーバーを起動するたびに以前のログ・ファイルが削除されるので、注意して使用する必要があります |
| Rotation Dir | ハンドラーがログ・ファイルのローテーション機能を使用する場合にのみ意味があります |
| Buffer Size | <p>ファイルに出力する時に使用するバッファのサイズを指定します。バッファが大きいほどロギングの性能は良くなりますが、予想外の状況でJEUSが異常終了した場合は、そのバッファ・サイズ分のログが失われる可能性があります。(デフォルト値: 1024、単位: KB)</p> <p>バッファにログ・メッセージを溜めておいて、バッファのサイズが指定したサイズより大きくなると、ログ・メッセージをファイルに出力します</p> |

● SMTPハンドラー

SMTPハンドラーは、ログ・メッセージをメールに送信するハンドラーです。1つのログ・メッセージが1つのメールに送信されます。

以下は、設定項目についての説明です。

| 項目 | 説明 |
|------------------|---|
| Name | ハンドラーの名前を指定します。この名前は1つのロガー内で一意である必要があります。指定しない場合、クラス名とハッシュ・コードを組み合わせた名前が使用されます |
| Level | <p>ハンドラーが出力するメッセージのレベルを指定します。ログ・メッセージを出力するときに、ロガーを通過するログ・メッセージはロガーが使用するそれぞれのハンドラーに送信されますが、このとき、指定したレベルに適合するログ・メッセージだけがハンドラーによって出力されます。</p> <p>デフォルト値を設定すると、ロガーを通過するすべてのログ・メッセージがハンドラーによって出力されます(デフォルト値: FINEST)</p> |
| Smtphost Address | メールを送信するホスト・アドレスを指定します |

| 項目 | 説明 |
|-----------------------|--|
| From Address | メール送信者のアドレスを指定します |
| Sender ID | メール送信者のIDを指定します |
| Sender Password | メール送信者のパスワードを指定します。暗号化して保存する場合は、「{暗号化アルゴリズム}暗号化されたパスワード」の形式で入力します |
| To Address | メール受信者のアドレスを指定します |
| Property | 使用するメール・サーバーごとに特定のSMTPプロパティを要求することがあります。schema(WebAdmin)に明示された基本的なSMTPプロパティ以外に、追加で必要なプロパティがある場合は、「key,value」形式で入力することができます。schemaに明示された基本的なSMTPプロパティと重複する項目は、schema(WebAdmin)の値に従います |
| Send For All Messages | すべてのメッセージをSMTPハンドラーで送信するかどうかを決定します。falseの場合は、JEUSシステムがメールで送信するように設定したメッセージのみをこのハンドラーを使って送信します。現在、この設定はユーザー・ロガーでのみ有効です(デフォルト値: false) |
| Encoding | ハンドラーが出力する文字列のエンコーディングを指定します。デフォルトではシステム・エンコーディングが設定されます |
| Filter Class | ハンドラーがログ・メッセージをフィルタリングして出力する場合に使われるクラスです。 ロガーのFilter Class設定と同様に、lib/applicationにこのクラスを含むJARファイルが存在する必要があります |
| Cc Address | 同報(Cc)メール受信者のアドレスを指定します |
| Bcc Address | 隠し同報(Bcc)メール受信者のアドレスを指定します |

● ソケット・ハンドラー

ソケット・ハンドラーは、ログ・メッセージをソケットに送信するハンドラーです。

以下は、設定項目についての説明です。

| 項目 | 説明 |
|-------|---|
| Name | ハンドラーの名前を指定します。この名前は1つのロガー内で一意である必要があります。指定しない場合、クラス名とハッシュ・コードを組み合わせた名前が使用されます |
| Level | ハンドラーが出力するメッセージのレベルを指定します。ログ・メッセージを出力するときに、ロガーを通過するログ・メッセージはロガーが使用するそれぞれのハンドラーに送信されますが、このとき、指定したレベルに適合するログ・メッセージだけがハンドラーによって出力されます。 |

| 項目 | 説明 |
|--------------|---|
| | デフォルト値を設定すると、ログを通過するすべてのログ・メッセージがハンドラーによって出力されます(デフォルト値: FINEST) |
| Address | ハンドラーが接続するマシンのIPアドレスを指定します |
| Port | ハンドラーが接続するマシンのポート番号を指定します |
| Encoding | ハンドラーが出力する文字列のエンコーディングを指定します。デフォルトではシステム・エンコーディングが設定されます |
| Filter Class | <p>ハンドラーがログ・メッセージをフィルタリングして出力する場合に使われるクラスです。</p> <p>ログのFilter Class設定と同様に、lib/applicationにこのクラスを含むJARファイルが存在する必要があります</p> |

