JRun Version 3.1
機能および移行ガイド

Windows®、UNIX™、および
Linux™ 用 JRun 3.1

Allaire® Corporation

版権告知

本書とその中に記載されているソフトウェアは、ライセンス契約のもとに供給され、このライセンスの条項に従ってのみ使用または複製することができます。本書の内容は、情報の提供のみを目的としており、予告なく変更されることがあります。これについて、Allaire Corporation は一切責任を負いません。Allaire Corporation は、本書の誤りについて一切責任を負いません。

ライセンスによる許可がある場合を除いて、Allaire Corporation の事前の書面による許可なしに、この出版物の一部または全部の複製、検索システムへの保存、あるいは電子的、機械的な記録、または他のいかなる形態や手段による転送を行うことはできません。

ColdFusion および HomeSite は、米国における Allaire Corporation の登録商標です。Allaire、Allaire ロゴ、Allaire Spectra、Allaire Spectra ロゴ、および JRun は、Allaire Corporation の商標です。Java は、Sun Microsystems Inc. の商標です。Microsoft、Windows、Windows NT、Windows 95、Microsoft Access、および FoxPro は、Microsoft Corporation の登録商標です。PostScript は、Adobe Systems Inc. の商標です。Solaris は、Sun Microsystems Inc. の商標です。UNIX は、The Open Group の商標です。その他の会社名、商標名、および製品名は、各所有者に帰属する商標です。

この製品には RSA Data Security からライセンス供与されたコードが含まれています。このソフトウェアの著作権の一部は、Merant, Inc. に帰属します。1991-2001

部品番号 : ZJR31MFMGJ

目次

本書の概要

『JRun Version 3.1 機能および移行ガイド』では JRun バージョン 3.1 の新機能について説明します。また、バージョン 3.0 からバージョン 3.1 への移行に関連した検討事項についても説明します。

目次

対象読者

『JRun Version 3.1 機能および移行ガイド』は、JRun バージョン 3.1 にアップグレードし、新機能を利用する必要がある方を対象としています。新機能、変更、バグ修正、および J2EE 互換性テスト 群 (CTS) に合格するための JRun 設定の一部として行われた機能変更について説明します。

開発者リソース

(株) アイ・ティ・フロンティア (株式会社シリウスは、2001 年 4 月に株式会社アイ・ティ・フロンティアに社名変更いたしました) では、開発者の教育、テクニカルサポートなどのサービスによりカスタマサポートを充実させております。次の表に記載されているサイト では、各サービスの内容の詳細が掲載されています。

リソース	説明	URL
(株)アイ・ティ・フロンティア JRun のサイト	JRun の詳細な製品情報および関連トピック	http://cfusion.sirius.co.jp/jrun/
Allaire社 Web サイト	開発元である Allaire 社のサイト	www.allaire.com/
JRun サポート フォーラム	オンライン フォーラムでは豊かな経験を持つ JRun 開発者と連絡をとり、JRun に関連した数多く のトピックについてメッセージを書き込んだり、回答を得ることができます。	http://forums.allaire.com/jrunconf/
開発者 コミュニティ	JRun による開発に必要な最先端の情報を提供する、オンライン ディスカッション グループ、知識ベース、技術文書などのあらゆるリソース	www.allaire.com/developer/
JRun 開発者 センター	開発のヒント、記事、文書、ホワイトペーパーに関する情報サイト	www.allaire.com/developer/jrunreferencedesk/

JRun 文書の概要

JRun 文書は、JSP 開発者、サーブレット 開発者、EJB クライアント 開発者、EJB 開発者、システム管理者を含むすべての JRun ユーザにサポートを提供することを目的としています。印刷物で提供されている場合でも、オンラインの場合でも、必要な情報を速やかに探し出せるように構成されています。JRun オンライン文書には、HTML 形式と Adobe Acrobat ファイル形式があります。

印刷およびオンライン文書セット

JRun の文書セットには、次の文書があります。

文書	説明
『JRun セットアップ ガイド』	JMC を使用した JRun のインストール、設定、および管理について説明します。
『JRun による アプリケーションの開発』	Java サーブレット、JavaServer Pages、および Enterprise JavaBeans から構成される Web アプリケーションの開発方法について説明します。
『JRun サンプル ガイド』	サーブレット、JavaServer Pages、Enterprise JavaBeans のコード サンプルおよびサンプル アプリケーションを提供します。
『JRun タグ ライブラリ リファレンス』	JRun タグ ライブラリの JavaServer Pages (JSP) カスタム タグについて説明します。
『JRun 拡張設定ガイド』	ISP、ISV、および OEM カスタマ用の JRun のインストール、使用、設定に関する情報があります。
『JRun JSP クイック リファレンス』	JavaServer Pages (JSP) のディレクティブ、アクション、およびスクリプト 要素の簡単な説明と構文が記載されています。
『JRun Version 3.1 機能および移行ガイド』	JRun バージョン 3.1 の機能と、既存のアプリケーションをバージョン 3.1 に移行する方法について説明します。
『JRun タグ ライブラリ クイック リファレンス』	JRun タグ ライブラリの JavaServer Pages (JSP) カスタム タグの簡単な説明と構文について記載されています。

オンライン文書へのアクセス

すべての JRun 文書は、HTML 形式と Adobe Acrobat ファイル形式でオンラインで利用できます。文書にアクセスするには、JRun を実行しているサーバーにある URL、*JRun のルート ディレクトリ*/docs/dochome.htm を開きます。

JRun Studio 文書

JRun Studio には、JRun 文書が完全に収録された広範なオンライン サポート があります。JRun Studio 内から JRun オンライン文書を表示するには、[Help] リソース タブをクリックします。Web プログラミングに関する情報のほかに、JRun および JRun Studio に関する文書の展開可能な一覧が表示されます。

JRun Studio オンライン文書には検索機能があり 、各ページをブックマークに追加できます。詳細については、JRun Studio 文書セットを参照してください。

メモ
JRun Studio は開発スケジュールが異なるため、JRun バージョン 3.1 のリリース後に入手できます。

その他のリソース

本書で扱っているトピックの詳細については、次のリソースも参照してください。

書籍

サーブレットと JavaServer Pages	
『Java Servlets』	Karl Moss 著、 McGraw Hill 刊、1999 年、 ISBN: 0071351884
『JavaServer Pages Application Development』	Scott Stirling 他著、 Sams 刊、2000 年、 ISBN: 067231939X
『Java Servlet Programming (Second Edition)』	Jason Hunter、William Crawford 著、 O'Reilly & Associates 刊、2001 年、 ISBN: 0596000405
『Core Servlets and Java Server Pages』	Marty Hall 著、 Prentice Hall 刊、2000 年、 ISBN: 0130893404
『Inside Servlets: Server-Side Programming for the Java Platform (Second Edition)』	Dustin R. Callaway 著、 Addison-Wesley 刊、2001 年、 ISBN: 0201709066
『Web Development with JavaServer Pages』	Duane K. Fields、Mark A. Kolb 著、 Manning Publications 刊、2000 年、 ISBN: 1884777996

Enterprise JavaBeans	
『Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition』	Ed Roman 著、 John Wiley & Sons 刊、1999 年、 ISBN: 0471332291
『Enterprise JavaBeans』	Richard Monson-Haefel 著、 O'Reilly & Associates 刊、2000 年、 ISBN: 1565928695
『Applying Enterprise JavaBeans: Component-Based Development for the J2EE Platform』	Vlada Matena、Beth Stearns 著、 Addison-Wesley Pub Co 刊、2000 年、 ISBN: 0201702673

Enterprise Java プログラミング	
『Professional Java Server Programming J2EE Edition』	Danny Ayers 他著、 Wrox Press 刊、2000 年、 ISBN: 1861004656
『Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition』	Nicholas Kassem 著、 Addison-Wesley 刊、2000 年、 ISBN: 0201702770 (java.sun.com/j2ee/download.html#blueprints から無料ダウンロード可能)
『Building Java Enterprise Systems with J2EE』	Paul Perrone、Venkata S.R. "Krishna" R. Chaganti 著、 Sams 刊、2000 年、 ISBN: 0672317958
『J2EE: A Bird's Eye View (e-book)』	Rick Grehan 著、 Fawcette Technical Publications 刊、2001 年、 ISBN: B00005BAZV

オンライン リソース

Java servlet API	java.sun.com/products/servlet
JavaServer Pages API	java.sun.com/products/jsp
Enterprise JavaBeans API	java.sun.com/products/ejb/
Java 2 Standard Edition API	java.sun.com/products/jdk/1.3/docs/api/index.html
Servlet Source	www.servletsource.com
JSP Resource Index	www.jspin.com
Server Side	www.theserverside.com
Dot Com Builder	dcb.sun.com
Servlet Forum	www.servletforum.com

問題の解決方法

プログラミングの問題を解決する最善の方法の 1 つは、JRun フォーラム で、JRun 開発者コミュニティの幅広い経験に基づいたアドバイスを受けることです。JRun の利用方法についてどのようなことでも、メンバーであるほかの開発者のアドバイスを得ることができます。さらに、検索機能を使用すると、過去 12 か月間のメッセージを呼び出すことができるため、同じ問題をほかの開発者がどのように解決したかを知ることができます。

お問い合わせ先

販売元

株式会社アイ・ティ・フロンティア
シリウス事業部

電話 : 03-5562-4099
Fax : 03-5562-4070

http://cfusion.sirius.co.jp/jrun/
E-mail : jrunsales@sirius.co.jp

(株式会社シリウスは、2001 年 4 月に株式会社アイ・ティ・フロンティアに社名変更いたしました)

テクニカルサポート

Allaire 社では、電話および Web による幅広いサポート オプションを提供しています。テクニカル サポート サービスについては、http://www.allaire.com/support/ をご覧ください。

JRun サポート フォーラム (http://forums.allaire.com) へは、いつでも投稿できます。

第 1 章

JRun バージョン 3.1 の機能

JRun バージョン 3.1 にはさまざまな新機能があります。

目次

概要

JRun バージョン 3.1 の主な特徴は CTS 準拠ですが、多くの新機能も追加されました。これらの新機能は、開発者が簡単にアプリケーションの作成、テスト、および公開を行うことができるように選択され、設計されました。

