JRun タグ ライブラリ リファレンス

Windows®、UNIX、および
Linux 用 JRun 3.1

Allaire® Corporation

版権告知

部品番号 : AA-JJTAG-RK

目次

本書の概要

『JRun タグ ライブラリ リファレンス』は、Web アプリケーションで JRun JSP カスタム タグを使用する Java 開発者を対象としています。

目次

対象読者

『JRun タグ ライブラリ リファレンス』は、Web アプリケーションで JRun JSP カスタム タグを使用する Java 開発者を対象としています。

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 に関する詳細な製品情報および関連トピック	http://cfusion.sirius.co.jp/jrun/
テクニカル サポート	Allaire 社の提供するプロフェッショナル サポート プログラム	http://www.allaire.com/support/
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/jrun/
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、あらゆる関係者をサポートできるように設計されています。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、以下の文書が含まれます。

文書	説明
『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 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 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 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 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 セットアップ ガイド』	JRun 管理コンソール (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 タグ ライブラリ クイック リファレンス カード』	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。
『JSP クイック リファレンス カード』	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。
『Allaire ClusterCATS の使用』	マルチサーバーの負荷管理およびサーバー障害の保護を行う ClusterCATS の使用方法に関する情報が記載されています。
『JRun Studio 入門』	JRun Studio を使用した Web コンテンツの構築、テスト、および公開方法について説明します。さまざまなスクリプトおよびマークアップ言語用の組み込みエディタの使用方法についても説明します。

オンライン文書の表示

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。HTML 文書を表示するには、JRun を実行している Web 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 は、JRun アプリ ケーション を作成するためのビジュアル開発ツールです。JRun Studio には直観的に操作できる GUI インターフェイスがあり、アプリケーションの構築に必要なツールが利用できます。また、JRun Studio によって、任意の JDBC データベースのデータを選択、挿入、更新、または削除を行う複雑な SQL ステートメントを作成できます。また、HTTP を通じてリモート サーバー上のデータベースに接続することもできます。これは複雑なネットワーク設定を必要としません。

JRun Studio は JRun Developer、JRun Professional、JRun Enterprise とは別売になっています。

JRun 文書の印刷

印刷版の文書を読むには、製品とともにインストールされた Adobe Acrobat PDF ファイルを探します。PDF ファイルからは、優れた印刷出力を得ることができます。文書の全体または一部を印刷できます。

その他の情報リソース

本書で扱っているトピックの詳細については、以下のリソースも参照してください。

書籍

- 『Java Servlets』Karl Moss 著、McGraw Hill 刊、1999 年、ISBN: 0071351884
- 『Java Servlets: By Example』Alan R. Williamson 著、Manning Publications 刊、1998 年、ISBN: 188477766X
- 『Java Servlet Programming』Jason Hunter、William Crawford 共著、O'Reilly & Associates 刊、1998 年、ISBN: 156592391X
- 『Developing Java Servlets』James Goodwill 著、Sams 刊、1999 年、ISBN: 0672316005
- 『Inside Servlets: Server-Side Programming for the Java Platform』Dustin R. Callaway 著、Addison-Wesley Pub.Co. 刊、1999 年、ISBN: 0201379635
- 『Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition』Ed Roman 著、Wiley 刊、ISBN: 0471332291
- 『Enterprise JavaBeans』Richard Monson-Haefel 著、O'Reilly & Associates 刊、ISBN: 1565928695
- 『Enterprise Javabeans: Developing Component-Based Distributed Applications』Thomas C. Valesky 著、Addison Wesley Publishing Company 刊、ISBN: 0201604469

オンライン リソース

- Java servlet API (http://java.sun.com/products/servlet)
- JavaServer Pages (http://java.sun.com/products/jsp)
- Enterprise JavaBeans (http://java.sun.com/products/ejb)
- JSP Resource Index (http://www.jspin.com)
- Servlet Source (http://www.servletsource.com)
- ServerPages.com (http://www.serverpages.com)

疑問の解決方法

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。このフォーラムは、JRun の利用方法を知るための優れた情報源であるとともに、JRun 開発者がリアルタイムで活動する様子を知ることのできるすばらしい機会でもあります。

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、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 サービスの詳細については、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、いつでも投稿することができます。

第 1 章

JRun タグ ライブラリの概要

JRun タグ ライブラリは、JavaServer Pages 1.1 (JSP) タグ拡張 API を基準にして JavaServer Pages (JSP) カスタム タグをまとめたものです。これらのタグは次のカテゴリ の機能を提供します。

- J2EE 技術
- クライアント側のフォーム処理
- フロー制御

JRun タグ ライブラリを使用すると、関連する API のすべてを理解していなくても、データ クエリ、トランザクション、非同期メッセージ送受信、電子メール処理、および XML/XSL データ処理が実行できます。JRun タグでは、JSP と最先端の J2EE 技術をシームレスに統合します。

本書では、各 JRun タグの理解に役立つ簡単な例を示します。JRun 管理コンソール (JMC) ホーム ページからアクセスできる JRun デモ アプリケーションでは、詳細なタグの例が提供されます。リンクをクリックして各例のソース コードを表示することもできます。

JRun タグの中には、`javax.sql.RowSet` インターフェイスを実装するオブジェクトを返すものがあり 、それにより `java.sql.ResultSet` が拡張されます。`RowSet` 実装は切断されて読み取り専用になるため、メソッドを呼び出してデータベースへの接続や書き込みを行うことはできません。JRun `sql`、`storedproc`、`getmsg`、`getmail`、および `jndi` タグを使用する前に、`javax.sql.RowSet` および `java.sql.ResultSet` について把握しておくことをお勧めします。詳細については、次の URL にある JDBC API のマニュアルを参照してください。

http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html

設計目標と方針

JRun タグ ライブラリは、次の設計上の目標に沿って開発されました。

- **共通に利用できる Java API およびサーバー側処理の抽象化** - JSP 開発者が JSP に書き込む Java コード行が多すぎる場合、JSP ページの読み取りと管理が困難になることがあります。また、開発者は、標準 JSP 仕様では直接処理しない作業を実行するために各種の Java API を理解しなければならない場合もあります。これらの問題は、JSP で JavaBeans を使用することによって一部は処理されます。ただし、Bean 間の通信が必要な場合にこの方法が必ずしも有効であるとは限りません。JSP カスタム タグ モデルでは、タグベースのスクリプト 作成、タグ間の通信、親 - 子タグの通信の機能により、これらの問題がより適切な形で解決されます。

  J2EE では、Web アプリケーション開発で最も一般的に使用される Java API の概要が説明されます。開発者は、JRun タグ ライブラリを使用すると、Java コードの書き込みを行わずにほとんどのタスクが実行できます。データベース クエリは、JRun タグ ライブラリによって複雑な処理が抽象化される領域の例です。タグ拡張機能がない場合、開発者は必要なデータベース クエリを実行するために Java スクリプトレットに JDBC コードを書き込み、高度にフォーマットされたプレゼンテーション コードで、結果セットを表示する必要があります。その結果、開発サイクルが非常に長くなり、エラーの多発を招く可能性があります。また、開発者が複数の Java API 実装を使用してコードを書き込む必要がある場合には、さらに状況が悪化します。
- **簡潔な構文の提供** - タグの抽象化が特定の Java API のすべての局面をカバーするわけではありませんが、JRun タグ ライブラリは、重要な API の大部分の処理を可能にする簡潔な構文を提供することによって、バランスをとっています。
- **移植性の提供** - JSP 仕様により、カスタム タグを持つ JSP を含む JSP が、さまざまなベンダから提供される JSP コンテナ内でまったく同じ機能を持つようになりました。移植性を最大限に活用するため、JRun タグ ライブラリは、関連するすべての J2EE の要求事項に完全に合致しています。
- **業界標準のサポート** - JRun タグ ライブラリ では、移植性を最大限に活用するため、可能な限り標準 API を使用しています。