● ユーザー・ハンドラー

ユーザー・ハンドラーは、ユーザーが作成したハンドラー・クラスを指定する項目です。

以下は、設定項目についての説明です。

| 項目 | 説明 |
|------------------|--|
| Name | ハンドラーの名前を指定します。この名前は1つのログ内で一意である必要があります。指定しない場合、クラス名とハッシュ・コードを組み合わせた名前が使用されます |
| File Name | <p>ハンドラーがログ・メッセージを出力するファイルの名前を指定します。絶対パスを指定する場合はそのパスにファイルが生成され、相対パスを指定する場合は各ログのデフォルト・パスを基準とする相対パスとして認識します。</p> <p>設定しない場合、ログ別に指定されたパスにファイルを生成してログ・メッセージを出力します</p> |
| Handler Class | ユーザーが作成したハンドラーのクラスを指定します。このクラスは、lib/applicationディレクトリーのJARファイルに含まれている必要があります。また、このクラスはlogging APIのjava.util.logging.Handlerを継承し、jeus.util.logging.JeusHandlerを実装する必要があります |
| Handler Property | jeus.util.logging.JeusHandlerのsetProperty()に使用されるMapオブジェクトに含まれるプロパティを指定します |
| Formatter Class | ハンドラーが使用するフォーマッター・クラスを指定します。このクラスもlib/applicationディレクトリーのJARファイルに含まれている必要があります。また、jeus.util.logging.JeusFormatterインターフェースを実装する必要があります(デフォルト値: jeus.util.logging.SimpleFormatter) |

| 項目 | 説明 |
|--------------------|--|
| Formatter Property | jeus.util.logging.JeusFormatterのsetProperty()に使用されるMapオブジェクトに含まれるプロパティを指定します |
| Encoding | ハンドラーが出力する文字列のエンコーディングを指定します。デフォルトではシステム・エンコーディングが設定されます |
| Filter Class | <p>ハンドラーがログ・メッセージをフィルタリングして出力する場合に使われるクラスです。</p> <p>ローガーのFilter Class設定と同様に、lib/applicationIにこのクラスを含むJARファイルが存在する必要があります</p> |

8.3.5. ログ・ファイルのローテーションの設定

JEUSの起動時に、あるいはJEUSを運用中に設定した時間になったり設定したファイル・サイズを超えたりする場合、自動で以前ロギングしたファイルの名前を変更し、新規出力するログをそのログ・ファイルに継続してロギングする、**ログ・ファイルのローテーション機能**を提供します。

「Valid Day」、「Valid Hour」、「Valid Size」の3つの項目の設定に従ってローテーションを行います。この3つの項目のいずれも設定していない場合は、毎日00時00分にロギングしていたファイル名を変更してバックアップした後、既存のファイルに新しくロギングします。

WebAdminを使って**File Handler**のログ・ファイルのローテーションを設定するには、以下に表示した項目を設定します。

[図 8.13] WebAdminによるログ・ファイルのローテーションの設定

File Handler

HISTORY

ヘルプ

ログメッセージをファイルに出力する場合に使用するハンドラです。

Basic

Resource

Engine

Basic Info | Res Ref | Naming Server | System Thread Pool | System Logging | User Logging | Tm Config

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

| | |
|--|---|
| Name * | fileHandler <div>EX handler1</div> <div>ハンドラの名前を設定します。この名前は1つのログ一内で一意である必要があります。設定された名前は管理ツール(WebAdminのツール)などでハンドラを指すときに使用します。</div> |
| File Name | /jeusServer.log <div>EX /home/jeus/logs/mylog.log</div> <div>ハンドラが使用するファイル名を設定します。設定していない場合は、各ログ一のデフォルトファイル名が使用されます。各デフォルトファイル名については、「JEUS サーバガイド」を参照してください。</div> |
| Level | FINEST <div>EX</div> <div>[デフォルト: FINEST] ハンドラがレベルを設定します。ログ一が送信したメッセージのレベルがハンドラに指定されているレベルに該当する場合のみ出力されます。</div> |
| Enable Rotation | <input checked="" type="checkbox"/> <div>EX true</div> <div>[デフォルト: true] ハンドラが使用するファイルがログファイルローテーション機能を使用するか否かを設定します。特に設定していない場合はtrueに設定され、ファイルにロギングする際にローテーション機能を使用します。</div> |
| Rotation Count | <div>EX 10</div> <div>ハンドラが使用するファイルがログファイルローテーション機能を使用するとき、バックアップするファイル数を設定します。設定せずにファイルサイズでローテーションする場合、99999個まで蓄積されます。日付または時間でローテーションする場合は、ローテーションされたファイルは継続して蓄積されます。</div> |
| <input checked="" type="radio"/> Valid Day | 1 <div>EX 1</div> <div>[デフォルト: 1] ハンドラが使用するファイルをValid Dayに設定した期間のみ使用し、継続して更新されます。この設定は日数単位でファイルを更新するときに使います。この場合、ハンドラが使用するファイル名の最後にファイルが使用された日付が自動で付与されます。</div> |
| <input type="radio"/> Valid Hour | <div>EX 3</div> <div>ハンドラが使用するファイルをValid Hourに設定した時間のみ使用し、継続して更新されます。この設定は時間単位でファイルを更新するときに使います。この場合、ハンドラが使用するファイル名の最後にファイルが使用された日付と時間が自動で付与されます。</div> |
| <input type="radio"/> Valid Size | <div>EX 1024</div> <div>ハンドラが使用するファイルがValid Sizeに設定したサイズより小さい場合のみ使用し、継続して更新されます。この設定はサイズ単位でファイルを更新するときに使います。この場合、ハンドラが使用するファイル名の最後に順次的にインデックスが付与されます。</div> |