機能

バージョン 3.1 には次のような新機能があります。

- 「SSL サポート」
- 「JSP で書かれたカスタム タグ」
- 「Web アプリケーションの動的な公開および更新」
- 「カスタム タグ ハンドラのキャッシュへの格納」
- 「CMP メソッドのメソッド パラメータ サポート」
- 「JMC の強化」
- 「JSPC の強化」
- 「新規サンプルと強化されたサンプル」
- 「文書の改訂」
- 「組み込み型データベース」

SSL サポート

Secure Sockets Layer (SSL) は、インターネットでのメッセージ転送のセキュリティの管理に最も一般的に使用されているプロトコルです。JRun 3.1 では JRun Web サーバーで SSL をサポートし、JMC からこのサポートを公開します。これにより、非常に安全な Web アプリケーションを作成できます。

メモ

JRun 3.1 SSL 機能には、ブラウザと JRun のビルト イン Web サーバー間の安全なソケット がサポート されています。JRun 3.1 では、ブラウザと ネイティブ Web サーバー間のソケット は保護されていません。ほとんどの商用 Web サーバーにはこの機能が組み込まれているため、JRun でこの機能を提供する必要はありません。ただし、Apache は例外です。Apache の場合は SSL モジュールを追加するか、または SSL モジュールを追加しているベンダから Apache を購入する必要があります。ネイティブ Web サーバーと JRun のコネクタの間のソケットでは、SSL がサポートされていないことに注意してください。

SSL の詳細については、『SSL and TLS Essentials』(Stephen Thomas 著、Wiley Computer 刊、2000 年、http://www.amazon.com/exec/obidos/ASIN/0471383546) を参照してください。

異なる Web サーバーの SSL チュートリアルなどの便利なオンライン リソースは、http://www.verisign.com にあります。

さらに、SSL の詳細な技術情報には次のようにアクセスします。

- keytool ユーティリティおよび keystores の説明については、http://java.sun.com/j2se/1.3/docs/guide/security/spec/security-spec.doc6.html を参照してください。
- サーバー証明書のセットアップについては、http://java.sun.com/j2ee/j2sdkee/techdocs/guides/ejb/html/Security7.html#11638 を参照してください。
- `KeyStore`、`Certificate`、および `PrivateKey` クラスについては、J2EE JavaDocs を参照してください。

JSP で書かれたカスタム タグ

JRun 3.1 では、JSP で書かれたカスタム タグをサポートしています。JRun Server Tags (JST) と呼ばれるこの技術を使用すると、Java ベースのカスタム タグ API ではなく JSP を使用してカスタム タグを構築できます。Java コード は必要ありません。JST 技術では JSP 構文が使用されているので、JSP プログラマはカスタム タグを利用できます。JST を使用すると、Java で書かれたカスタム タグに比べて短期間でアプリケーションを開発できます。ページが JST ページであることを示すには、そのページの名前に拡張子 `.jst` を付けます。

この機能は、『JRun によるアプリケーションの開発』に記載されています。

Web アプリケーションの動的な公開および更新

JRun バージョン 3.1 では、Web アプリケーションを動的に更新して自動的に公開する機能が用意されています。これは主に操作性を向上させる機能であり、これにより次の機能が提供されます。

- **ホット デプロイ** たとえば、更新された WAR ファイルや web.xml ファイルへの変更など、既存の Web アプリケーション構造への物理的な変更が検出されると、実行中の Web アプリケーションが自動的に再起動されます。
- **オート デプロイ** 新しい WAR ファイルが検出されると、新しい Web アプリケーションが自動的に公開されます。

ホット デプロイおよびオート デプロイを有効にすると、Web アプリケーションへの変更後や新しい Web アプリケーションの追加後に、JRun サーバーを再起動する必要はありません。ただし、JDBC データ ソースの追加などのサーバーレベルの変更後は、JRun サーバーを再起動する必要があることに注意してください。

メモ
運用環境ではこの機能を無効にすることを強くお勧めします。

JRun 3.1 とともに提供されている Web サーバーおよびアプリケーションでは、この設定を次のように使用します。

- admin server　無効 (admin/local.properties で `webapp.hotdeploy.enabled=false` にします。)
- jmc-app (admin server)　無効 (admin/jmc-app/WEB-INF/webapp.properties で `webapp.hotdeploy.enabled=false` にします。)
- default server　有効 (global.properties で `webapp.hotdeploy.enabled=true` から引き継がれます。)
- default-app　有効 (global.properties で `webapp.hotdeploy.enabled=true` から引き継がれます。)
- demo-app　有効 (global.properties で `webapp.hotdeploy.enabled=true` から引き継がれます。)
- invoice-app　有効 (global.properties で `webapp.hotdeploy.enabled=true` から引き継がれます。)

ホット デプロイ

動的な公開の制御は、global.properties および local.properties にある次のプロパティによって行います。

- `webapp.hotdeploy.enabled=true|false`　動的な公開を有効または無効にします。
- `webapp.hotdeploy.interval=`*間隔*　JRun によって変更がチェックされる頻度を秒単位で指定します。既定値は 5 秒です。この値を 0 に設定すると、ホット デプロイ サービスは無効になります。
- `webapp.hotdeploy.onchange=`*ディレクトリおよびファイルのカンマ区切りリスト*　JRun によって監視されるディレクトリおよびファイルを指定します。これらのファイルのいずれかが変更されると、Web アプリケーションが自動的に再起動され、「hot deploy initiating」というメッセージがログ ファイルに書き込まれます。global.properties の既定の設定は次のとおりです。

  ```
  webapp.hotdeploy.onchange={webapp.rootdir}/WEB-INF/web.xml,
   {webapp.rootdir}/WEB-INF/webapp.properties,
   {webapp.rootdir}/WEB-INF/lib
  ```

- `webapp.hotdeploy.defaultname=`*default-app war ファイル*　既定のコンテキスト (URL の /) にマッピングされる WAR ファイルの名前を指定します。既定値は default.war です。Web アプリケーションでは通常、コンテキストとして */webappname* を使用しますが、既定のアプリケーションではスラッシュだけを使用します。詳細については、『JRun によるアプリケーションの開発』の「JRun によるサーブレットへの要求マッピング」を参照してください。

変更したファイルが WAR ファイルである場合は、以前に定義されたルート ディレクトリおよび URL マッピングを使用して Web アプリケーションを再公開し、その後で再起動します。変更したファイルが WAR ファイルでない場合は、Web アプリケーションの再起動のみを行います。

オート デプロイ

オート デプロイの制御は、global.properties および local.properties にある次のプロパティによって行います。

- `webapp.hotdeploy.autodeploy=` *ファイルの場所* 指定した場所で検出されたすべての新しい WAR ファイルが、自動的に公開されます。サーバーの起動時にこの場所が確認され、サーバーの実行中は新しいファイルの監視も行われます。

global.properties の既定の設定は次のとおりです。

```
webapp.hotdeploy.autodeploy=
        {jrun.rootdir}/servers/{jrun.server.name}/deploy/*.war
```

自動的に WAR ファイルが公開されると、拡張子のない WAR ファイル名を使用して Web アプリケーション名および URL マッピングが作成され、WAR ファイルが /servers/server name/*Web アプリケーション名*というディレクトリに展開されます。たとえば、default JRun サーバーの deploy ディレクトリのルート レベルに newapp.war を追加すると、次の項目が作成されます。

- **Web アプリケーション名** newapp
- **URL マッピング** /newapp
- **アプリケーションのルート ディレクトリ** *default JRun サーバーのルート* /newapp

deploy ディレクトリの下のサブディレクトリに WAR ファイルを格納すると、JRun で作成されるディレクトリ構造を制御できます。たとえば、deploy/accounting/payroll.war に WAR ファイルを保存すると、/accounting/payroll という URL マッピングを持つ accounting-payroll と命名された Web アプリケーションが作成されます。

カスタム タグ ハンドラのキャッシュへの格納

JRun 3.1 では、タグ ハンドラをキャッシュすることによってカスタム タグのパフォーマンスが向上しています。ハンドラ 開発者は次のことに注意してください。

- タグ ハンドラは、ページのインスタンスごとにキャッシュされます。たとえば、JSP ページで同じカスタム タグが 2 回呼び出された場合は、2 つのインスタンスがキャッシュされます。
- タグ ハンドラでインスタンスレベルの変数を保持するか、またはその他のステート管理を行う場合は、タグ インスタンスがキャッシュされる前に、`Tag.release` メソッドを実装して変数をリセットする必要があります。詳細については、17 ページの「カスタム タグ ハンドラの移行」を参照してください。

CMP メソッドのメソッド パラメータ サポート

JRun 3.1 には、SQL パラメータをメソッド引数から EJB メソッドに渡すためのサポートが含まれています。JRun 3.0 では、CMP メソッドはこの情報を Bean インスタンス フィールドからしか取得できませんでした。現在は、CMP Bean の SQL で使用する引数を渡す Bean メソッドを呼び出すことができます。メソッドの引数を使用して SQL を置き換える場合は、疑問符の後に数値を付けることによって SQL パラメータとメソッド引数を関連付けます。この疑問符と数値はメソッド引数内の位置を表します。たとえば、コンテナでは ?1 をメソッドの最初の引数に、?2 をメソッドの 2 番目の引数にというように置き換えます。たとえば、次のように指定します。

```
select key from table where value >?1 and value < ?2
```

ここで、?1 と ?2 は、検索メソッドの 1 番目と 2 番目の引数を表します。

次のことに注意してください。

- このテクニックは、クエリ フィールドが必ずしも Bean インスタンス変数に対応するとは限らない multi-row 検索メソッドに最も有効です。
- Bean インスタンス変数と一致する SQL パラメータには、`param` および `paramTypes` env-entry が必要ですが、メソッド引数を介して渡される SQL パラメータには必要ありません。
- これらのテクニックを組み合わせることができます。つまり、1 つの SQL ステートメントで、疑問符と、疑問符の後に数値を置く形式を使用できます。

また、このテクニックは、次の例に示すように `create` メソッドと併用できます。

```
<env-entry>
  <env-entry-name>ejipt.createSQL</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>
    INSERT INTO account (id, value) VALUES (?1, ?2)
  </env-entry-value>
</env-entry>
```

この env-entry に対応する `ejbCreate` メソッドには、次の例に示すように初期化コードは不要です。

```
public void ejbCreate(int accountId, int value)
  throws CreateException, RemoteException { }
```

クライアント側の `create` メソッド呼び出しでは、次のように `id` および `value` 引数を渡します。

```
...
BalanceHome home = (BalanceHome)javax.rmi.PortableRemoteObject.narrow
    (_context.lookup("sample3a.BalanceHome"), BalanceHome.class);
...
Balance b = null;
b = bh.create(key, 0);
...
```