第 2 章

JRun タグ

この章では JRun タグ ライブラリのタグについて説明します。

目次

表記規約

各タグの説明には、状況に応じて次のセクションが含まれています。

- 説明
- 構文
- 属性
- スクリプト変数
- 前提条件
- 例

「構文」セクションでは、オプション属性が角かっこ内に示され、オプション属性の既定値が**太字**で示されます。

JSP カスタム タグ ライブラリの詳細については、『JRun によるアプリケーションの開発』を参照してください。

複数オブジェクト タイプ

状況に応じて、JRun タグ属性で複数の Java オブジェクト タイプがサポートされます。たとえば、foreach タグ **グループ**属性では、java.util.Enumeration、javax.naming.NamingEnumeration、java.util.Iterator、および java.lang.Object[] がサポートされています。

タグ属性の中には便利な表記をサポートするものもあります。

- 一般にタグ属性で java.lang.String 以外のオブジェクト タイプが必要とされる場合は構文のフォームが attribute="<%= x %>" になります。ここで x は、オブジェクト またはオブジェクト を返すメソッド を表します。通常、この構文では Java スクリプトレットで生成されるスクリプト変数が必要となります。ただし、JRun タグ ライブラリを使用するときに、pageContext オブジェクトにこの非文字列オブジェクト がある場合は、次の例のように attribute="scope.name" というフォームの単純な構文を使用できます。

  ```
  <tag1 id="x" scope="page" ... />
  <tag2 id="y" scope="request" .../>
  <% Object z = new Object(); %>
  <tag3 attr1="page.x" attr2="request.y" attr3="<%= z %>" .../>
  ```

- 通常は JNDI コンテキストにバインドされているオブジェクトがタグ属性で要求される場合、完全な JNDI 検索 (ルックアップ) 名の略称を指定してこのオブジェクトを自動的に取得できます。たとえば、java:comp/env/jdbc/source1 という JRun JNDI コンテキストにバインドされたデータ ソースがある場合、sql タグの datasrc 属性に "source1" という値を指定できます。このテクニックは、java:comp/env/jms、java:comp/env/mail、java:comp/env/ejb、および java:comp/env/url などの J2EE サブコンテキストにも適用されます。

タグの概要

次の表は、この章で説明するタグの概要を示します。

タグ名	説明
データベース タグ	
「sql」 8 ページ	データベース操作を実行します。
「sqlparam」 16 ページ	sql タグと併用して SQL ステートメントを動的に構築します。
「storedproc」 18 ページ	ストアド プロシージャ操作を実行します。
「procparam」 25 ページ	storedproc タグと併用して SQL ステートメントを動的に構築します。
「query2xml」 27 ページ	データベース クエリの結果を XML 形式のテキストに変換します。xslt タグと併用される場合があります。
「xslt」 29 ページ	Extensible Stylesheet Language (XSL) 変換を実行します。sql および query2xml タグと併用してクエリ結果を XML から HTML に変換する場合がありますが、外部 XML ファイルを HTML に変換する場合にも使用されます。
Java Message Service (JMS) タグ	
「sendmsg」 32 ページ	非同期メッセージを送信します。
「msgparam」 35 ページ	sendmsg タグと併用してメッセージのプロパティを設定します。
「getmsg」 36 ページ	非同期メッセージを受信します。
電子メール タグ	
「sendmail」 40 ページ	電子メール メッセージを送信します。
「mailparam」 44 ページ	sendmail タグと併用して追加のメール ヘッダを設定し、添付ファイル付きメッセージなど、マルチパートの電子メール メッセージを構築します。
45 ページの「getmail」	電子メール メッセージを受信します。
サーブレット タグ	
「servlet」 51 ページ	Java サーブレットを呼び出します。
「servletparam」 52 ページ	servlet タグと併用してサーブレットの属性を設定します。
ほかの J2EE タグ	
「jndi」 53 ページ	EJB、LDAP、CORBA などのオブジェクトやディレクトリを検索します。
「transaction」 57 ページ	分散型トランザクションを実行します。

タグ名	説明
「param」59 ページ	数式およびスクリプトレットで使用するために JSP スクリプト変数を宣言します。
フロー制御タグ	
「foreach」61 ページ	オブジェクトの集合をループ化します。
「if」63 ページ	条件ブロックを実行します。
「switch」64 ページ	Java のスイッチケース構築同様の条件ブロックを実行します。
「case」65 ページ	`switch` タグと併用して条件ブロックを実行します。
HTML フォーム タグ	
「form」66 ページ	クライアント側の JavaScript で HTML フォームを生成し、フォームの妥当性を確認します。
「input」67 ページ	クライアント側の JavaScript で HTML フォーム `input` セクションを生成し、入力フィールドの妥当性を確認します。
「select」69 ページ	クライアント側の JavaScript で HTML フォーム `select` セクションおよび `option` セクションを生成し、選択の妥当性を確認します。

タグの詳細

sql

説明 囲まれた SQL ステートメントを指定の JDBC データ ソースに送信して、データベースの操作を実行します。

使用法 sql タグの本文にフロー制御タグをネストして SQL ステートメントを動的に構築できます。sql タグの本文に sqlparam タグをネストして、バイナリ オブジェクトの SQL ステートメント への挿入などのように SQL ステートメントを構築することもできます。

sql タグが transaction タグの本文にネストされている場合、sql タグがトランザクションの構成部分となり、sql 動作を完了できるかどうかはそのトランザクションの全動作によって決まります。**トランザクション**内の sql、sendmsg、getmsg などのすべての動作が完了すると、変更がすべて保存されます。すべての動作が完了しない場合は、例外が発生し、変更がロール バックされます。

sql タグを param および foreach タグと併用すると、JSP にクエリの結果を表示できます。sql タグを query2xml および xslt タグと併用すると、クエリの結果を XML に変換した後に、XML を HTML に変換して JSP に表示することもできます。

sql タグでは、次のセクションで説明しているように 3 種類の構文がサポートされています。構文 2 に示すように、接続プールは、JDBC データ ソース オブジェクトの名前を指定した場合にサポート されます。JRun 管理者は、接続プールのサポートを使用してデータ ソース オブジェクトをあらかじめ定義し、Web アプリケーションのパフォーマンスを最大限に高める必要があります。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。

構文 1 JDBC 接続オブジェクトによってデータ ソースに接続します。

```
<jrun:sql
  connection="<%= connection name %>"
  [fetchsize="number of rows"]
  [timeout="time in seconds"]
  [maxrows="number of rows"]
  [id="variable name"]
  [scope="page|request|session|application"]>
  SQL statement
  optional <jrun:sqlparam/>
</jrun:sql>
```