詳細設定

すべてを閉じる

Encoding

ハンドラがメッセージを記録するときに使用するエンコーディングを設定します。

Filter Class

EX com.tmax.logging.filter.MyFilter

ハンドラに指定するフィルタクラスの完全修飾クラス名を設定します。フィルタクラスに実装した内容をベースにしてログメッセージがフィルタリングされ、出力されます。

Append

☒

[デフォルト: true] ハンドラが使用するファイルが既に存在する場合、ファイルをローテーションするか否かを設定します。イネーブルローテーションがtrueに設定された場合はブートタイムにローテーションし、イネーブルローテーションがfalseに設定された場合は既存ファイルを上書きします。

Rotation Dir

EX /home/jeus/backup_logs

ハンドラが使用するファイルがログファイルローテーション機能を使用するとき、ローテーションされたファイルが保存される場所を設定します。特に設定していない場合は、現在ロギングしているファイルのディレクトリに保存されます。

Buffer Size

byte

[デフォルト: 1024] ハンドラがファイルに出力するときに使用するバッファのサイズを設定します。

8.3.6. プロパティの設定

必要なプロパティの設定は以下のとおりです。

- システム・プロパティ

ログ・レベルをシステム・プロパティで設定できます。

- スタンドアロン・クライアントでロガーを設定する場合

- ロギング・プロパティ・ファイル

Javaロギング・プロパティ・ファイルにロギング設定が可能です。

- Javaロギング・プロパティ(logging.properties)ファイル

-Djava.util.logging.config.fileでプロパティ・ファイルのパスを指定できます。

(デフォルト・パス: JEUS_HOME/bin/logging.properties)

参考

JEUS 6まで使用できたjeuslogging.propertiesファイルは、JEUS 7からは使用できません。

レベル設定の優先順位

JEUSツールを利用した動的設定を除く他の設定で、レベル設定の優先順位は以下のようになります。

1. システム・プロパティ
2. Javaロギング・プロパティ(logging.properties)ファイル
3. domain.xmlのロギング設定

参考

ハンドラーに設定したログ・レベルは上述した優先順位をオーバーライドするわけではありません。しかし、最終的にはハンドラーによってログが出力されるため、優先順位が最も高いといえます。ハンドラーの基本ログ・レベルはFINESTです。

付録 A. JEUSで使用するポート

本付録では、JEUSで使用するポートを紹介します。ファイアウォールを設定する時に参照してください。

A.1. サーバー・ポート

- BASEPORT

| | |
|-----------|--|
| 説明 | JEUSサーバーがJNDIサービスや運用のために必要とする基本サービス・ポートです。baseリスナーの設定で設定できます |
| Base Port | 9736 |

- COS Naming Server Port

| | |
|-----------|----------------------------|
| 説明 | COSネーミング・サービスのために使用するポートです |
| Base Port | BASEPORT + 4 (e.g. 9740) |

- ORB Port

| | |
|-----------|--------------|
| 説明 | IIOPポートです |
| Base Port | BASEPORT + 1 |

- ORB SSL Port

| | |
|-----------|---------------|
| 説明 | IIOP SSLポートです |
| Base Port | BASEPORT + 2 |

- ORB SSL Mutual Authorization Port

| | |
|-----------|---------------|
| 説明 | IIOP相互認証ポートです |
| Base Port | BASEPORT + 3 |

- EJB RMI Port

| | |
|-----------|-----------------------|
| 説明 | EJBにアクセスするためのRMIポートです |
| Base Port | BASEPORT + 7 |