JMC の強化

2 つの一般的な管理タスクによって、追加の JRun サーバー インスタンスが公開され、JRun ログ ファイルが表示されます。JRun 3.1 JMC では、次の点が強化されています。

- **新規サーバーの追加**　JMC を使用して JRun サーバーの追加および削除を行うことができます。
- **ログ ファイルの表示**　JMC を使用すると、ファイル システムから Event、Stderr、および Stdout ログに直接アクセスしなくても、それらのログを表示できます。

JSPC の強化

JSPC コンパイラは、Web サーバーのコンテキスト の外部での JSP ページのコンパイルに使用するコマンドライン ツールです。JSPC コンパイラを使用すると、JRun を使用して JSP ページの要求時にページをコンパイルすることなく、コマンド ラインから JSP ページを明示的にコンパイルできます。

JRun 3.1 では JSPC 構文が変更されました。新しい構文は次のとおりです。

```
java [-classpath classpath]
  JSPC
  -j jrun_root_dir
  -s server
  -a web_application_name
  [-webroot path_to_web_root]
  [-vghn]
  [-d output_directory]
  [-compiler "compiler_spec"]
  JSP_path
```

引数の定義は次のとおりです。

- `classpath` - (オプション) Java コマンドへのクラスパスを指定します。このクラスパスは、コマンド ラインから指定したり、CLASSPATH 環境変数を使用して指定できます。クラスパスでは、次のパスを指定する必要があります。
  - *JRun のルート ディレクトリ*¥lib ディレクトリ内のすべての .jar ファイル
  - *JRun のルート ディレクトリ*¥lib¥ext ディレクトリ内のすべての .jar ファイル
  - 使用するコンパイラによって、JDK に提供されているほかのファイルを指定しなければならない場合もあります。
- `j` - JRun ルート ディレクトリを指定します。
- `s` - サーバー名を指定します。
- `a` - Web アプリケーション名を指定します。
- webroot - (オプション) Web アプリケーションのルート ディレクトリへの絶対パス。この引数は、`use-webserver-root` プロパティが TRUE に設定されている Web アプリケーションに必要です。これを `-w` に短縮することもできます。
- `v` - (オプション) JSPC によるコンパイル時に、各 JSP ファイルの名前を表示するように指定します。

- g - (オプション) デバッグ メッセージを表示するように指定します。
- h - (オプション) JSPCコンパイラのヘルプ メッセージを表示するよう に指定します。
- n - (オプション) 対応する .classファイルよりも新しい JSP ページのみをコンパイルするように指定します。
- d - (オプション) JSPCコンパイラが出力 .class および .java ファイルを書き込む場所を指定します。既定では、このディレクトリ は現在の作業ディレクトリ に設定されます。通常は、Web アプリケーションのディレクトリ構造の WEB-INF/jsp ディレクトリを指定します。
- compiler - (オプション) コンパイラおよびコンパイルに使用するコンパイラ設定を含む、引用文字列を指定します。このパラメータを省略すると、Sun javac コンパイラを使用して、インプロセスで JSP ページがコンパイルされます。次に例を示します。

  ```
  java JSPC -compiler "javac -nowarn -classpath %c -d %d %f" ...
  ```

  または

  ```
  java JSPC -compiler "jvc /cp:c %c /dest:%d %f" ...
  ```

- JSP_path - ファイルシステム上の JSP ページの物理パスを、-a パラメータによって Web アプリケーションへの相対パスで指定します。スペースで区切って、複数のファイルを指定できます。ワイルド カード 文字を使用して、複数のファイルを指定することもできます。-a によって指定されたディレクトリと JSP_path を組み合わせて、コンパイルする JSP ページを指定します。たとえば、-a が c:¥myapps¥store にある Web アプリケーションであり、コンパイルする JSP ページが c:¥myapps¥store¥my.jsp の場合、ページの JSP_path は my.jsp となります。複数の JSP ページをコンパイルする 場合は、JSP_path にワイルド カード 文字を指定できます。

メモ

JSPC コンパイラを使用して、use-webserver-root プロパティが TRUE に設定されている Web アプリケーションを処理するときは、-webroot 引数を指定する必要があります。たとえば、default JRun サーバーにある既定のアプリケーションでは、use-webserver-root が TRUE に設定されています。詳細については、『JRun によるアプリ ケーション の開発』を参照してください。

新規サンプルと強化されたサンプル

JRun バージョン 3.1 には強化されたサンプルがあります。以前と同様に、サンプルは default JRun サーバーで実行し、JMC の 「ようこそ」ページからアクセスできます。サンプルでは次の点が強化されています。

- **新規の Web アプリケーション**　サンプルには、invoice-app という名前の Web アプリケーションが追加されました。このアプリケーションの特徴は、Fax Cover Sheet Generator および Invoice Generator です。Invoice Generator には 3 つのバーションがあり、それぞれ異なるコーディング方法 (純粋な JSP、JavaBean、および JSP カスタム タグ) を示します。この Web アプリケーションには、使用方法の注意とソース コードのカラー表示があります。
- **データベースの統合**　サンプルは組み込み型 PointBase データベースに同梱されています。カスタム タグ サンプルおよび EJB サンプルは、このデータベースを使用するように変更されています。
- **強化されたカスタム タグ サンプル**　追加のカスタム タグ サンプルが作成されました。さらに、既存のカスタム タグ サンプルが整理され、データベース アクセス コードが追加され、ソース コードのカラー表示へのアクセスが容易になりました。
- **強化された EJB サンプル**　EJB サンプルがサンプル データベースと統合され、より標準的な EJB コーディング サンプルを提供します。また、追加のサーブレット サンプルである 9b も追加されました。リレーショナル データベースの追加により、instance.store の使用法を示したサンプルである 1a、1b、4a、および 7a が削除されました。

文書の改訂

JRun バージョン 3.1 では、文書が次のように改訂されました。

- 『JRun Version 3.1 機能および移行ガイド』(本書) では新機能の概要と、JRun 3.0 から 3.1 への移行に関連した問題について説明します。
- 『JRun セットアップ ガイド』には「コネクタ」の章が含まれています。この章は、以前は『JRun 拡張設定ガイド』に記載されていました。
- 『JRun タグ ライブラリ リファレンス』は、内容が完全に変更されました。
- 『JRun サンプル ガイド』では、サンプル アプリケーションおよび EJB サンプルの変更が反映されています。
- 『JRun によるアプリケーションの開発』の EJB 文書は、CTS ベースの機能変更が反映されています。

JRun 3.1 には PDF 形式と HTML 形式の文書が同梱されています。各 HTML ページには、http://livedocs.allaire.com の対応するページへのリンクがあります。このリンクを使用して、ほかの JRun 開発者が書き込んだコメントを確認して、コメントを追加できます。

組み込み型データベース

JRun バージョン 3.1 には、組み込み型 PointBase DBMS を使用するサンプル データベースが同梱されています。このデータベースは主に、JRun カスタム タグおよび EJB サンプルのサポートを目的としています。このデータベースは、開発データベースまたは公開データベースとしての使用を予定していません。このデータベースは簡単なテストに使用できますが、事前に制限事項について認識しておく必要があります。

制限

PointBase が組み込まれているデータベースは、接続すると自動的に起動されますが、1 つの JVM からの接続に限定されます。これは次のことを意味しています。

- JMC JDBC データソースの [Test] ボタンは、PointBase が組み込まれたデータベースについては無効になっています。
- PointBase Console を使用するには、default サーバーを停止しておく必要があります。
- EJB サンプル 4b および 5a は複数の接続を使用するため、機能しません。

JRun バージョン 3.1 に同梱の組み込み型 PointBase データベースのデータは、5 MB に限定されます。

組み込み型データベースの使用

JRun には、サンプル データベース内のデータに追加や変更を行うユーティリティが同梱されています。これらのユーティリティは *JRun のルート ディレクトリ*/pointbase ディレクトリにあります。ユーティリティ名は次のとおりです。

- `commander` (UNIX) および `commander.bat` (Windows)　コマンドライン インターフェイスによって SQL ステートメントを入力できます。
- `console` (UNIX) および `console.bat` (Windows)　Swing ベースの GUI からデータベースのテーブルおよび行を変更できます。

メモ
これらのユーティリティは、default JRun サーバーを停止してから実行してください。

詳細については、*JRun のルート ディレクトリ*/pointbase/docs ディレクトリにある PointBase のマニュアルを参照してください。

PointBase Console では、新しいデータベースを作成できます。サーブレット、JSP、および EJB からこのデータベースにアクセスするには、次の設定値を使用してください。

- **ドライバ** `com.pointbase.jdbc.jdbcUniversalDriver`
- URL `jdbc:pointbase://embedded/`*データベース名*
- **ユーザ名** PUBLIC (既定値)
- **パスワード** PUBLIC (既定値)

サンプル データベースは、ネットワーク バージョンの PointBase DBMS と互換性があります。評価バージョンおよびアップグレード情報については、http://www.pointbase.com を参照してください。

ロック ファイルの削除

JRun の全サーバーを停止しても、`commander` または `console` ユーティリティから PointBase データベースにアクセスできない場合は、*JRun のルート ディレクトリ*/pointbase/databases ディレクトリに *.lck ファイルがあるかどうかを確認してください。*.lck ファイルは PointBase が異常終了した場合に生成されます。これらのファイルを削除できない場合は、次の手順を実行してください。

1 コンピュータを再起動します。

2 再起動で JRun が自動的に実行されたら停止します。

3 *JRun のルート ディレクトリ*/pointbase/databases ディレクトリからすべての *.lck ファイルを削除します。

第 2 章

JRun Version 3.1 への移行

CTS テストのサポートにおいて JRun バージョン 3.1 に多くの変更が加えられました。開発者がこれらの変更のほとんどに気付くことはないと思われますが、JRun の管理および開発に影響を与える変更がいくつかあります。この章では、移行に関する検討事項と、JRun アプリケーションをバージョン 3.0 から 3.1 に移行する際に行う作業について説明します。