属性 **connection**

必須。java.sql.Connection または java.lang.String を取ります。

JDBC 接続オブジェクト名。文字列を指定すると、connection オブジェクトは pageContext オブジェクト内にあると仮定されます。sql タグには、JDBC 接続オブジェクト があるため、このタグを閉じなくても接続オブジェクト が返されます。

fetchsize

オプション。java.lang.String、java.lang.Integer、または int を取ります。

行の追加が必要なときにデータベースから取り込まれる行数。正の整数のみ有効です。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

タグがクエリを待つ秒数を設定します。この制限を超えると例外が発生します。

maxrows

オプション。java.lang.String、java.lang.Integer、または int を取ります。

いずれかのクエリ オブジェクトに含めることができる最大行数の制限を設定します。この制限を超えると、超過した行が自動的に削除されます。

id

オプション。java.lang.String を取ります。

クエリ結果オブジェクトのスクリプト変数名。SQL ステートメントが結果セットを返さない場合や更新回数が重要ではない場合は、この属性を省略できます。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

構文 2 JDBC データ ソース オブジェクトによってデータ ソースに接続します。

```
<jrun:sql
  datasrc="data source name"
  [fetchsize="number of rows"]
  [timeout="time in seconds"]
  [maxrows="number of rows"]
  [id="variable name"]
  [scope="page|request|session|application"]
  [username="user name"]
  [password="password"]>
  SQL statement
  optional <jrun:sqlparam/>
</jrun:sql>
```

属性 **datasrc**

必須。javax.sql.DataSource または java.lang.String を取ります。

JDBC データ ソース名。文字列を指定すると、"java:comp/env/jdbc/[datasrc]" で JNDI 検索 (ルックアップ) を実行することで、データ ソース オブジェクトが取得できると仮定されます。

分散型トランザクションの一部として transaction タグの本文で sql タグを使用するには、この属性をタイプ java.lang.String にする必要があります。

fetchsize

オプション。java.lang.String、java.lang.Integer、または int を取ります。

行の追加が必要なときにデータベースから取り込まれる行数。正の整数のみ有効です。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

タグがクエリを待つ秒数を設定します。この制限を超えると例外が発生します。

maxrows

オプション。java.lang.String、java.lang.Integer、または int を取ります。

いずれかのクエリ オブジェクトに含めることができる最大行数の制限を設定します。この制限を超えると、超過した行が自動的に削除されます。

id

オプション。java.lang.String を取ります。

クエリ結果オブジェクトのスクリプト変数名。SQL ステートメントが結果セットを返さない場合や更新回数が重要ではない場合は、この属性を省略できます。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

username

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのユーザ名。

password

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのパスワード。

構文 3 JDBC ドライバ クラスとデータベース URL を使用してデータ ソースに接続します。

```
<jrun:sql
  driver="JDBC driver class"
  url="JDBC URL"
  [fetchsize="number of rows"]
  [timeout="time in seconds"]
  [maxrows="number of rows"]
  [id="variable name"]>
  SQL statement
  optional <jrun:sqlparam/>
</jrun:sql>
```

属性 **driver**

必須。java.lang.String を取ります。

JDBC ドライバのクラス名。

url

必須。java.lang.String を取ります。

データベースへの JDBC URL。

fetchsize

オプション。java.lang.String、java.lang.Integer、または int を取ります。

行の追加が必要なときにデータベースから取り込まれる行数。指定した行数が 0 の場合、この属性は無視されます。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

タグがクエリを待つ秒数を設定します。この制限を超えると例外が発生します。

maxrows

オプション。java.lang.String、java.lang.Integer、または int を取ります。

いずれかのクエリ オブジェクトに含めることができる最大行数の制限を設定します。この制限を超えると、超過した行が自動的に削除されます。

id

オプション。java.lang.String を取ります。

クエリ結果オブジェクトのスクリプト変数名。SQL ステートメントが結果セットを返さない場合や更新回数が重要ではない場合は、この属性を省略できます。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

username

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのユーザ名。

password

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのパスワード。

スクリプト変数 id を指定した場合、sql タグによって返される結果セットは pageContext オブジェクトに allaire.taglib.QueryTable のインスタンスとして格納されます。返されるオブジェクトのスコープは、sql タグの scope によって決まり、その既定値は page です。param タグを使用してこのオブジェクトのスクリプト変数を宣言すると、定義されたスコープ内でその変数を暗黙オブジェクトのように使用できます。

QueryTable には javax.sql.RowSet インターフェイスが実装されています。このインターフェイスによって java.sql.ResultSet が拡張されます。sql、storedproc、getmsg、getmail、および jndi タグを使用する前に、javax.sql.RowSet および java.sql.ResultSet について把握しておくことをお勧めします。詳細については、次の URL にある JDBC API のマニュアルを参照してください。

http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html

この特定の RowSet 実装は接続解除されて読み取り専用になるため、メソッドを呼び出してデータベースへの接続や書き込みを行うことはできません。QueryTable では、バイト配列として BLOB (Binary Large Object)、文字配列として CLOB (Character Large Object) を返します。

JDBC API を熟知しているユーザは、sql 結果セットを、QueryTable ではなく RowSet に直接タイプ変換する方が適切な場合があります。これは、sql タグとスクリプト変数を併用する場合の共通のシナリオです。この例では getObject が使用されていますが、RowSet には異なるオブジェクト タイプを使用するほかの多くの getter メソッドがあります。

- 次に示すように param タグで QueryTable または RowSet にタイプ変換します。
  ```
  <jrun:param id="result" type="QueryTable"/>
  ```
  または
  ```
  <jrun:param id="result" type="javax.sql.RowSet"/>
  ```
- 結果セットをループ化します。
  ```
  <jrun:foreach group='<%=result%>'>
  <%= result.getObject("column1") %><br>
  <%= result.getObject("column2") %><br>
  </jrun:foreach>
  ```

次の表では、開発者が sql クエリ結果で一般的に使用するメソッドについて説明しています。

メソッド	返されるタイプ	説明
next()	Boolean	TRUE の場合、結果セットで次の行を取得します。
getObject(int)	Object	インデックス番号によって列を取得します。
getObject(String)	Object	名前によって列を取得します。
get(int)	String	インデックス番号によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(int) の使用をお勧めします。
get(String)	String	名前によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(String) の使用をお勧めします。

RowSet には、QueryTable の専用メソッドと同じ標準メソッドが用意されています。ただし、次の機能を実行するには QueryTable を使用する必要があります。

- **テーブルのチェーン化**：RowSet とは異なり、QueryTable では、テーブルをチェーン化して、複数の結果セットを返すデータベース クエリがサポートされます。この機能は、複数の行セットを返すクエリがデータベース ドライバでサポートされている場合にのみ有効であり、sql タグでこの機能を使用します。
- **更新回数**：QueryTable getUpdateCount メソッドによってデータベースの更新回数を取得できます。
- **拡張 next メソッド**：QueryTable next メソッドを呼び出すたびに、Object 配列としてフィールド値を使用できます。そのため、RowSet を扱うときに各値について getObject を呼び出す必要はありません。