付録 B. JDBCデータソースの構成例

本付録では、主要データベース・ベンダーのデータソースの設定例を提供します。

B.1. 概要

以下は、本付録で設定例を提供するJDBCデータソースです。

- Oracle Thin
- Oracle OCI
- DB2 Type4(JCC)
- DB2 Type2(JCC)
- Sybase jConnect 5.x, 6.x
- MSSQL 2005 Type4
- Informix Type4
- Tibero Type4
- MySQL 5.x Type4

本付録を参照して、WebAdminを使ってデータソースを設定してください。WebAdminを使ってデータソースを設定する方法については「[6.4. データソース設定](#)」を参照してください。

B.2. Oracle Thin(Type4)の構成例

B.2.1. Oracle Thinコネクション・プール・データソース

以下は、WebAdminを使ってOracleシン・コネクション・プール・データソースを構成する例です。

【図 B.1】 Oracle Thinコネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|---|
| Data Source Id * | <div>ora_thin_cpds</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>ora_thin_cpds</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>oracle</div> <div>入力</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>oracle.jdbc.pool,OracleConnectionPoolDataSource</div> <div>EX oracle.jdbc.pool,OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>ConnectionPoolDataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。</div> |
| Server Name | <div>192.168.1.1</div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div>1521</div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>ord</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>scott</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>*****</div> <div>入力</div> <div>EX (DES)FQrLbQ/D8O1IDVS71L28rw==</div> <div>DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

【図 B.2】 Oracle Thinコネクション・プール・データソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java.lang.String=thin</div> <div>EX name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

B.2.2. Oracle Thin XAデータソース

以下は、WebAdminを使ってOracle Thin XAデータソースを構成する例です。

[図 B.3] Oracle Thin XAデータソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|---|
| Data Source Id * | <div>ora_thin_xads</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>ora_thin_xads</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>oracle</div> <div>入力</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>oracle.jdbc.xa.client,OracleXADataSource</div> <div>EX oracle.jdbc.pool,OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>XADataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプールサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプールサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプールサービスと共にXA運動がサポートされます。</div> |
| Server Name | <div>192.168.1.1</div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div>1521</div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>ord</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>scott</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>.....</div> <div>入力</div> <div>EX {DES}FQrLbQ/D8O1IDVS71L28rw==</div> <div>DBユーザのパスワードを指定します。暗号化して保存するときは、"{algorithm}ciphertext"の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

【図 B.4】 Oracle Thin XAデータソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java.lang.String=thin</div> <div>EX name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

B.3. Oracle OCI (Type2)の構成例

B.3.1. Oracle OCIコネクション・プール・データソース

Oracle OCIドライバを使用するためには、JEUSを実行するときには、-Djava.library.pathにOracle OCIドライバのネイティブ・ライブラリー・パスを設定する必要があります。

[図 B.5] Oracle OCIコネクション・プール・データソースの構成例

The screenshot shows the 'Database' configuration window in JEUS. The 'Connection Pool' tab is selected. The 'Data Source Id' is set to 'ora_oci_cpds'. The 'Export Name' is also 'ora_oci_cpds'. The 'Vendor' is 'oracle'. The 'Data Source Class Name' is 'oracle.jdbc.pool.OracleConnectionPoolDataSource'. The 'Data Source Type' is 'ConnectionPoolDataSource'. The 'Server Name' is empty. The 'Port Number' is empty. The 'Database Name' is empty. The 'User' is 'scott'. The 'Password' is masked with dots. The 'Support Xa Emulation' checkbox is unchecked. The 'Confirm' button is highlighted.

| Field | Value | Description |
|--------------------------|---|--|
| Data Source Id * | ora_oci_cpds | データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。 |
| Export Name | ora_oci_cpds | データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。 |
| Vendor | oracle | JDBCドライバベンダの名前を指定します。 |
| Data Source Class Name * | oracle.jdbc.pool.OracleConnectionPoolDataSource | JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。 |
| Data Source Type * | ConnectionPoolDataSource | データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA連動がサポートされます。 |
| Server Name | | DBが実行されるホスト名、またはIPを設定します。 |
| Port Number | | DBリスナーのポート番号を設定します。 |
| Database Name | | DBの名前を指定します。Oracleの場合、DBのSIDを設定します。 |
| User | scott | DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。 |
| Password | | DBユーザのパスワードを指定します。暗号化して保存するときは、{algorithm}ciphertextLの形式で記述します。 |
| Support Xa Emulation | <input type="checkbox"/> | [デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。 |

[図 B.6] Oracle OCIコネクション・プール・データソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java.lang.String=oci TNSEntryName:java.lang.String=ORCL</div> <div>EX name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