目次

概要

Allaire Corp. は JRun version 3.0 のリリース後間もなく Sun Microsystems との戦略的提携を発表し、正式に Sun の Java 2 Enterprise Edition (J2EE) のライセンス供与を受けました。J2EE 規格に対する Allaire の責任の一環として、J2EE 互換性テスト群 (CTS) を使用して JRun の互換性を確認しています。CTS は、JavaServer Pages (JSP)、サーブレット、および Enterprise JavaBeans (EJB) がほかの J2EE 対応アプリケーションサーバーとの互換性を維持しながら実行されることを確認できる 5,000 以上の包括的なテスト セットです。バージョン 3.1 は JRun の最初の CTS 互換リリースとなります。

CTS スクリプトとテスト ハーネスを実行する際に多数の問題が発生したため、JRun、特に EJB エンジンへの変更が必要になりました。これらの変更は次のカテゴリに分類されます。

- **仕様への準拠** CTS では JSP、サーブレット、EJB、J2EE のすべての仕様に対して適合性をテストします。仕様に関連する非一貫性が明らかになり、修正されました。アプリケーションがこれらのいずれかの動作に依存している場合は、サーブレット、JSP、EJB、EJB クライアント コードのいずれかの変更が必要になることがあります。
- **JNDI アーキテクチャ** JRun のサーブレット エンジンおよび EJB エンジンは 2 つの別個の JNDI ネームスペースを保持していました。CTS テストに合格するために、これらのネームスペースが統合されました。
- **EJB 公開記述子への移行** EJB バージョン 1.1 仕様書では、Bean の情報を公開ツールに伝えるために XML 公開記述子のサポートが必須になっています。JRun 3.0 の EJB エンジンでは公開記述子がサポートされていましたが、主にプロパティ ファイルに依存していました。そのため、特定の公開記述子機能がサポートされていませんでした。この問題が修正されました。
- **認証** JRun サーブレット エンジンと EJB エンジンでは認証情報を共有していませんでした。1 つの認証ソースを使用したシングル サインオンを実行できるようになりました。
- **新しい機能** CTS に合格するには、Secure Sockets Layer (SSL) だけでなく、ほかの必要な機能もサポートされている必要があります。

この章ではこの後、JRun 3.1 の移行、CTS 関連の変更、および必要なコード 変更について説明します。

移行のメリット

J2EE 準拠アーキテクチャに移行するメリットについて説明します。次のメリットがあります。

- **移植性** 特定の J2EE 準拠アプリケーション サーバー上で動作するアプリケーションは、ほとんど変更せずにほかの J2EE 準拠アプリケーション サーバー上でも動作するはずです。
- **SSL** HTTPS クライアントと JRun Web サーバー間で保護された通信が行われます。
- **今後のリリースへの準備** JRun の今後のリリースはすべてこのアーキテクチャに準拠します。この時点で移行することによって、アプリケーションを JRun の今後のリリースに簡単に移行できるようにします。
- **より密接な統合 (シングル サインオン、結合されたデータ ソース)** シングル サインオン機能により、1 セットのコードを使用して 1 つのユーザ レポジトリから認証を制御できます。データ ソースが結合されたため、EJB は JNDI を使用して JRun データ ソースにアクセスできます。

移行による影響

JRun には J2EE アプリケーション サーバーのすべての機能が用意されていますが、多くのアプリケーションですべての機能を使用することはありません。たとえば、サーブレットと JSP だけを使用するアプリケーションや、スタンドアロン EJB エンジンだけを使用するアプリケーション (特に OEM アプリケーション) があります。JRun 3.1 への移行による影響は、次のように JRun の使用によって異なります。

- **サーブレット、JSP、および EJB をすべて使用する場合** EJB エンジンを変更すると、公開記述子、プロパティ ファイル、EJB、EJB クライアント、認証メカニズムのすべての変更が必要になる場合があります。本書をよく読み、必要に応じて変更を適用し、運用アプリケーションを JRun バージョン 3.1 に移行する前にアプリケーションを十分にテストしてください。
- **サーブレットと JSP のみ** これに関しては最小限の変更が行われました。アプリケーションにはほとんど変更が不要なはずです。JRun バージョン 3.1 の機能リストを確認して、アップグレードによる影響を調べる必要があります。アップグレードする場合は、運用アプリケーションを JRun バージョン 3.1 に移行する前に必ずアプリケーションを十分にテストしてください。
- **スタンドアロン EJB エンジンのみ** 可能な場合、JRun 3.1 は下位互換性を維持します。JRun バージョン 3.1 の機能リストを確認して、アップグレードによる影響を調べる必要があります。アップグレードする場合は、運用アプリケーションを JRun バージョン 3.1 に移行する前に必ずアプリケーションを十分にテストしてください。

移行手順

既存のアプリ ケーショ ンを J2EE 準拠 JRun 3.1 アプリ ケーショ ンに移行する 手順は次のとおりです。

1 サーブレットと JSP を移行します。

2 Bean のプロパティを公開記述子に移行し ます。

3 deploy.properties を local.properties に移行します。

4 deploy.properties データソースを JRun データソースに移行し ます。

5 EJB クライアントを移行します。

6 EJB 認証 / 承認をシングル サインオンに移行し ます。

サーブレットおよび JSP の移行

JRun 3.0 のサーブレット EJB クライアントは、セッション スレッドの保守を行うために `user.begin` および `user.end` メソッド を呼び出す必要がありまし た。JRun 3.1 ではこれらの呼び出しは不要になりました。詳細については、25 ページの「サーブレット クライアントの変更」を参照してください。

カスタム タグ ハンドラの移行

JRun 3.1 では、タグ ハンドラをキャッシュすることによってカスタム タグのパフォーマンスが向上しました。ただし、このことはインスタンス変数が自動的に初期ステートにリセットされないことを意味します(JRun 3.0 ではカスタム タグはキャッシュされなかったので、JRun がカスタム タグを読み込むたびにインスタンス変数は自動的にリセットされていました)。

タグ ハンドラでインスタンスレベルの変数を保持するか、またはその他のステート管理を行う 場合は、タグ インスタンスがキャッシュされる前に release メソッドを実装して変数をリセットする必要があります。たとえば、タグ ハンドラによって counter という名前の変数を保持する場合は、次の例のように release メソッド内で再初期化する必要があります。

```
...
public class MyHandler extends TagSupport {
  private int counter = 0;
...
public int doAfterBody() throws JspException {
  // 反復の記録
  counter++;
  ...
  }
  public void release() {
    counter = 0;
  }
...
```

詳細については、javax.servlet.jsp.tagext.Tag、javax.servlet.jsp.tagext.TagSupport、および javax.servlet.jsp.tagext.TagSupport に関する JavaDocs を参照してください。

Bean プロパティの公開記述子への移行

EJB バージョン 1.1 仕様書では、Bean 情報とその他の Bean 関連属性を伝える公開記述子の使用が必須になっています。JRun 3.0 では基本的なシナリオについて公開記述子がサポートされていましたが、完全にはサポートされていませんでした。JRun 3.1 では Bean 公開記述子が完全にサポートされているので、EJB および Bean 属性を定義するための方法として使用することをお勧めします。J2EE 準拠アプリケーションを開発するには、EJB で Bean プロパティ ファイルの代わりに公開記述子を使用する必要があります。

メモ

`ejb-name`、`ejb-ref`、および `res-ref` 要素は、JRun 3.0 ではサポートされていませんでしたが、JRun 3.1 ではサポートされています。

Bean 公開記述子について知識を得るには、まず EJB 1.1 仕様書をお読みください。その後、JRun サンプルについて JRun 3.0 の bean.properties ファイルを JRun 3.1 の公開記述子と比較します。次の例は、サンプル 2a の JRun 3.0 プロパティ ファイルを示します。

```
ejb.homeInterfaceClassName=ejbeans.BalanceHome
ejb.remoteInterfaceClassName=ejbeans.Balance
ejb.enterpriseBeanClassName=ejbeans.BalanceBean
ejb.primaryKeyClassName=java.lang.Integer
ejb.beanHomeName=sample2a.BalanceHome

create.ejb.allowedIdentities=all
getValue.ejb.allowedIdentities=all
save.ejb.allowedIdentities=saver
spend.ejb.allowedIdentities=spender

ejipt.isCreateSilent=true
```

次の表は、対応する公開記述子要素の一覧です。

Bean プロパティ ファイル	公開記述子
ejb.homeInterfaceClassName	home 要素
ejb.remoteInterfaceClassName	remote 要素
ejb.enterpriseBeanClassName	ejb-class 要素
ejb.primaryKeyClassName	prim-key-class 要素
ejb.beanHomeName	ejb-name 要素
create.ejb.allowedIdentities	method-permission サブ要素
getValue.ejb.allowedIdentities	method-permission サブ要素
save.ejb.allowedIdentities	method-permission サブ要素
spend.ejb.allowedIdentities	method-permission サブ要素
ejipt.isCreateSilent	JRun 3.1 では使用されません。

さらに公開記述子には、以前 deploy.properties 内に保持されていたセキュリティ ロール情報を指定します。default.properties からのプロパティ (ejipt.maxContexts など) も指定できます。

次の例は、サンプル 2a の JRun 3.1 の公開記述子全体を示します。

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
        EnterpriseJavaBeans 1.1//EN" "http://java.sun.com/j2ee/dtds/
        ejb-jar_1_1.dtd">
<ejb-jar>
  <description>Default Authentication</description>
  <display-name>Default Authentication</display-name>
  <enterprise-beans>
    <entity>
      <display-name>BalanceBean</display-name>
      <ejb-name>sample2a.BalanceHome</ejb-name>
      <home>ejbeans.BalanceHome</home>
      <remote>ejbeans.Balance</remote>
      <ejb-class>ejbeans.BalanceBean</ejb-class>
      <prim-key-class>java.lang.Integer</prim-key-class>
      <persistence-type>Bean</persistence-type>
      <reentrant>False</reentrant>
      <env-entry>
          <env-entry-name>ejipt.maxContexts</env-entry-name>
          <env-entry-type>java.lang.String</env-entry-type>
          <env-entry-value>unspecified</env-entry-value>
      </env-entry>
    </entity>
  </enterprise-beans>
  <assembly-descriptor>
    <security-role>
        <role-name>spender</role-name>
    </security-role>
    <security-role>
        <role-name>saver</role-name>
    </security-role>
    <security-role>
        <role-name>all</role-name>
    </security-role>
      <method-permission>
         <role-name>spender</role-name>
         <method>
             <ejb-name>sample2a.BalanceHome</ejb-name>
             <method-name>spend</method-name>
         </method>
      </method-permission>
      <method-permission>
         <role-name>saver</role-name>
         <method>
             <ejb-name>sample2a.BalanceHome</ejb-name>
             <method-name>save</method-name>
         </method>
```