次の表は、関連する allaire.taglib.QueryTable メソッドの一覧です。

メソッド	返されるタイプ	説明
テーブルのチェーン化		
nextTable()	Table	結果セット チェーンで次のテーブルを取得します。
previousTable()	Table	結果セット チェーンで前のテーブルを取得します。
firstTable()	Table	結果セット チェーンで最初のテーブルを取得します。
lastTable()	Table	結果セット チェーンで最後のテーブルを取得します。
setNextTable(Table rs)	なし	結果セット チェーンで次のテーブルを設定します。
setPreviousTable(Table rs)	なし	結果セット チェーンで前のテーブルを設定します。
更新回数		
getUpdateCount()	Int	データベース更新回数を取得します。
拡張 next メソッド		
next()	Boolean	TRUE の場合、テーブルの次の行を取得します。フィールド値は Object 配列で使用できます。

前提条件 Web アプリケーションで sql タグを使用する前に、JDBC データ ソースを使用可能にする必要があります。JDBC データ ソース オブジェクトに接続するには (sql 構文 2)、JRun で JDBC データ ソースをセットアップする必要があります。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。

例 この JSP は、sql タグの 3 種類の使用方法と、クエリ結果の表示方法を示しています。

```
<%@ page import="java.sql.*,javax.sql.*,allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>sql Tag Example</title></head><body>
<%--
sql タグの構文 1 フォームを使用する前に、JDBC 接続オブジェクトを作成する必要があり
        ます。
--%>
<%
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
Connection con =
DriverManager.getConnection("jdbc:odbc:DBNAME", "username",
"password");
%>
<%--
(sql 構文 1) JDBC 接続オブジェクトを使用してデータ ソースに接続し、SQL ステートメ
        ントを実行します。
--%>
<jrun:sql connection="<%= con %>" id="q1">
select * from Table1
</jrun:sql>

<%--
(sql 構文 2) JDBC データ ソース (java:comp/env/jdbc/dsn1) を使用してデータ
        ソースを検索し、SQL ステートメントを実行します。
--%>
<jrun:sql datasrc="dsn1" id="q2">
select * from Table1
</jrun:sql>

<%--
(sql 構文 3) JDBC ドライバ クラスとデータベースへの URL を指定することによってデー
        タ ソースに接続し、SQL ステートメントを実行します。
--%>
<jrun:sql driver="sun.jdbc.odbc.JdbcOdbcDriver"
url="jdbc:odbc:DBNAME" id="q3">
select * from Table1
</jrun:sql>

<%--
param タグを使用して変数を宣言し、sql タグによって返されるオブジェクトを
        javax.sql.RowSet にタイプ変換します。getObject メソッドを
        使用してクエリ結果からデータベース列を取得します。
--%>
<jrun:param id="q3" type="javax.sql.RowSet"/>
<jrun:foreach group="<%= q3 %>">
<%= q3.getObject("id") %><br>
<%= q3.getObject("lastname") %><br>
<%= q3.getObject("firstname") %><br>
</jrun:foreach>
<%--
```

```
java:comp/env/jdbc/db1a を使用してデータ ソースを検索し、複数の結果セットを返す
        SQL ステートメントを実行します。param タグを使用して変数を宣言し、sql タ
        グによって返されるオブジェクトを QueryTable にタイプ変換します。最初の結
        果セットをループ化し、getObject メソッドを使用してデータベース列を取得し
        ます。nextTable メソッドを使用して次の結果セットにループ化します。
--%>
<jrun:sql datasrc="db1a" id="result1">
select * from tablex
select * from table2
select * from table3
select * from table4
</jrun:sql>
<jrun:param id="result1" type="QueryTable"/>
<% do { %>
<jrun:foreach group="<%= result1 %>">
<%= result1.getObject("name") %><br>
<%= result1.getObject("amount") %><br>
</jrun:foreach>
<% } while ((result1 = (QueryTable)result1.nextTable()) != null); %>
</body></html>
```

sqlparam

説明 sql タグと併用して SQL ステートメントを動的に構築します。

使用法 このタグは、テキスト形式の表示がないオブジェクトを sql タグ本文にネストされた SQL ステートメントに挿入する必要がある場合に非常に便利です。

構文

```
<jrun:sqlparam
  value="SQL parameter"
  [sqltype="SQL type"]
  [scale="number of digits after decimal"]
/>
```

属性

value

必須。java.lang.Object を取ります。

SQL パラメータ値。

sqltype

オプション。java.lang.String、java.lang.Integer、または int を取ります。

SQL データ タイプ。value 属性を null として指定する場合は、必ず sqltype を指定してください。

sqlparam でサポートされる整数値は、java.sql.Types のサポート対象の SQL タイプを基準にしています。文字列を指定した場合は、java.sql.Types のいずれかの定数文字列である必要があります。有効な文字列値は次のとおりです。

ARRAY	DISTINCT	REAL
BIGINT	DOUBLE	REF
BINARY	FLOAT	SMALLINT
BIT	INTEGER, JAVA_OBJECT	STRUCT
BLOB	LONGVARBINARY	TIME
CHAR	LONGVARCHAR	TIMESTAMP
CLOB	NULL	TINYINT
DATE	NUMERIC	VARBINARY
DECIMAL	OTHER	VARCHAR

scale

オプション。java.lang.String、java.lang.Integer、または int を取ります。

小数点以下の桁数。sqltype が NUMERIC または DECIMAL の場合にのみ有効です。

例 この JSP は、sql タグ内での sqlparam タグの使用方法を示しています。

```
<%@ page import=allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>title>sqlparam Tag Example</title></head><body>
<%--
JDBC ドライバ クラスとデータベースへの URL を指定することによってデータ ソースに接
        続し、次に sqlparam 値を指定します。
--%>
<jrun:sql driver="sun.jdbc.odbc.JdbcOdbcDriver"
url="jdbc:odbc:DBNAME" id="result">
select * from Table1 where id = <jrun:sqlparam value="<%= “param1" %>"/>
</jrun:sql>

</body></html>
```

storedproc

説明 囲まれたストアド プロシージャを指定の JDBC データ ソースに送信して、データベースの操作を実行します。

使用法 storedproc タグが transaction タグの本文にネストされている場合、storedproc タグがトランザクションの構成部分となり、storedproc 動作を完了できるかどうかはそのトランザクションの全動作によって決まります。トランザクション内の storedproc、sendmsg、getmsg などのすべての動作が完了すると、変更がすべて保存されます。すべての動作が完了しない場合は、例外が発生し、変更がロール バックされます。

storedproc タグを param および foreach タグと併用すると、JSP にクエリの結果を表示できます。storedproc タグを query2xml および xslt タグと併用すると、クエリの結果を XML に変換した後に、XML を HTML に変換して JSP に表示することもできます。

storedproc タグ内で sqlparam タグを使用すると、IN パラメータ渡しとして扱われます。procparam タグを使用すると、OUT/INOUT パラメータ渡しとして扱われます。

storedproc タグでは、次のセクションで説明しているように 3 種類の構文がサポートされています。次の構文 2 に示すように、接続プールは、JDBC データ ソース オブジェクト の名前を指定した場合にサポート されます。JRun 管理者は、接続プールのサポートを使用してデータ ソース オブジェクトをあらかじめ定義し、Web アプリケーションのパフォーマンスを最大限に高める必要があります。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。