B.4. DB2の構成例

B.4.1. DB2 Type4(JCC)コネクション・プール・データソース

以下は、WebAdminを使ってDB2 Type4コネクション・プール・データソースを構成する例です。

【図 B.7】 DB2 Type4コネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ


BasicConnection Pool

動的設定 必須項目





確認再設定

| | | |
|------------------------|---|--|
| Data Source Id | db2_type4_cpds | データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。 |
| Export Name | db2_type4_cpds | データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。 |
| Vendor | db2 | JDBCドライバベンダの名前を指定します。 |
| Data Source Class Name | com.ibm.db2.jcc.DB2ConnectionPoolDataSource | JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。 |
| Data Source Type | ConnectionPoolDataSource | データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。 |
| Server Name | 192.168.1.1 | DBが実行されるホスト名、またはIPを設定します。 |
| Port Number | 50000 | DBリスナーのポート番号を設定します。 |
| Database Name | TEST1 | DBの名前を指定します。Oracleの場合、DBのSIDを設定します。 |
| User | db2inst1 | DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。 |
| Password | | DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。 |
| Support Xa Emulation | | [デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。 |

[図 B.8] DB2 Type4コネクション・プール・データソースの構成例


詳細設定

すべてを開く

| | |
|---|--|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> ▼ |
| Auto Commit  | <input type="text"/> ▼ |
| Stmt Query Timeout  | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java.lang.Integer=4</div> <div> name:type=value</div> |
| Action On Connection Leak  | <input type="text"/> ▼ |

確認

再設定

B.4.2. DB2 Type4(JCC) XAデータソース

以下は、WebAdminを使ってDB2 Type4 XAデータソースを構成する例です。

[図 B.9] DB2 Type4 XAデータソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

動的設定 必須項目

確認 再設定

| | |
|--------------------------|--|
| Data Source Id * | db2_type4_xads データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。 |
| Export Name | db2_type4_xads データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。 |
| Vendor | db2 入力 JDBCドライバベンダの名前を指定します。 |
| Data Source Class Name * | com.ibm.db2.jcc.DB2XADataSource oracle.jdbc.pool.OracleConnectionPoolDataSource JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。 |
| Data Source Type * | XADataSource データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプールサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプールサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプールサービスと共にXA運動がサポートされます。 |
| Server Name | 192.168.1.1 DBが実行されるホスト名、またはIPを設定します。 |
| Port Number | 50000 DBリスナーのポート番号を設定します。 |
| Database Name | TEST1 DBの名前を指定します。Oracleの場合、DBのSIDを設定します。 |
| User | db2inst1 DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。 |
| Password | 入力 EX (DES)FQrLbQ/D8O1IDVS71L28rw== DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。 |
| Support Xa Emulation | <input type="checkbox"/> [デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。 |

[図 B.10] DB2 Type4 XAデータソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> ▼ |
| Auto Commit | <input type="text"/> ▼ |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java,lang,Integer=4</div> <div> name:type=value</div> |
| Action On Connection Leak | <input type="text"/> ▼ |

確認

再設定

B.4.3. DB2 Type2(JCC) XAデータソース

DB2 Type 2ドライバを使用するためには、DB2クライアントをインストールしてDB Aliasを設定した後、JEUSを実行するときに、-Djava.library.pathまたはシステムのライブラリー・パスにDB2クライアントのネイティブ・ライブラリー・パスを設定します。下記例で、データベース名がDB Aliasになり、DB2サーバー・アドレスとポート番号の入力は不要です。

[図 B.11] DB2 Type2 XAデータソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|--|
| Data Source Id * | <div>db2_type2_xads</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>db2_type2_xads</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>db2</div> <div>入力</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>COM.ibm.db2.jdbc.DB2XADataSource</div> <div>oracle.jdbc.pool.OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>XADataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。</div> |
| Server Name | <div></div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div></div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>TEST1</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>db2inst1</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>.....</div> <div>入力</div> <div>(DES)FQrLbQ/D8O1IDVS71L28rw==</div> <div>DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

[図 B.12] DB2 Type2 XAデータソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|--|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <input type="text"/> <div>EX name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

B.5. Sybaseの構成例

B.5.1. Sybase jConnect 5.xコネクション・プール・データソース

以下は、WebAdminを使ってSybase jConnect 5.xコネクション・プール・データソースを構成する例です。

[図 B.13] Sybase jConnect 5.xコネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|---|
| Data Source Id * | <div>syb_jconn5_cpds</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>syb_jconn5_cpds</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>sybase</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource</div> <div>EX oracle.jdbc.pool.OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>ConnectionPoolDataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。</div> |
| Server Name | <div>192.168.1.1</div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div>5000</div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>testdb</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>sa</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>.....</div> <div>入力</div> <div>EX {DES}FQrLbQ/D8O1IDVS71L28rw==</div> <div>DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