```
        </method-permission>
        <method-permission>
           <role-name>all</role-name>
           <method>
               <ejb-name>sample2a.BalanceHome</ejb-name>
               <method-name>create</method-name>
           </method>
        </method-permission>
        <method-permission>
           <role-name>all</role-name>
           <method>
               <ejb-name>sample2a.BalanceHome</ejb-name>
               <method-name>getValue</method-name>
           </method>
        </method-permission>
    </assembly-descriptor>
</ejb-jar>
```

すべての JRun EJB サンプルが公開記述子と連動するように変換されました。特定の EJB 公開記述子のシナリオについては、次のサンプルを参照してください。

- **Bean 管理パーシスタンス**　サンプル 2a
- **コンテナ管理パーシスタンス**　サンプル 3a
- **ステートレス セッション Bean**　サンプル 7a、CalculatorBean
- **ステートフル セッション Bean**　セッション 7a、LoanBean

Allaire Developer's Exchange には、プロパティ ファイルと公開記述子との間の変換作業のほとんどを自動処理する `Transformer` ユーティリティが用意されています。このユーティリティは http://devex.allaire.com/developer/gallery/info.cfm?ID=6244656D-C1CA-11D4-AAB000508B94F380&method=Full で入手可能です。

メモ

`Transformer` ユーティリティを使用して JRun 2.3.3 プロパティ ファイルを Web アプリケーション web.xml ファイルに変換することもできます。

---

deploy.properties の local.properties への移行

deploy.properties ファイルには、1 台の JRun サーバーのセキュリティ情報とその他の EJB 固有の情報が含まれています。このファイルは *JRun サーバーのルート ディレクトリ* /deploy ディレクトリにあります。JRun サーバーごとに 1 つの deploy.properties ファイルがあります。

JRun 3.0 では、deploy.properties ファイルと local.properties ファイルには同じサーバーのプロパティを指定していましたが、これらのファイルを両方とも維持していました。JRun 3.1 では、local.properties ファイル内にすべてのサーバー固有のプロパティを統合できます。EJB Deploy ツールでは、runtime.properties ファイルを作成するとき、local.properties ファイルと deploy.properties ファイルの両方を使用します。

local.properties 内で EJB プロパティを指定するときは、接頭辞 **ejb** を使用します。たとえば、local.properties 内で ejipt.classServer.port プロパティを設定するには、**ejb.**ejipt.classServer.port と指定します。

local.properties 内にすべての EJB プロパティを入れ、deploy.properties ファイルを空にしておくことをお勧めします。ただし、必要に応じて、引き続き deploy.properties ファイル内に EJB プロパティを保持することもできます。JRun Deploy ツールおよび実行時システムは次のようなプロパティにアクセスします(リストの一番上にあるソース内のプロパティは、それより下側のソース内にあるプロパティによって無効になります)。

- Deploy ツールは次の順番でプロパティにアクセスします。
  - ejipt.properties
  - システム環境 (`java -D` コマンドライン引数によって設定)
  - global.properties
  - local.properties
  - deploy.properties
- JRun 実行時システムは次の順番でプロパティにアクセスします。
  - ejipt.properties
  - システム環境
  - global.properties
  - local.properties
  - runtime.properties

メモ

EJB エンジンをスタンドアロン モードで実行する場合は、JRun 実行時システムと同じ順番が使用されます。

データ ソースの移行

以前にデータ ソースを重複して維持していた場合 (JMC 内に 1 つ、deploy.properties 内に 1 つ) は、deploy.properties 内のデータ ソースを削除できます。データ ソースの定義を統合する 場合は、23 ページの「EJB データ ソース検索」の説明に従って EJB コード 内のデータソース アクセスを変更する必要もあります。

JMC 内のデータ ソースを定義する 方法の詳細については、『JRun セットアップ ガイド』を参照してください。

メモ
JRun EJB サンプルの場合は、スタンドアロン モードで EJB エンジンが使用され、deploy.properties ファイル内に source1 データ ソースが定義されています。このデータ ソースは local.properties ファイル内にも定義されていた可能性があります。

EJB クライアントの移行

サイトでの EJB の使用法と移行方法によっては、クライアント コードの変更が必要になる場合があります。変更する可能性のある領域については、次のセクションで説明します。

PortableRemoteObject の縮小

JRun 3.0 (および EJB 仕様書の 1.0 バージョン) では、タイプ変換を使用してリモート参照を適切なタイプに変換しました。次に例を示します。

```
...
BidderHome bidderHome =
  (BidderHome)context.lookup("java:comp/env/ejb/sample5a.BidderHome");
...
```

JRun 3.1 では、EJB 1.1 仕様書で推奨されている `javax.rmi.PortableRemoteObject.narrow` メソッドがサポートされています。次に例を示します。

```
...
Object obj = context.lookup("java:comp/env/ejb/sample5a.BidderHome");
BidderHome bidderHome =
  (BidderHome)javax.rmi.PortableRemoteObject.narrow(obj,
     BidderHome.class);
...
```

JNDI 検索

EJB 検索

JNDI 検索に beanname または java:comp/env/ejb/beanname を使用できるようになりました。EJB 1.1 仕様書により厳密に従うには、java:comp/env/ejb/*beanname* の使用をお勧めします。次に例を示します。

```
...
Object obj = context.lookup("java:comp/env/ejb/sample5a.BidderHome");
BidderHome bidderHome =
  (BidderHome)javax.rmi.PortableRemoteObject.narrow(obj,
     BidderHome.class);
...
```

EJB データ ソース検索

JRun 3.0 では、次の 2 か所にデータ ソースを定義していました。

- JMC (サーブレットおよび JSP によって使用)
- deploy.properties (EJB によって使用)

サーブレットおよび JSP は次のように JNDI 検索を使用してデータ ソースにアクセスしていました。

```
String thisDataSource = “products”;

// 実行するSQLステートメント
String sqlStatement = "select * from products where productid='" +
        thisId + "' and version='" + thisVersion + "'";

// 接続オブジェクト
Connection dbConnection = null;
// ステートメント オブジェクト
Statement dbStatement = null;
// 結果セット オブジェクト
ResultSet dbResultSet = null;
// データベース アクセス コードの開始
try {
  // JNDI InitialContext オブジェクトの定義
  InitialContext ctx = new InitialContext();
  // InitialContext 内でのデータソースの検索
  DataSource ds =
    (DataSource)ctx.lookup("java:comp/env/jdbc/" + thisDataSource);
  dbConnection = ds.getConnection();

  // ステートメント オブジェクトの作成
  dbStatement = dbConnection.createStatement();
  // クエリの実行
  dbResultSet = dbStatement.executeQuery(sqlStatement);
  ...
```

EJB は次のように ResourceManager.getConnection への呼び出しを使用してデータ ソースにアクセスしていました。

```
try {
  Connection connection = ResourceManager.getConnection("source1");
  try {
    final Statement statement = connection.createStatement();
    final ResultSet results =
      statement.executeQuery("SELECT value FROM account WHERE id = " +
        _context.getPrimaryKey());
...
```

JRun 3.1 では、EJB は JNDI 検索を使用して JMC 定義のデータ ソースにアクセスできます。1 つのデータ ソースに対して 2 つの定義を維持する必要はなくなりました。たとえば、EJB は次のコードを使用してデータ ソースにアクセスできるようになりました。

```
try {
  final Context context = new InitialContext();
  final DataSource source =
    (DataSource)context.lookup("java:comp/env/jdbc/source1");
  final Connection connection = source.getConnection();
  final Statement statement = connection.createStatement();
  final ResultSet results =
    statement.executeQuery("SELECT value FROM account WHERE id = " +
      _context.getPrimaryKey());
...
```

詳しい例については、EJB サンプル 2a 内の BalanceBean.java を参照してください。

リモート EJB クライアントのデータ ソース

JRun バージョン 3.1 では、クライアントはリモート データ ソースを検索して使用できます。それには、コンテキスト ファクトリ、プロバイダの URL、ユーザ名、パスワードのすべてを使用して Properties オブジェクトを形成します。その後、InitialContext オブジェクトのインスタンスを作成し、Properties オブジェクトに渡します。次のコードのように、JNDI 検索を実行してリモート データ ソースにアクセスできるようになりました。

```
...
try {
  Properties properties = new Properties();
  // 初期コンテキストでのリモートプロパティの定義
  properties.setProperty(Context.INITIAL_CONTEXT_FACTORY,
    "allaire.ejipt.ContextFactory");
  properties.setProperty(Context.PROVIDER_URL,
    "ejipt://remotehost:2773");

  String user_name = "client";
  String password = "client";
  // ユーザ名およびパスワードの定義 (この例ではハードコード)
  properties.setProperty(Context.SECURITY_PRINCIPAL, user_name);
  properties.setProperty(Context.SECURITY_CREDENTIALS, password);
  // メッセージのデバッグ
```