構文 1 JDBC 接続オブジェクトによってデータ ソースに接続します。

```
<jrun:storedproc
  connection="<%= connection name %>"
  [fetchsize="number of rows"]
  [timeout="time in seconds"]
  [maxrows="number of rows"]
  [id="scripting variable name"]
  [scope="page|request|session|application"]>
  {call stored_procedure_name(
  optional <jrun:sqlparam/>
  optional <jrun:procparam/>
  )}
</jrun:storedproc>
```

属性 **connection**

必須。java.sql.Connection または java.lang.String を取ります。

JDBC 接続オブジェクト名。文字列を指定すると、connection オブジェクトは pageContext オブジェクト 内にあると 仮定されます。storedproc タグには、JDBC 接続オブジェクトがあるため、このタグを閉じなくても接続オブジェクトが返されます。

fetchsize

オプション。java.lang.String、java.lang.Integer、または int を取ります。

行の追加が必要なときにデータベースから取り込まれる行数。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

タグがクエリを待つ秒数を設定します。この制限を超えると例外が発生します。

maxrows

オプション。java.lang.String、java.lang.Integer、または int を取ります。

いずれかのクエリ オブジェクトに含めることができる最大行数の制限を設定します。この制限を超えると、超過した行が自動的に削除されます。

id

オプション。java.lang.String を取ります。

クエリ結果オブジェクトのスクリプト変数名。ストアド プロシージャが結果セットを返さない場合や更新回数が重要ではない場合は、この属性を省略できます。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

構文 2 JDBC データ ソース オブジェクトによってデータ ソースに接続します。

```
<jrun:storedproc
  datasrc="data source name"
  [fetchsize="number of rows"]
  [timeout="time in seconds"]
  [maxrows="number of rows"]
  [id="variable name"]
  [scope="page|request|session|application"]
  [username="user name"]
  [password="password"]>
  {call stored_procedure_name(
  optional <jrun:sqlparam/>,
  optional <jrun:procparam/>
  )}
</jrun:storedproc>
```

属性 **datasrc**

必須。javax.sql.DataSource または java.lang.String を取ります。

JDBC データ ソース オブジェクト名。文字列を指定すると、"java:comp/env/jdbc/[datasrc]" で JNDI 検索 (ルックアップ) を実行することで、データ ソース オブジェクトが取得できると仮定されます。

分散型トランザクションの一部として transaction タグの本文で storedproc タグを使用するには、この属性をタイプ java.lang.String にする必要があります。

fetchsize

オプション。java.lang.String、java.lang.Integer、または int を取ります。

行の追加が必要なときにデータベースから取り込まれる行数。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

タグがクエリを待つ秒数を設定します。この制限を超えると例外が発生します。

maxrows

オプション。java.lang.String、java.lang.Integer、または int を取ります。

いずれかのクエリ オブジェクトに含めることができる最大行数の制限を設定します。この制限を超えると、超過した行が自動的に削除されます。

id

オプション。java.lang.String を取ります。

クエリ結果オブジェクトのスクリプト変数名。ストアド プロシージャが結果セットを返さない場合や更新回数が重要ではない場合は、この属性を省略できます。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

username

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのユーザ名。

password

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのパスワード。

構文 3 JDBC ドライバ クラスとデータベース URL を使用してデータ ソースに接続します。

```
<jrun:storedproc
  driver="JDBC driver class"
  url="JDBC URL"
  [fetchsize="number of rows"]
  [timeout="time in seconds"]
  [maxrows="number of rows"]
  [id="variable name"]>
  {call stored_procedure_name(
  optional <jrun:sqlparam/>,
  optional <jrun:procparam/>
  )}
</jrun:storedproc>
```

属性 **driver**

必須。java.lang.String を取ります。

JDBC ドライバのクラス名。

url

必須。java.lang.String を取ります。

データベースへの JDBC URL。

fetchsize

オプション。java.lang.String、java.lang.Integer、または int を取ります。

行の追加が必要なときにデータベースから取り込まれる行数。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

タグがクエリを待つ秒数を設定します。この制限を超えると例外が発生します。

maxrows

オプション。java.lang.String、java.lang.Integer、または int を取ります。

いずれかのクエリ オブジェクトに含めることができる最大行数の制限を設定します。この制限を超えると、超過した行が自動的に削除されます。

id

オプション。java.lang.String を取ります。

クエリ結果オブジェクトのスクリプト変数名。SQL ステートメントが結果セットを返さない場合や更新回数が重要ではない場合は、この属性を省略できます。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

username

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのユーザ名。

password

オプション。java.lang.String を取ります。

データ ソースにアクセスするためのパスワード。

スクリプト変数

id を指定した場合、storedproc タグによって返される結果セットは pageContext オブジェクトに allaire.taglib.QueryTable のインスタンスとして格納されます。返されるオブジェクトのスコープは、タグの scope によって決まり、その既定値は page です。param タグを使用してこのオブジェクトのスクリプト変数を宣言すると、定義されたスコープ内でその変数を暗黙オブジェクトのように使用できます。

QueryTable には javax.sql.RowSet インターフェイスが実装されています。このインターフェイスによって java.sql.ResultSet が拡張されます。sql、storedproc、getmsg、getmail、および jndi タグを使用する前に、javax.sql.RowSet および java.sql.ResultSet について把握しておくことをお勧めします。詳細については、次の URL にある JDBC API のマニュアルを参照してください。

http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html

この特定の RowSet 実装は接続解除されて読み取り専用になるため、メソッドを呼び出してデータベースへの接続や書き込みを行うことはできません。QueryTable では、バイト配列として BLOB (Binary Large Object)、文字配列として CLOB (Character Large Object) を返します。

JDBC API を熟知しているユーザは、storedproc 結果セットを、QueryTable ではなく RowSet に直接タイプ変換する方が適切な場合があります。これは、sql タグとスクリプト変数を併用する場合の共通のシナリオです。この例では getObject が使用されていますが、RowSet には異なるオブジェクト タイプを使用するほかの多くの getter メソッドがあります。

- 次に示すように param タグで QueryTable または RowSet にタイプ変換します。
  ```
  <jrun:param id="result" type="QueryTable"/>
  ```
  または
  ```
  <jrun:param id="result" type="javax.sql.RowSet"/>
  ```
- 結果セットをループ化します。
  ```
  <jrun:foreach group='<%=result%>'>
  <%= result.getObject("column1") %><br>
  <%= result.getObject("column2") %><br>
  </jrun:foreach>
  ```

次の表では、storedproc クエリ結果で最も一般的に使用される RowSet メソッドについて説明しています。

メソッド	返されるタイプ	説明
next()	Boolean	TRUE の場合、結果セットで次の行を取得します。
getObject(int)	Object	インデックス番号によって列を取得します。
getObject(String)	Object	名前によって列を取得します。
get(int)	String	インデックス番号によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(int) の使用をお勧めします。
get(String)	String	名前によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(String) の使用をお勧めします。

RowSet には、QueryTable の専用メソッドと同じ標準メソッドが用意されています。ただし、次の機能を実行するには QueryTable を使用する必要があります。

- **テーブルのチェーン化**：RowSet とは異なり、QueryTable では、テーブルをチェーン化して、複数の結果セットを返すデータベース クエリがサポートされます。この機能は、複数の行セットを返すクエリがデータベース ドライバでサポートされている場合にのみ有効であり、storedproc タグでこの機能を使用します。