[図 B.14] Sybase jConnect 5.xコネクション・プール・データソースの構成例

[詳細設定](#)

[すべてを開く](#)

| | |
|---------------------------|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> ▼ |
| Auto Commit | <input type="text"/> ▼ |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <input type="text" value="networkProtocol:java.lang.String=Tds"/> name:type=value |
| Action On Connection Leak | <input type="text"/> ▼ |

確認

再設定

B.5.2. Sybase jConnect 6.x XAデータソース

以下は、WebAdminを使ってSybase jConnect 6.x XAデータソースを構成する例です。

[図 B.15] Sybase jConnect 6.x XAデータソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。


ヘルプ

動的設定 必須項目





確認 再設定

| | |
|--------------------------|--|
| Data Source Id * | <input type="text" value="syb_jconn6_xads"/> データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。 |
| Export Name | <input type="text" value="syb_jconn6_xads"/> データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。 |
| Vendor | <input type="text" value="sybase"/> 入力 JDBCドライバベンダの名前を指定します。 |
| Data Source Class Name * | <input type="text" value="com.sybase.jdbc3.jdbc.SybXADataSource"/> JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。 EX oracle.jdbc.pool,OracleConnectionPoolDataSource |
| Data Source Type * | <input type="text" value="XADataSource"/> データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプールサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプールサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプールサービスと共にXA運動がサポートされます。 |
| Server Name | <input type="text" value="192.168.1.1"/> DBが実行されるホスト名、またはIPを設定します。 |
| Port Number | <input type="text" value="5000"/> DBリスナーのポート番号を設定します。 |
| Database Name | <input type="text" value="testdb"/> DBの名前を指定します。Oracleの場合、DBのSIDを設定します。 |
| User | <input type="text" value="sa"/> DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。 |
| Password | <input type="password" value="....."/> 入力 EX {DES}FQrLbQ/D8O1IDVS71L28rw== DBユーザのパスワードを設定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。 |
| Support Xa Emulation | <input type="checkbox"/> [デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。 |

[図 B.16] Sybase jConnect 6.x XAデータソースの構成例


詳細設定

すべてを開く

| | |
|---|--|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit  | <input type="text"/> |
| Stmt Query Timeout  | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <input type="text" value="networkProtocol:java.lang.String=Tds"/>  name:type=value |
| Action On Connection Leak  | <input type="text"/> |

確認

再設定

B.6. MSSQLの構成例

B.6.1. MSSQL 2005コネクション・プール・データソース

以下は、WebAdminを使ってMSSQL 2005コネクション・プール・データソースを構成する例です。

[図 B.17] MSSQL 2005コネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|---|
| Data Source Id * | <div>mssql_2005_cpds</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>mssql_2005_cpds</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>mssql</div> <div>入力</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>com.microsoft.sqlserver.jdbc.SQLServerConnectionPoc</div> <div>oracle.jdbc.pool,OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>ConnectionPoolDataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。</div> |
| Server Name | <div>192.168.1.1</div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div>1411</div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>jeusdb1</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>jeusdb1</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>.....</div> <div>入力</div> <div>(DES)FQrLbQ/D8O1IDVS71L28rw==</div> <div>DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

[図 B.18] MSSQL 2005コネクション・プール・データソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|--|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div><input type="text"/></div> <div>name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

参考

ODBCの設定は、JDBCの設定より先に設定される必要があります。

B.7. Informixの構成例

B.7.1. Informixコネクション・プール・データソース

以下は、WebAdminを使ってInformixコネクション・プール・データソースを構成する例です。

[図 B.19] Informixコネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定


必須項目

確認





再設定

| | |
|--------------------------|---|
| Data Source Id * | <div>informix_cpds</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>informix_cpds</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>informix</div> <div>入力</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>com.informix.jdbcx.lfxConnectionPoolDataSource</div> <div>EX oracle.jdbc.pool,OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>ConnectionPoolDataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。</div> |
| Server Name | <div>ids93fc</div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div>2002</div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>informix</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>informix</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>*****</div> <div>入力</div> <div>EX (DES)FQrLbQ/D8O1IDVS71L28rw==</div> <div>DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

[図 B.20] Informixコネクション・プール・データソースの構成例


[詳細設定](#)

[すべてを開く](#)

| | |
|---|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit  | <input type="text"/> |
| Stmt Query Timeout  | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <input type="text" value="IfxIFXHOST:java.lang.String=localhost"/>  name:type=value |
| Action On Connection Leak  | <input type="text"/> |