```
  System.out.println("new InitialContext");
  Context ctx = new InitialContext(properties);
  System.out.println("lookup");
  // リモート データ ソースの検索（この例ではハードコード）
  // Java以外のクライアントは、
  // タイプ変換の代わりに PortableRemoteObject.narrow を使用する必要あり
  DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/DB1");
  System.out.println("getConnection");
  // 接続の確立
  Connection conn = ds.getConnection();
  // SQLステートメントが続く
...
  // 終了メソッドが続く
...
```

クライアントがリモート データ ソース検索を行うには、クラスパス内に jrun.jar が必要です。クラスパス内に jrun.jar がない場合は、`ClassNotFoundException` が返されます。

メモ
リモート クライアント が JRun データ ソースにアクセスするには、JRun サーバー内で *datasourcename*`.ejipt.remotelyAccessible` プロパティが true に設定されている必要があります。local.properties と deploy.properties のどちらのファイル内でコーディングするかによって、このプロパティの構文は異なります。詳細については、38 ページの「新しいプロパティ」を参照してください。

サーブレット クライアントの変更

JRun 3.1 では、サーブレットと EJB 間の通信方法に次の変更が行われました。

- **スレッド保守の強化**　JRun バージョン 3.0 では、サーブレット クライアントは `UserSession.begin` および `UserSession.end` メソッドへの呼び出しをコーディングすることによってプログラムによるスレッド保守を行っていました。JRun バージョン 3.1 では、EJB コンテキストを `allaire.jrun.ejbContext` セッション変数および JRun メッセージ スレッド内に自動的に保存します。セッションが終了したら、`session.invalidate` メソッドを呼び出します。その例については、EJB サンプル 9a を参照してください。
- **シングル サインオン**　JRun バージョン 3.1 の統合セキュリティ フレームワークを利用できます。サーブレットおよび JSP によって呼び出されると、EJB は Web アプリケーション認証を利用できます。詳細については、26 ページの「シングル サインオンの使用」を参照してください。シングル サインオンを使用すると、EJB コンテキスト は自動的に保存されます。`allaire.jrun.ejbContext` セッション変数を管理する必要はありません。その例については、EJB サンプル 9b を参照してください。

JRun バージョン 3.1 でクライアントをコーディングする方法の詳細については、『JRun によるアプリケーションの開発』の「EJB クライアントのコーディング」を参照してください。

シングル サインオンの使用

サーブレット および EJB の仕様書には、認証の条件 (つまり 、ロールベースのセキュリティによるアクセス コントロール) の概要が記載されていますが、ベンダ固有の認証メカニズムも認められています。JRun バージョン 3.0 には、サーブレット /JSP および EJB についてさまざまな認証メカニズムが用意されていましたが、重複するユーザ / ロールの定義または各メカニズムへのカスタマイズされた拡張子を維持する必要がありました。JRun バージョン 3.1 は、シングル サインオンと呼ばれる統合された認証メカニズムを特長としています。

J2EE セキュリティの条件および用語の詳細については、『Java 2 Platform Enterprise Edition Specification, v1.2』を参照してください。補足情報については、サーブレット 2.2 の仕様書および EJB 1.1 の仕様書を参照してください。

メモ

この説明を読む前に、Web アプリケーションの認証についての知識を持っている必要があります。詳細については、『JRun によるアプリケーションの開発』の「Web アプリ ケーション認証」を参照してください。また、EJB 認証についての知識を持っている必要があります。詳細については、JRun EJB サンプルを参照してください。

セキュリティ関連の移行オプション

JRun バージョン 3.1 のセキュリティ アーキテクチャにはほとんど下位互換性があります。そのため、現在の認証方法を維持することも、現在のアーキテクチャに移行することもできます。

JRun バージョン 3.1 の既定の Web および EJB 認証仕様には、シングル サインオン機能が実装されています。この機能を使用するには、次のタスクを実行します。

- JRun バージョン 3.1 とともに配布された local.properties ファイルを使用します。
- `allaire.jrun.security.PropertyFileAuthentication` クラス (Windows) または `jrunpasswd` ユーティリティ (UNIX) を使用して deploy.properties ファイル内に以前に定義されていたユーザを users.properties ファイルに移動します。これらのツールによって、ユーザおよび暗号化されたパスワードが users.properties に追加されます。詳細については、『JRun によるアプリケーションの開発』を参照してください。

EJB 認証またはサーブレット /JSP 認証をカスタマイズしていた場合は、ユーザ ストアの実行時保守の必要性を検討し、必要に応じて実行時シングル サインオンを実装します。実行時シングル サインオンについては、35 ページの「動的なユーザの追加および削除」を参照してください。

JRun 3.0 の認証メカニズム

JRun 3.1 シングル サインオンによる影響を完全に理解するには、JRun 3.0 で提供されていた 2 つの部分からなる認証メカニズムを理解する必要があります。次の説明では、これらの異なる 2 つのメカニズムの概要について説明します。

- **サーブレット および JSP 認証** サーブレット および JSP の場合、JRun では web.xml ファイルに行われた指定と users.properties ファイル内に格納されている既定のユーザおよびパスワードを使用した Web 認証がサポートされていました。開発者はこのメカニズムをカスタマイズして、サイト固有のユーザ ストアを実装できました。
- **EJB 認証** EJB の場合、JRun EJB エンジンは ejb-jar.xml ファイルまたは Bean プロパティ ファイルを使用してセキュリティの条件にアクセスし、deploy.properties ファイルを使用してユーザ、パスワード 、およびロール割り 当てを格納しました。開発者は UserBean、RoleBean、および LoginSessionBean EJB を拡張することによってこのメカニズムをカスタマイズできました。

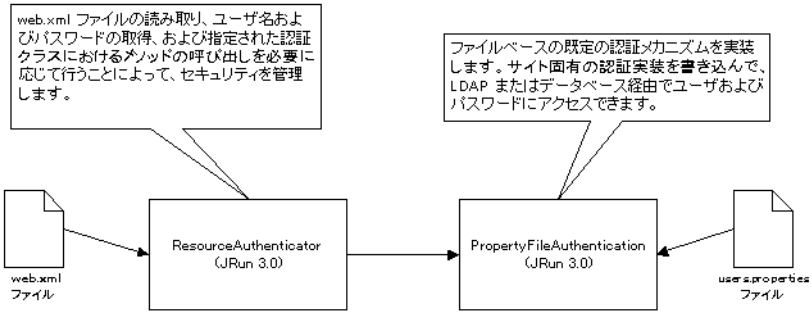
JRun 3.0 でのサーブレットおよび JSP 認証

次の表は、JRun 3.0 サーブレット エンジンの認証およびロールベースのセキュリティの概要を示します。

アクティビティ	メカニズム	コメント
保護されたリソースの指定	web.xml ファイル内の url-pattern 要素	
保護されたリソースにアクセスできるロールの定義	web.xml ファイル内の role-name 要素	
ユーザのロールへの割り当て	users.properties ファイル内の role 接頭辞	既定のメカニズム。書き換え可能。
実行時のユーザ名およびパスワードの指定	web.xml ファイル内の auth-method 要素がログイン フォームの表示方法を制御します。	auth-method に BASIC を指定すると、Web ブラウザからユーザ名とパスワード の入力が要求されます。auth-method に FORM を指定する場合は、ログイン フォームを作成する必要があります。
ユーザ名およびパスワードのレポジトリ	users.properties ファイル内の user 接頭辞	既定のメカニズム。書き換え可能。

アクティビティ	メカニズム	コメント
認証ハンドラ	ResourceAuthenticator クラス	authentication サービスに関連付けられています。 書き換え不可。 認証実装クラス内のメソッドを呼び出します。
既定の認証実装	PropertyFileAuthentication クラス	authentication サービスに関連付けられています。 書き換え可能。

次の図は、JRun 3.0 サーブレット エンジン内での既定の認証を示します。



次の一覧は、JRun 3.0 サーブレットおよび JSP セキュリティ コンポーネントの詳しい説明を示します。

- web.xml login-config 要素には、ユーザ名およびパスワードが読み込まれる方法を指定するサブ要素が含まれています。security-constraint 要素には、保護するリソースと、これらのリソースにアクセスできるロールを指定するサブ要素が含まれています。次の web.xml スニペットはこれらの要素を示します。

```
...
<login-config>
 <auth-method>FORM</auth-method>
 <form-login-config>
  <form-login-page>/loginpage.htm</form-login-page>
  <form-error-page>/loginerror.htm</form-error-page>
 </form-login-config>
</login-config>
...
<security-constraint>
 <web-resource-collection>
  <web-resource-name>Financial Reports</web-resource-name>
  <url-pattern>/financials/*</url-pattern>
 </web-resource-collection>
 <auth-constraint>
  <role-name>manager</role-name>
 </auth-constraint>
</security-constraint>
...
```

- global.properties　このファイルにより、認証ハンドラ (ResourceAuthenticator) および認証実装クラス (既定は PropertyFileAuthentication) が確立されます。次のスニペットはこれらのプロパティを示します。

```
...
webapp.ResourceAuthenticator=authentication
...
authentication.class=allaire.jrun.servlet.ResourceAuthenticator
...
authentication.service=propfile
authentication.propfile.class=allaire.jrun.security.PropertyFileAuth
     entication
authentication.propfile.filename={jrun.rootdir}/lib/users.properties
...
```

- users.properties　このファイルは、ユーザ、グループ、およびロールを定義するために PropertyFileAuthentication (既定の認証実装クラス) によって使用されます。次のスニペットはこれらのプロパティを示します。

```
...
group.all=*
role.user=all
role.users=all
user.randyn=raNztUnCaeccI
user.clare=clZCPw38r6yH2
...
```

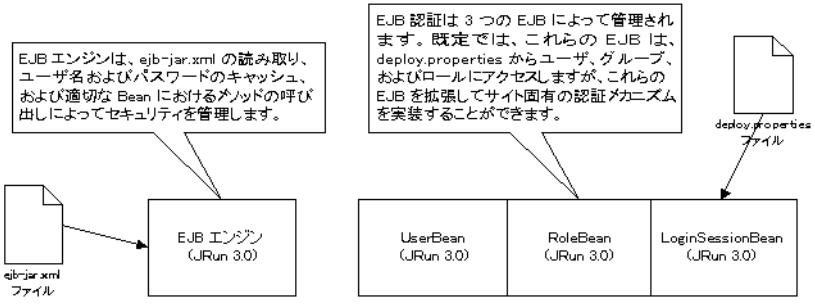
- ResourceAuthenticator　これは認証ハンドラです。このハンドラはセキュリティ関連のアクションを調整します。web.xml ファイルから設定を読み込み、PropertyFileAuthentication (またはサイト固有の認証実装クラス) 内のメソッドを呼び出してユーザを認証します。
- PropertyFileAuthentication　これは既定の認証実装クラスであり、テキスト ファイル (既定のファイルは users.properties) に対してユーザを認証するために ResourceAuthenticator によって呼び出されます。このクラスには、ResourceAuthenticator によって呼び出されるメソッドを定義する AuthenticationInterface インターフェイスが実装されています。また、関連付けられているテキスト ファイルをプログラムによって更新できる AuthenticationManager インターフェイスも実装されています。

JRun 3.0 での EJB 認証

次の表は、JRun 3.0 EJB エンジンでの認証およびロールベースのセキュリティの概要を示します。

アスペクト	メカニズム	コメント
保護方法の指定	Bean プロパティ ファイル内の allowedIdentities プロパティまたは ejb-jar.xml ファイル内の method-permission 要素	
保護されたリソースにアクセスできるロールの定義	Bean プロパティ ファイル内の allowedIdentities プロパティまたは ejb-jar.xml ファイル内の role-name 要素	
ユーザのロールへの割り当て	deploy.properties ファイル内の ejipt.roles プロパティ	既定のメカニズム。書き換え可能。
実行時のユーザ名およびパスワードの指定	Properties オブジェクト内のユーザ (SECURITY_PRINCIPAL) およびパスワード (SECURITY_CREDENTIALS) を InitialContext コンストラクタに渡します。	
ユーザ名およびパスワードのレポジトリ	deploy.properties ファイル内の ejipt.users プロパティ	既定のメカニズム。書き換え可能。
認証ハンドラ	UserManager	認証実装 Bean 内のメソッドを呼び出します。書き換え不可。
既定の認証実装	LoginSessionBean、UserBean、および RoleBean	deploy.properties ファイル内のプロパティによって確立されます。書き換え可能。

次の図は、JRun 3.0 EJB エンジン内での既定の認証を示します。



次の一覧は、JRun 3.0 EJB セキュリティ コンポーネントの詳しい説明を示します。

- ejb-jar.xml (または Bean プロパティ) ファイル　security-role および method-permission 要素にはロールと、ロールがアクセス可能なメソッドを指定します。次の ejb-jar.xml スニペットはこれらの要素を示します。