- **更新回数**：QueryTable getUpdateCount メソッドによってデータベースの更新回数を取得できます。
- **拡張 next メソッド**：QueryTable next メソッドを呼び出すたびに、Object 配列としてフィールド値を使用できます。そのため、RowSet を扱うときに各値について getObject を呼び出す必要はありません。

次の表は、関連する allaire.taglib.QueryTable メソッドの一覧です。

メソッド	返されるタイプ	説明
テーブルのチェーン化		
nextTable()	Table	結果セット チェーンで次のテーブルを取得します。
previousTable()	Table	結果セット チェーンで前のテーブルを取得します。
firstTable()	Table	結果セット チェーンで最初のテーブルを取得します。
lastTable()	Table	結果セット チェーンで最後のテーブルを取得します。
setNextTable(Table rs)	なし	結果セット チェーンで次のテーブルを設定します。
setPreviousTable(Table rs)	なし	結果セット チェーンで前のテーブルを設定します。
更新回数		
getUpdateCount()	Int	データベース更新回数を取得します。
拡張 next メソッド		
next()	Boolean	TRUE の場合、テーブルで次の行を取得します。フィールド値は Object 配列で使用できます。

前提条件 Web アプリケーションで storedproc タグを使用する前に、JDBC データ ソースを使用可能にする必要があります。JDBC データ ソース オブジェクトに接続するには (storedproc 構文 2)、JRun で JDBC データ ソースをセットアップする必要があります。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。

例 この JSP は、JDBC データ ソース オブジェクトと storedproc タグを併用する方法を示しています。データ ソースへの接続およびクエリ結果の表示の例については、sql の例を参照してください。

```
<%@ page import="java.sql.*,javax.sql.*,allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>storedproc Tag Example</title></head><body>
<%--
```

```
JDBC データ ソース (java:comp/env/jdbc/db3a) を使用してデータ ソースを検索し、
        ストアド プロシージャを実行します。
--%>
jrun:storedproc datasrc="db3a" id="result">
{call sp_get_all_client_account}
</jrun:storedproc>
<%--
結果セットを表示します。
--%>
<%= pageContext.getAttribute("result") %>
</body></html>
```

procparam

説明 storedproc タグと併用してストアド プロシージャを動的に構築します。storedproc タグで procparam タグを使用すると、OUT/INOUT パラメータ渡しとして扱われます。

構文

```
<jrun:procparam
  id="variable name"
  [value="stored procedure parameter"]
  [scope="page|request|session|application"]
  [sqltype="SQL type"]
  [scale="number of digits after decimal"]
/>
```

属性 **id**

必須。java.lang.String を取ります。

パラメータの value のスクリプト変数名。

value

オプション。java.lang.Object を取ります。

ストアド プロシージャ パラメータ値。

scope

オプション。java.lang.String を取ります。

パラメータの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

sqltype

オプション。java.lang.String、java.lang.Integer、または int を取ります。

procparam が、java.sql.Types でサポートされている SQL タイプを基準にしている整数値。文字列を指定した場合は、java.sql.Types のいずれかの定数文字列である必要があります。有効な文字列値は次のとおりです。

ARRAY	DISTINCT	REAL
BIGINT	DOUBLE	REF
BINARY	FLOAT	SMALLINT
BIT	INTEGER, JAVA_OBJECT	STRUCT
BLOB	LONGVARBINARY	TIME
CHAR	LONGVARCHAR	TIMESTAMP
CLOB	NULL	TINYINT
DATE	NUMERIC	VARBINARY
DECIMAL	OTHER	VARCHAR

scale

オプション。java.lang.String、java.lang.Integer、または int を取ります。

小数点以下の桁数。sqltype が NUMERIC または DECIMAL の場合にのみ有効です。

例 この JSP は、procparam タグを storedproc タグと併用する方法を示しています。

```
<%@ page import="java.sql.*,javax.sql.*,allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>procparam Tag Example</title></head><body>
<%--
JDBC データ ソース (java:comp/env/jdbc/db3a) を使用してデータ ソースを検索し、
sqlparam および procparam タグが含まれている storedproc タグを使用してストアド
プロシージャを実行します。
--%>
<jrun:storedproc datasrc="db3a" id="result">
{call sp_add_new_client(
<jrun:sqlparam value="Fiona"/>,
<jrun:sqlparam sqltype="VARCHAR"/>,
<jrun:procparam id="clientId" sqltype="INTEGER"/>
)}
</jrun:storedproc>
<%--
結果セットと clientId を表示します。
--%>
<%= pageContext.getAttribute("result") %><br>
<%= pageContext.getAttribute("clientId") %>
</body>
</html>
```

query2xml

説明 テーブル形式データを XML 形式に単純変換します。

使用法 このタグを使用して、sql、getmail、および getmsg 結果を XML に変換できます。クエリ オブジェクト の文字列に < や & などの分離されたマークアップ文字が含まれている場合、これらの文字は結果の XML の CDATA セクションに配置されます。この機能は HTML マークアップを扱う場合に特に役立ちます。query2xml タグと xslt タグを併用すると、XML データを HTML やほかの XML ドキュメント タイプに変換できます。

構文

```
<jrun:query2xml
  query="query result object name"
  [id="variable name"]
  [type="TEXT|DOM"]
  [scope="page|request|session|application"]
  [rootname="table|root XML element name"]
  [rowname="row|XML element name for rows"]
/>
```

属性

query

必須。java.lang.String、java.sql.ResultSet、javax.sql.RowSet を取り ます。

クエリ結果オブジェクト名。文字列を指定すると、pageContext オブジェクト内からクエリ結果オブジェクトを見つけることができると仮定されます。

id

オプション。java.lang.String を取ります。

XML ドキュメント オブジェクトやリーダー オブジェクトを参照するスクリプト変数名。id を指定しない場合、生成される XML 出力が呼び出し JSP に埋め込まれます。

type

オプション。java.lang.String を取ります。

このタグが生成する XML オブジェクトのタイプ。有効な値は DOM および TEXT です。DOM を指定すると、org.w3c.dom.Document のインスタンスが作成されます。TEXT を指定すると、java.io.BufferedReader のインスタンスが作成されます。既定値は TEXT です。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

rootname

オプション。java.lang.String を取ります。

クエリ タグ名。既定値は table です。

rowname

オプション。java.lang.String を取ります。

行の XML 要素名。既定値は row です。

スクリプト変数

id を指定した場合、query2xml タグによって返される結果は、type 属性に従って、org.w3c.dom.Document または java.io.BufferedReader のインスタンスとして pageContext オブジェクトに格納されます。返されるオブジェクトのスコープは、タグの scope によって決まり、その既定値は page です。

例

この JSP は、query2xml タグを使用してクエリ結果を XML 形式に変換する方法を示しています。

```
<%@ page contentType"text/xml" %>
<?xml version="1.0"?>
<%@ taglib uri="jruntags" prefix="jrun" %>
<%--
JDBC ドライバ クラスとデータベースへの URL を指定することによってデータ ソースに接
        続し、SQL ステートメントを実行します。
--%>
<jrun:sql driver="sun.jdbc.odbc.JdbcOdbcDriver"
url="jdbc:odbc:DBNAME" id="rs">
select * from Table1
</jrun:sql>
<%--
データベース クエリ結果を XML に変換します。
--%>
<jrun:query2xml query="rs" />
```

xslt

説明 Extensible Stylesheet Language Transformations (XSLT) を使用して、XML 形式のデータを HTML 形式やほかの XML 形式に変換します。

このタグは、query2xml 変換の結果を動的に HTML に変換する際に有用です。

構文 1 囲まれた XML 入力を使用して XSL 変換を実行します。

```
<jrun:xslt
  xsl="URL to stylesheet"
  [id="variable name"]
  [scope="page|request|session|application"]>
  XML input
</jrun:xslt>
```

属性 **xsl**

必須。java.lang.String、java.net.URL を取ります。

XSL ファイルへの URL。文字列を使用して次のように指定できます。

- 絶対 URL (/..)
- このタグの使用するページに対する相対 URL
- 絶対 URL (http://..)