確認

再設定

B.8. Tiberoの構成例

B.8.1. Tiberoコネクション・プール・データソース

以下は、WebAdminを使ってTiberoコネクション・プール・データソースを構成する例です。

[図 B.21] Tiberoコネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

動的設定 必須項目

確認 再設定

| | |
|--------------------------|--|
| Data Source Id * | tibero_cpds データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。 |
| Export Name | tibero_cpds データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。 |
| Vendor | tibero JDBCドライバベンダの名前を指定します。 |
| Data Source Class Name * | com.tmax.tibero.jdbc.ext.TbConnectionPoolDataSource oracle.jdbc.pool.OracleConnectionPoolDataSource JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。 |
| Data Source Type * | ConnectionPoolDataSource データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA運動がサポートされます。 |
| Server Name | 192.168.1.1 DBが実行されるホスト名、またはIPを設定します。 |
| Port Number | 8629 DBリスナーのポート番号を設定します。 |
| Database Name | tibero DBの名前を指定します。Oracleの場合、DBのSIDを設定します。 |
| User | jeusdb1 DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。 |
| Password | (DES)FQrLbQ/D8O1IDVS71L28rw== DBユーザのパスワードを指定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。 |
| Support Xa Emulation | <input type="checkbox"/> [デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。 |

[図 B.22] Tiberoコネクション・プール・データソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|---|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <div>driverType:java,lang,String=thinEX name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

B.9. MySQL 5.xの構成例

B.9.1. MySQL Connector/Jコネクション・プール・データソース

以下は、WebAdminを使ってMySQL Connector/Jコネクション・プール・データソースを構成する例です。

[図 B.23] MySQL Connector/Jコネクション・プール・データソースの構成例

Database

HISTORY

JDBCドライバのデータソースインスタンスを作成し、コネクションプールを構成するための特性が含まれています。

ヘルプ

Basic

Connection Pool

動的設定

必須項目

確認

再設定

| | |
|--------------------------|---|
| Data Source Id * | <div>mysql_cpds</div> <div>データソースのIDを設定します。1つのドメインにおいてデータソースIDは、データソースの一意の識別子として動作するように設定します。</div> |
| Export Name | <div>mysql_cpds</div> <div>データソースのJNDI名を設定します。異なる2つのデータソースが異なるサーバのJNDIにバインドされることが保証できるなら、該当するデータソースは同じJNDI名が指定可能です。これは、任意のサーバにおいて同じJNDI名を持つ異なるデータソースを許可しないことを意味します。設定していない場合は、データソースIDをJNDI名として使用します。</div> |
| Vendor | <div>mysql</div> <div>入力</div> <div>JDBCドライバベンダの名前を指定します。</div> |
| Data Source Class Name * | <div>com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolC</div> <div>oracle.jdbc.pool.OracleConnectionPoolDataSource</div> <div>JDBCドライバデータソースクラスの名前を指定します。パッケージ名を含む完全修飾名で記述します。</div> |
| Data Source Type * | <div>ConnectionPoolDataSource</div> <div>データソースのタイプを指定します。DATA_SOURCEに設定すると、コネクションプーリングサービスは提供されません。CONNECTION_POOL_DATA_SOURCEに設定するとコネクションプーリングサービスが提供されます。XA_DATA_SOURCEに設定した場合は、コネクションプーリングサービスと共にXA連動がサポートされます。</div> |
| Server Name | <div>192.168.1.1</div> <div>DBが実行されるホスト名、またはIPを設定します。</div> |
| Port Number | <div>3306</div> <div>DBリスナーのポート番号を設定します。</div> |
| Database Name | <div>test</div> <div>DBの名前を指定します。Oracleの場合、DBのSIDを設定します。</div> |
| User | <div>tester</div> <div>DBユーザのIDを設定します。トランザクション処理などを行うためには十分な権限を有する必要があります。</div> |
| Password | <div>*****</div> <div>入力</div> <div>{DES}FQrLbQ/D8O1lDV571L28rw==</div> <div>DBユーザのパスワードを設定します。暗号化して保存するときは、「{algorithm}ciphertext」の形式で記述します。</div> |
| Support Xa Emulation | <div><input type="checkbox"/></div> <div>[デフォルト: false] コネクションプールデータソースタイプのデータソースにのみ有効な設定です。この設定を適用した場合、コネクションプールデータソースのコネクションがグローバルトランザクション(XA)に参加するようにエミュレーションします。JEUS6までのLocalXADataSourceに代替されるオプションとして、ConnectionPoolDataSourceタイプのコネクションプールに使用します。1つのトランザクションには1つのコネクションプールデータソースのみ参加できる点に注意してください。</div> |