```
...
<assembly-descriptor>
  <security-role>
    <role-name>spender</role-name>
  </security-role>
  <security-role>
    <role-name>saver</role-name>
  </security-role>
  <security-role>
    <role-name>all</role-name>
  </security-role>
  <method-permission>
    <role-name>spender</role-name>
    <method>
      <ejb-name>sample1a.BalanceHome</ejb-name>
      <method-name>spend</method-name>
    </method>
  </method-permission>
...
```

- deploy.properties　UserBean および RoleBean は、このファイル内のプロパティによって定義されたユーザおよびロールにアクセスします。次のスニペット はこれらのプロパティを示します。

```
...
ejipt.userHomeName=default.UserHome
ejipt.roleHomeName=default.RoleHome
ejipt.loginSessionHomeName=default.LoginSessionHome
...
ejipt.users=spender1:pass;spender2:pass;saver1:pass;saver2:pass;
       chief:pass
ejipt.roles=spender:spender1,spender2,chief;saver:saver1,saver2,
       chief
...
```

- LoginSessionBean　このステートフル セッション Bean は、UserBean.checkPassword メソッドを呼び出すことによってパスワードを確認します。パスワードが有効な場合、LoginSessionBean は UserManager.login メソッドを呼び出します。
- UserBean　このエンティティ Bean には、LoginSessionBean によって呼び出される checkPassword などのユーザ保守メソッドが指定されます。この EJB を使用して、UserHome インターフェイス内で定義される create メソッドによって新しいユーザを作成することもできます。

- RoleBean　このエンティティ Bean には、addMember や isMember などのロール保守メソッドが指定されます。この EJB を使用して、RoleHome インターフェイス内で定義される create メソッドによって新しいロールを作成することもできます。
- UserManager　このクラスには、ユーザおよびロールを管理する機能が用意されています。UserManager を拡張して、サイト固有のユーザおよびロール ストアを実装できます。拡張方法の詳細については、文書のホーム ページから入手可能な UserManager JavaDocs を参照してください。

JRun 3.1 のシングル サインオン

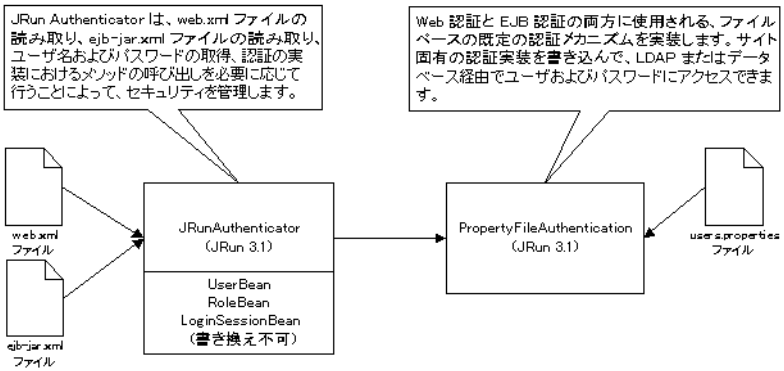
JRun 3.1 の シングル サインオン機能を利用すると、以前はサーブレットと JSP でしか使用できなかった認証実装クラスを EJB コンポーネント でも 使用できます。既定では、このことは user.properties ファイル内で EJB ユーザを定義することを意味します。JRun バージョン 3.0 の Web アプリケーション認証に対して行った拡張は、JRun バージョン 3.1 EJB 認証に自動的に適用されます。

次の表は、JRun 3.1 での認証およびロールベースのセキュリティの概要を示します。

アクティビティ	メカニズム	コメント
保護されたリソースの指定	web.xml ファイル内の url-pattern 要素	
保護方法の指定	ejb-jar.xml ファイル内の method-permission 要素	
保護されたリソースにアクセスできるロールの定義	web.xml ファイル内の role-name 要素または ejb-jar.xml ファイル内の role-name 要素	
ユーザのロールへの割り当て	users.properties ファイル内の role 接頭辞	既定のメカニズム。書き換え可能。
実行時のユーザ名およびパスワードの指定	web.xml ファイル内の auth-method 要素がログイン フォームの表示方法を制御します。	auth-method に BASIC を指定すると、Web ブラウザからユーザ名とパスワード の入力が要求されます。auth-method に FORM を指定する場合は、ログイン フォームを作成する必要があります。サーブレット および JSP ページはユーザ名およびパスワードを EJB エンジンに渡す必要はありません。
ユーザ名およびパスワードのレポジトリ	users.properties ファイル内のuser 接頭辞	既定のメカニズム。書き換え可能。

アクティビティ	メカニズム	コメント
認証ハンドラ	JRunAuthenticator クラス	jrunauth サービスに関連付けられています。書き換え不可。 認証実装クラス内のメソッドを呼び出します。
既定の認証実装	PropertyFileAuthentication クラス	jrunauth サービスに関連付けられています。 サーブレットおよび EJB セキュリティの両方に使用されます。 書き換え可能。

次の図は JRun 3.1 での既定の認証を示します。



次の一覧は、JRun 3.1 サーブレットおよび JSP セキュリティ コンポーネントの詳しい説明を示します。

- web.xml　`login-config` 要素には、ユーザ名およびパスワードが読み込まれる方法を指定するサブ要素が含まれています。`security-constraint` 要素には、保護するリソースと、これらのリソースにアクセスできるロールを指定するサブ要素が含まれています。これらの要素の例については、27 ページの「JRun 3.0 でのサーブレットおよび JSP 認証」を参照してください。
- ejb-jar.xml　`security-role` および `method-permission` 要素にはロールと、ロールがアクセス可能なメソッドを指定します。これらの要素の例については、30 ページの「JRun 3.0 での EJB 認証」を参照してください。

- global.properties　このファイルにより、認証ハンドラ (JRunAuthenticator) および認証実装クラス (既定は PropertyFileAuthentication) が確立されます。次のスニペットはこれらのプロパティを示します。

```
...
webapp.ResourceAuthenticator=authentication
...
jrunauth.class=allaire.jrun.servlet.JRunAuthenticator
...
jrunauth.service=propfile
jrunauth.propfile.class=allaire.jrun.security.PropertyFileAuthentica
     tion
jrunauth.propfile.filename={jrun.rootdir}/lib/users.properties
...
```

- user.properties　このファイルは、ユーザ、グループ、およびロールを定義するために PropertyFileAuthentication (既定の認証実装クラス) によって使用されます。次のスニペットはこれらのプロパティを示します。

```
...
group.all=*
role.user=all
role.users=all
role.spender=spender1,spender2,chief
role.saver=saver1,saver2,chief
user.randyn=raNztUnCaeccI
user.clare=clZCPw38r6yH2
user.spender1=chpP1JdC4Xpqg
user.spender2=chpP1JdC4Xpqg
user.saver1=chpP1JdC4Xpqg
user.saver2=chpP1JdC4Xpqg
user.chief=chpP1JdC4Xpqg
...
```

- JRunAuthenticator　これは認証ハンドラです。このハンドラはセキュリティ関連のアクションを調整します。JRunAuthenticator は web.xml および ejb-jar.xml ファイルから設定を読み込み、PropertyFileAuthentication (またはサイト固有の認証実装クラス) 内のメソッドを呼び出してユーザを認証します。
  JRunAuthenticator は UserBean、RoleBean、および LoginSessinBean も使用しますが、拡張はできません。

- PropertyFileAuthentication　これは既定の認証実装クラスであり、テキスト ファイル ベースのユーザ レポジトリ (既定のファイルは users.properties) に対してユーザを認証するために JRunAuthenticator によって呼び出されます。このクラスには JRunAuthenticator によって呼び出されるメソッドを定義する AuthenticationInterface インターフェイスが実装されています。また、ユーザ レポジトリをプログラムによって更新できる AuthenticationManager インターフェイスも実装されています。

EJB エンジンでは UserManager と UserBean、RoleBean、および LoginSessionBean EJB も使用されます。これらの違いは次のとおりです。

- EJB は allaire.ejipt.ejbeans パッケージではなく allaire.jrun.ejbeans パッケージ内にあります。
- allaire.jrun.ejbeans パッケージ内で EJB を拡張することはできません。
- UserBean は認証実装クラス (既定では PropertyFileAuthentication) を介してユーザにアクセスすることによって UserManager ユーザ リストを初期化します。
- スタンドアロン EJB エンジンは allaire.ejipt.ejbeans パッケージ内の EJB を使用します。

カスタム認証実装の作成

サーブレットまたは JSP 認証をカスタマイズしていた場合は、その認証が EJB と連動する必要があります。UserBean、RoleBean、および LoginSessionBean を使用して EJB スタンドアロン認証をカスタマイズした場合は、カスタマイズした EJB スタンドアロン認証ロジックを JRun 認証フレームワーク内に実装する必要があります。このトピックの情報については、『JRun によるアプリケーションの開発』と http://www.allaire.com/handlers/index.cfm?ID=17555&Method=Full にある JRun 開発者センター (JRun DevCenter) を参照してください。

動的なユーザの追加および削除

JRun バージョン 3.0 では、ユーザをユーザ ストアに追加した場合、新しいユーザを認識させるために JRun を再起動する必要がありました。JRun バージョン 3.1 ではユーザの追加および削除を動的に行うことができます。この機能は実行時シングル サインオンとも呼ばれています。

メモ
この機能は、JRun 3.1 の既定である PropertyFileAuthentication 認証実装クラスに含まれています。

実行時シングル サインオンは EjiptAuthenticationManager クラスによって実装されます。カスタマイズした認証実装クラス内でこの機能を有効にするには、次の手順を実行してください。

1 認証実装クラス内に allaire.jrun.security.AuthenticationManagerクラス を実装します。