移植性を高めるために、java.net.URL オブジェクトを指定して絶対 URL を使用することもできます。

id

オプション。java.lang.String を取ります。

XSL 変換出力を参照するスクリプト変数名。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、または application です。既定値は page です。

構文 2 URL から XML 入力を取得して XSL 変換を実行します。

```
<xslt
  xml="URL to XML file"
  xsl="URL to stylesheet"
  [id="scripting variable name"]
  [scope="page|request|session|application"]
/>
```

属性 **xml**

オプション。java.lang.String を取ります。

XML ファイルへの URL。文字列を使用して絶対 URL (/..)、このタグが使用されているページからの相対 URL あるいは、完全な URL (http://..) を指定することができます。移植性を高めるために、java.net.URL オブジェクトを指定して絶対 URL を使用することもできます。

xsl

必須。java.lang.String を取ります。

XSL ファイルへの URL。文字列を使用して絶対 URL (/..)、このタグが使用されているページからの相対 URL あるいは、完全な URL (http://..) を指定することができます。移植性を高めるために、java.net.URL オブジェクトを指定して絶対 URL を使用することもできます。

id

オプション。java.lang.String を取ります。

変換出力名。

scope

オプション。java.lang.String を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、または application です。既定値は page です。

スクリプト変数 id を指定した場合、xslt タグによって返される結果は、java.io.BufferedReader のインスタンスとして pageContext オブジェクト に格納されます。返されるオブジェクトのスコープは、タグの scope によって決まり、その既定値は page です。

例 この JSP は、両方のタイプの xslt 構文の使用方法を示しています。最初に、構文 1 で、query2xml 変換の結果を変換する方法を示します。次に、構文 2 で、参照される XSL ファイルを使用して参照される XML ファイルを変換する方法を示します。

```
<%@ page import="java.sql.*,javax.sql.*,allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>xslt Tag Example</title></head><body>
<%--
(xslt 構文 1) xslt タグを使用して query2xml 変換の結果を変換します。
--%>
<jrun:sql driver="sun.jdbc.odbc.JdbcOdbcDriver"
url="jdbc:odbc:DBNAME" id="rs">
select * from Table1
</jrun:sql>
<jrun:xslt xsl="format.xsl">
<jrun:query2xml query="rs"/>
</jrun:xslt>
<%--
(xslt 構文 2) xslt タグを使用して XML ファイルを変換します。
--%>
```

```
<jrun:xslt xml="test.xml"xsl="format.xsl"/>

</body></html>
-----------------------------
```

次の例は、名前および量の列が含まれている結果セットについて、xslt タグと併用できる簡単な XSL ファイルです。

```
<xsl:stylesheet xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"/>
<xsl:template match="table">
<table border="1" width="250">
<tr><th>Name</th><th>Amount</th></tr>
<xsl:apply-templates/>
</table>
</xsl:template>
<xsl:template match="row">
<tr><xsl:apply-templates/></tr>
</xsl:template>
<xsl:template match="name">
<td><xsl:apply-templates/></td>
</xsl:template>
<xsl:template match="amount">
<td><xsl:apply-templates/></td>
</xsl:template>
</xsl:stylesheet>
```

sendmsg

説明 Java Message Service (JMS) を使用して、囲まれたテキスト メッセージを送信します。

sendmsg タグの本文内にフロー制御タグをネストして、テキスト メッセージを動的に構築できます。sendmsg タグに msgparam タグをネストして、追加のメッセージ プロパティを指定することもできます。sendmsg タグが transaction タグ内にネストされている場合、sendmsg タグがトランザクションの構成部分となり、sendmsg 動作を完了できるかどうかはそのトランザクションの全動作によって決まります。トランザクション内の sql、sendmsg、getmsg などのすべての動作が完了すると、変更がすべて保存されます。すべての動作が完了しない場合は、例外が発生し、変更がロール バックされます。

メモ
sendmsg タグは、ポイントツーポイントのメッセージにのみ使用されます。

sendmsg タグでは、JMS QueueConnectionFactory および JNDI 検索 (ルックアップ) による JMS メッセージ キューを使用します。JMS 接続プールが指定されます。JRun 管理者は、Web アプリケーションで使用するメッセージ キュー オブジェクトをあらかじめ定義しておく必要があります。詳細については、33 ページの「前提条件」セクションを参照してください。

構文

```
<jrun:sendmsg
  msgsrc="message queue connection factory"
  queue="message queue"
  [username="user"]
  [password="password"]
  [delivery="PERSISTENT|NON_PERSISTENT"]
  [priority="integer"]
  [expire="never|time in milliseconds"]>
  オプション <jrun:msgparam/> のある電子メール メッセージ本文
</jrun:sendmsg>
```

属性

msgsrc

必須。java.lang.String または javax.jms.QueueConnectionFactory を取ります。

メッセージのソースであるメッセージ QueueConnectionFactory。文字列を指定すると、"java:comp/env/jms/[msgsrc]" で JNDI 検索 (ルックアップ) を実行することで、QueueConnectionFactory が取得できると仮定されます。QueueConnectionFactory オブジェクトを既定の InitialContext から取得できない場合は、jndi タグを使用して別の InitialContext を検索してファクトリを見つける必要があります。詳細については、「jndi」53 ページセクションを参照してください。

queue

必須。属性として、java.lang.String または javax.jms.Queue を取ります。

メッセージ キュー オブジェクト。文字列を指定すると、"java:comp/env/jms/[queue]" で JNDI 検索 (ルックアップ) を実行することで、キューが取得できると仮定されます。Queue オブジェクトを既定の InitialContext から取得できない場合は、jndi タグを使用して別の InitialContext を検索してキューを見つける必要があります。詳細については、「jndi」53 ページセクションを参照してください。

username

オプション。java.lang.String を取ります。

メッセージ キュー認証のためのユーザ名。

password

オプション。java.lang.String を取ります。

メッセージ キュー認証のためのパスワード。

delivery

オプション。java.lang.String を取ります。

メッセージ配信モード。有効な値は PERSISTENT と NON_PERSISTENT です。既定値は PERSISTENT です。

priority

オプション。java.lang.String、java.lang.Integer、または int を取ります。

メッセージの優先レベル。

expire

オプション。java.lang.String、java.lang.Long、または long を取ります。

メッセージが期限切れになるまでの時間間隔 (ミリ秒単位)。既定では期限はありません。

前提条件 sendmsg および getmsg タグを使用するには、EJB および JMS オプションを含む完全な JRun インストールを実行する必要があります。ユーザ独自のメッセージ キューを指定する場合は、JRun サーバーの local.properties ファイルを編集して、次のエントリを加えます。

```
jms.queue.[your queue name].description=[description]
jms.queue.[your queue name].display-name=[displayname in JMC]
```