[図 B.24] MySQL Connector/Jコネクション・プール・データソースの構成例

詳細設定

すべてを開く

| | |
|---------------------------|--|
| Description | <input type="text"/> |
| Login Timeout | <input type="text"/> s |
| Isolation Level | <input type="text"/> |
| Auto Commit | <input type="text"/> |
| Stmt Query Timeout | <input type="text"/> ms |
| Pool Destroy Timeout | <input type="text"/> ms |
| Property | <input type="text"/> <div>EX name:type=value</div> |
| Action On Connection Leak | <input type="text"/> |

確認

再設定

索引

A

Active Timeout, 266

B

Bean-Managed Transaction, 272

Bean管理トランザクション, 272

C

Client Transaction Manager, 256

Client-Managed Transaction, 270

Commit Timeout, 266

Container-Managed Transaction, 255, 274

Custom Resource, 7

D

DAS, 3

Data Source, 6

Domain Administrator Server, 1

E

EJBエンジン, 7

External Source, 7

G

Global Transaction, 254, 255

H

HTTPセッション・クラスタリング・サービス, 5

I

IBM MQ, 110

Incomplete Timeout, 267

InitialContext, 92

J

JAXR Source, 7

JCAリソース・マネージャー, 257

JDBCリソース・マネージャー, 257

JEUS Transaction Manager, 253

jeus.server.enable.restart.in.memory.shortage, 90

jeus.server.memorymonitor.duration, 90

jeus.server.memorymonitor.enabled, 90

jeus.server.memorymonitor.interval, 90

jeus.server.memorymonitor.ratio, 90

jeus.tm.noLogging, 281

jeus.tm.recoveryInterval, 281

jeus.tm.recoveryTrial, 281

jeus.tm.tmMax, 265

jeus.tm.tmMin, 265

jeusadmin, 63

JEUSTランザクション・マネージャー, 253

JMSエンジン, 8

JMSリソース・マネージャー, 257

JNDI, 91

JNDIクラスタリング, 95

JNDIサービス, 5

JNDIツリー, 91

JNDIネーミング・サーバー, 5, 91

JNSClient, 91, 94

JNSContext.CONNECT_TIMEOUT, 101

JNSContext.CONNECTION_DURATION, 101

JNSContext.RESOLUTION, 101

JNSServer, 91, 94

L

Local Transaction, 254, 255, 268

M

Mail Source, 6

Managed Server, 2, 4

Managed Serverサービス, 4

Message Bridge, 7

P

Prepare Timeout, 266

Prepared Timeout, 266

R

Recovery LogFile, 279
Recovery Timeout, 267
Resource Manager, 256

S

Server Transaction Manager, 255
Sonic MQ, 110

T

Tm Config
 Active Timeout, 262
 Automatic Recovery, 262
 Commit Timeout, 262
 Incomplete Timeout, 263
 Prepared Timeout, 262
 Recovery Timeout, 262
 Rprepare Timeout, 262
Tmax (WebT)リソース・マネージャー, 257

U

URL Source, 6
URLリソース, 109
User Managed Transaction, 254

W

WebAdminサービス, 2, 3
Webエンジン, 7

あ

エンジン, 7
エンジン・サービス, 4

か

外部リソース, 110
外部リソース, 6
共用スレッド・プールの設定, 98
カスタム・リソース, 110
クライアント側のJNSClient, 95
クライアント側のJNSClientの設定, 101
クライアント管理トランザクション, 270

クライアント・トランザクション・マネージャー, 256
クラスFTPサービス, 6
クラスター管理サービス, 2, 4
クラス・ローダーの構造, 8
グローバル・トランザクション, 254, 255
コンテナ管理(Container Managed)トランザクション, 255
コンテナ管理トランザクション, 274

さ

初期コンテキスト, 101, 102
サーバー・トランザクション・マネージャー, 255
サーバー側のJNSClient, 94
サーバー側のJNSClientの設定, 101
サーバー実行スクリプト, 63
サービス専用スレッド・プールの設定, 98
サーブレット・エンジン, 7
スケジューラー・サービス, 6
スレッドのインターラプト, 84
セキュリティー・サービス, 5

た

動的設定反映サービス, 2, 3
独立したクラス・ローダー, 8
データソース, 109
データベース接続サービス, 6
トランザクション・サービス, 6
ドメイン・アプリケーション管理サービス, 2, 3
ドメイン・データソース管理サービス, 2, 4

は

並行処理ユーティリティー・リソース, 110
ハンドラー
 SMTPハンドラー, 313
 ソケット・ハンドラー, 314
 ファイル・ハンドラー, 311
 ユーザー・ハンドラー, 315

ま

マネージメント・サービス, 5
メッセージ・ブリッジ, 110
メール・リソース, 109

や

ユーザー管理(User Managed)トランザクション, 254

ら

リソース・マネージャー, 256

ルックアップ, 102

ロギング・サービス, 6

ログ・ファイルのローテーション, 316

ローカル・トランザクション, 254, 255, 268