```
public class MyAuthentication implements AuthenticationInterface,
     AuthenticationManager {
...
```

2 EjiptAuthenticationManager および ServiceContext の変数を追加します。

```
EjiptAuthenticationManager ejiptAuth = null;
ServiceContext serviceContext;
```

3 ServiceContext に渡される init メソッドをコーディングします。

```
public void init(ServiceContext serviceContext)
```

4 init メソッドでは、渡された ServiceContext を使用して EjiptAuthenticationManager のインスタンスを作成します。

```
ejiptAuth = new EjiptAuthenticationManager(serviceContext);
```

5 次のように AuthenticationManager インターフェイスからメソッドを認証実装クラスに追加します。

- boolean isUser(String user)
- boolean isUserInGroup(String user, String role)
- boolean isUserInRole(String user, String group)
- boolean addUser(String user, String password) throws IOException
- boolean addUser(String user, String password, String[] group, String[] role) throws IOException
- boolean changePassword(String user, String oldPassword, String newPassword) throws IOException
- boolean removeUser(String user) throws IOException
- Enumeration getUserList()
- Enumeration getGroupList()
- Enumeration getGroupMembers(Object group)
- Enumeration getRoleList()
- Enumeration getRoleMembers(Object role)
- boolean addGroup(String group, String[] users, String[] roles) throws IOException
- boolean addToGroup(String user, String[] group) throws IOException
- boolean removeFromGroup(String user, String[] group) throws IOException
- boolean removeGroup(String group) throws IOException
- boolean addRole(String role, String[] groups, String[] users) throws IOException
- boolean addToRole(String user, String[] role) throws IOException
- boolean removeFromRole(String user, String[] role) throws IOException
- boolean removeRole(String role) throws IOException

6 追加および削除の接頭辞が付いたメソッドを実行し、該当するユーザ ストアにユーザ情報を追加および削除するときに必要な手順を行います。各 add* および remove* メソッド内では、JRun 実行時システム上でアクションを実行する EjiptAuthenticationManager 内の同等のメソッドを呼び出します。

7 追加の AuthenticationManager メソッドを実行します (オプション)。

8 詳細については、文書のホーム ページから入手可能な AuthenticationManager と EjiptAuthenticationManager JavaDocs を参照してください。

その他の機能強化

JRun 3.1 では、次のような多数の小規模な機能強化が行われています。

新しくサポートされた公開記述子の要素

JRun EJB エンジンでは、`ejb-name`、`ejb-ref`、および `res-ref` 要素が完全にサポートされるようになりました。これらの要素の詳細については、EJB バージョン 1.1 の仕様書を参照してください。

Deploy ツールの強化

JRun 3.1 では、Deploy ツールが次のように強化されています。

- **jrun_exports.jar および jrun_objects.jar** Deploy ツールによって jrun_exports.jar および jrun_objects.jar を検索できるようになりました。ファイルが見つからなかった場合は、自動的に default_exports.jar および default_objects.jar の使用に戻ります。default_exports.jar および default_objects.jar が見つからなかった場合は、以前のように警告がログに記録されます。
- **-keepgenerated フラグ** Deploy ツールに `-keepgenerated` フラグを使用するようになりました。生成されたソース、スタブ、およびコンパイルされたソースおよびスタブは公開ディレクトリの src および lib ディレクトリ内に維持されます。この機能はスタブ生成での問題を診断するときに役立ちます。

EarDeploy の強化

EJB jar ファイルを `ejipt.ejbJars` リストに追加するために `EarDeploy` ユーティリティが変更されました。

名前が同じで署名が異なるメソッドのセキュリティ

名前が同じで署名が異なる 2 つのメソッドを保護する場合もセキュリティが機能するようになりました。

その他の機能強化と修正

その他の機能強化、修正、および変更には次のものがあります。

- 起動時にデータソースがオフラインになっている場合でも、EJB エンジンが正常に起動するようになりました。
- ステートレス セッション Bean が既定で再入可能になりました。

プロパティ ファイルの変更

JRun バージョン 3.1 には多数の新しいプロパティが含まれています。さらに、既定の global.properties ファイルおよび local.properties ファイルが変更されました。

新しいプロパティ

次の表は JRun バージョン 3.1 の新しいプロパティの一覧です。

プロパティ	説明
iiop.tsn.path	tnameserv.exe へのパス。既定ではシステム パスを使用します。
iiop.ORBPort	ORB ポート。既定値は 900 です。
iiop.class	RMI/IIOP サポートを実装するクラス。既定では allaire.jrun.iiop.IIOPBindingService が使用され、通常は変更しません。
ejb.ejipt.*property*	以前は deploy.properties 内にあった EJB 設定を指定します。 個々のプロパティの詳細については、文書のホームページから入手可能な EjiptProperties JavaDocs を参照してください。
ejb.jdbcService.name	JDBC データ ソースを定義するサービスの名前
ejipt.libJars	Deploy ツールによって使用される JAR ファイルの一覧。既定では、jrun_ejbeans.jar および ejipt_ejbeans.jar が使用されます。
ejipt.disableOnePhase	2 フェーズ コミットまたは 1 フェーズ コミットのどちらを使用するかを示します。 true または false を指定します。既定値は false で、1 フェーズ コミットを有効にします。
ejipt.isAlwaysDirty	EJB エンジンが常に ejbStore を呼び出すかどうかを示します。 true または false を示します。既定値は false です。
ejipt.sourceInitSQL	新しい接続が接続プールに追加されるたびに実行される SQL ステートメント
ejipt.processInterval	エンティティ Bean が Passive になるまでの待機時間（秒）を指定します。既定値は 60 秒です。
datasourcename.ejipt.remotely Accessible (deploy.properties) jdbc.*datasourcename*.remotely Accessible (local.properties)	リモート クライアントがデータ ソースにアクセスできるかどうかを示します。既定値は false です。 詳細については、24 ページの「リモート EJB クライアントのデータ ソース」を参照してください。

プロパティ	説明
java.args.debug	デバッグ用の仮想マシン引数
jdbc.*datasourcename*.minSize	接続プールの最小サイズ。既定値は 0 です。
jdbc.*datasourcename*.maxSize	接続プールの最大サイズ。既定値は -1 または制限なしです。
jdbc.*datasourcename*.initSize	接続プールの初期サイズ。既定値は最小サイズです。
jdbc.*datasourcename*.maxUse	接続が閉じられる前に使用できる最大回数。既定値は -1 または制限なしです。この回数を超えて接続を使用すると、接続が閉じられ、プールから削除されます。
jdbc.*datasourcename*.maxWait	プール内の接続がすべて使用中の場合 (プールのサイズは maxSize)、接続が使用可能になるまでの最大時間 (単位は秒)。既定値は 5 秒です。maxWait が期限切れになった後でも接続が使用可能にならない場合は、SQLException が返されます。
jrunauth.*propertyname*	JRun 3.1 のシングル サインオン サポートの一部として使用されます。
jsp.jst_suffixes	JSP で作成されたカスタム タグに使用できるファイル接尾辞を指定します。
ssl.*propertyname*	JRun 3.1 JWS の SSL サポートを実装するときに使用します。
timing.*propertyname*	jrunstats、jsp、および jst の既定のタイミング設定を指定します。
webapp.hotdeploy.enabled	動的公開を有効または無効にします。 Web アプリケーションの動的公開の詳細については、3 ページの「Web アプリケーションの動的な公開および更新」を参照してください。
webapp.hotdeploy.interval	JRun 上での変更の有無を確認する頻度を秒単位で指定します。既定値は 5 秒です。 この値をゼロに設定すると、動的公開サービスは無効になります。
webapp.hotdeploy.onchange	JRun によって監視されるディレクトリおよびファイルを指定します。 これらのファイルのいずれかが変更されると、Web アプリケーションが自動的に再起動され、「hot deploy initiating」というメッセージがログ ファイルに書き込まれます。

プロパティ	説明
webapp.hotdeploy.defaultname	既定のコンテキスト（/ の URL）にマッピングされる必要がある WAR ファイルの名前を指定します。既定値は default.war です。 ほとんどの場合、Web アプリケーションではコンテキストとして */webappname* を使用しますが、既定のアプリケーションではスラッシュだけを使用します。
webapp.hotdeploy.autodeploy	指定された場所にある新しい WAR ファイルが自動的に公開される場所を指定します。 サーバーの起動時にこの場所が確認され、サーバーの実行中は新しいファイルの監視も行われます。

既定の global.properties ファイルへの変更

global.properties ファイル内の更新された設定

jrun.classpath プロパティには {jrun.updates.classpath} および JDK の lib/tools.jar ファイルが含まれています。

jrun.services プロパティ内では、authentication サービスではなく jrunauth サービスが既定で起動されるようになりました。JRun 3.0 で使用されていた authentication サービスの設定は、既定でコメント化されています。jrunauth サービスの詳細については、32 ページの「JRun 3.1 のシングル サインオン」を参照してください。

jsp.jikes.compiler プロパティに -g フラグを指定するようになりました。

webapp.classpath プロパティに /WEB-INF/jsp を指定しなくなりました。

global.properties 内の新しいプロパティ

global.properties ファイルには次の新しいプロパティが含まれています。

- java.args.debug
- jrunauth 接頭辞付きプロパティ
- ssl 接頭辞付きプロパティ
- jsp.jst_suffixes
- iiop.class
- timing 接頭辞付きプロパティ
- ejb.jdbcService.name
- ejb.ejipt 接頭辞付きプロパティ

新しいプロパティの説明については、38 ページの「新しいプロパティ」を参照してください。

既定の local.properties ファイルへの変更

local.properties ファイル内の更新された設定

servlet.services プロパティに jcp を指定しなくなりました。

余分な仮想マシン引数のプロパティは削除されました。

local.properties 内の新しいプロパティ

local.properties ファイルには次の新しいプロパティが含まれています。

- java.args.debug
- iiop.tsn.path
- iiop.tsn.ORBPort
- ejb.ejipt.enableMessaging
- ejb.ejipt.classServer.host
- ejb.ejipt.classServer.port
- ejb.ejipt.homePort

新しいプロパティの説明については、38 ページの「新しいプロパティ」を参照してください。

索引

J

K

L

M

P

R

S

T

U

W