例 この JSP は、sendmsg タグの最も一般的な使用方法を示しています。通常、Web アプリケーションでは、JNDI 検索 (ルックアップ) を実行して JMS QueueConnectionFactory を取得する必要があります。また、同じアプローチで JMS メッセージ キュー オブジェクトの事前定義を取得する必要もあります。param タグと jndi タグはこれらの分散オブジェクトの取得に使用されます。

まず、param タグを使用して、JMS QueueConnectionFactory と JMS メッセージキューを保持するスクリプト変数を宣言します。次に jndi タグを使用してオブジェクトを見つけます。最後に、sendmsg タグが JNDI から返されるオブジェクトを使用してメッセージ送信動作を実行します。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>sendmsg Tag Example</title></head><body>
<%--
java:comp/env/jms/QueueConnectionFactory を使用して
        QueueConnectionFactory を検索し、java:comp/env/jms/Queue1 を
        使用してキューを検索します。
--%>
<jrun:sendmsg msgsrc="QueueConnectionFactory" queue="Queue1">
This is a text message.
</jrun:sendmsg>

<%--
ファクトリとキューが既定の InitialContext で見つからない場合は、jndi タグを使用して検索できます。
--%>
<jrun:jndi action="lookup"
name="java:comp/env/jms/QueueConnectionFactory" id="f"/>
<jrun:jndi name="java:comp/env/jms/Queue1" id="q"/>

<jrun:sendmsg msgsrc="<%= page.getAttribute("f") %>"
queue="<%= page.getAttribute("q") %>">
This is a text message.
</jrun:sendmsg>
</body></html>
```

msgparam

説明 sendmsg タグと併用して JMS メッセージのプロパティを指定します。

構文
```
<jrun:msgparam
  name="name"
  value="value"
/>
```

属性

name

必須。java.lang.Object を取ります。

JMS メッセージプロパティ名。

value

必須。java.lang.String を取ります。

JMS メッセージプロパティ値。

例 この JSP は、sendmsg タグでの msgparam タグの使用方法を示しています。

```
<%@ page import="java.sql.*,javax.sql.*,allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>msgparam Tag Example</title></head><body>
<%--
sendmsg タグ内の msgparam タグにある foo という名前のメッセージ パラメータを指定
        します。
--%>
<jrun:sendmsg msgsrc="QueueConnectionFactory" queue="Queue1">
<jrun:msgparam name="foo" value="bar"/>
This is a text message.
</jrun:sendmsg>
</body></html>
```

getmsg

説明 指定された Java Message Service (JMS) メッセージ キューからメッセージを取得します。

このタグでは、JMS メッセージ セレクタを表す、SQL SELECT に囲まれた文字列を使用して、getmsg タグで返されるメッセージ キュー名、検索条件、メッセージ取得順序、メッセージのプロパティを識別します。getmsg タグ内でフロー制御タグを使用してメッセージ セレクタ ステートメントを動的に構築できます。getmsg タグが transaction タグの中にある場合、getmsg タグがトランザクションの構成部分となり、getmsg 動作を完了できるかどうかはそのトランザクションの全動作によって決まります。トランザクション内の sql、sendmsg、getmsg などのすべての動作が完了すると、変更がすべて保存されます。すべての動作が完了しない場合は、例外が発生し、変更がロール バックされます。getmsg タグでは、allaire.taglib.Transaction インターフェイスを使用して、transaction タグにネストされているかどうかを確認します。

getmsg タグでは、JMS QueueConnectionFactory および JNDI 検索 (ルックアップ) による JMS メッセージ キューを使用します。JMS 接続プールが指定されます。JRun 管理者は、Web アプリケーションで使用するメッセージ キュー オブジェクトをあらかじめ定義しておく必要があります。詳細については、38 ページの「前提条件」セクションを参照してください。

JMS メッセージ セレクタ文字列はオプションです。JMS メッセージ セレクタの詳細については、javax.jms.Message API のマニュアルを参照してください。

構文

```
<jrun:getmsg
  msgsrc="queue connection factory"
  [username="user name"]
  [password="password"]
  [id="variable name"]
  [scope="page|request|session|application"]>
  SELECT {プロパティ名のカンマ区切りリストまたは *}
  FROM {メッセージ キュー名}
  WHERE {JMS 仕様に定義された有効メッセージ セレクタ文字列}
</jrun:getmsg>
```

属性 **msgsrc**

必須。java.lang.String または javax.jms.QueueConnectionFactory を取ります。

メッセージのソースであるメッセージ QueueConnectionFactory。文字列を指定すると、"java:comp/env/jms/[msgsrc]" で JNDI 検索 (ルックアップ) を実行することで、QueueConnectionFactory が取得できると仮定されます。QueueConnectionFactory オブジェクトを既定の InitialContext から取得できない場合は、jndi タグを使用して別の InitialContext を検索してファクトリを見つける必要があります。詳細については、「jndi」53 ページセクションを参照してください。

username

オプション。java.lang.String を取ります。

メッセージ キュー認証のためのユーザ名。

password

オプション。java.lang.String を取ります。

メッセージ キュー認証に必要なパスワード。

id

必須。java.lang.String を取ります。

取得するメッセージのスクリプト変数名。param タグを使用してこの名前を持つスクリプト変数を宣言します。

scope

オプション。java.lang.String または int を取ります。

このタグによって返されるオブジェクトの既定の JSP スコープ。有効な値は、page、request、session、および application です。既定値は page です。

スクリプト変数

id を指定した場合、getmsg タグによって返される結果セットは pageContext オブジェクトに allaire.taglib.MessageTable のインスタンスとして格納されます。返されるオブジェクトのスコープは、タグの scope によって決まり、その既定値は page です。param タグを使用してこのオブジェクトのスクリプト変数を宣言すると、定義されたスコープ内でその変数を暗黙オブジェクトのように使用できます。

getmsg タグによって返される allaire.taglib.MessageTable の列は、メッセージ プロパティ名とメッセージ本文の結合を最後の列として示します。この列には、必ず Message という名前が付いています。Message は、byte[]、Map、Object、または String のいずれかになります。allaire.taglib.MessageTable クラスにより、allaire.taglib.QueryTable クラスが拡張されます。QueryTable には javax.sql.RowSet インターフェイスが実装されています。このインターフェイスによって java.sql.ResultSet が拡張されます。sql、storedproc、getmsg、getmail、および jndi タグを使用する前に、javax.sql.RowSet および java.sql.ResultSet について把握しておくことをお勧めします。詳細については、次の URL にある JDBC API のマニュアルを参照してください。

http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html

これは、getmsg タグとスクリプト変数を併用する場合の共通のシナリオです。この例では getObject が使用されていますが、RowSet には異なるオブジェクト タイプを使用するほかの多くの getter メソッドがあります。

- param タグで MessageTable または RowSet にタイプ変換します。

  ```
  <jrun:param id="result" type="MessageTable"/>
  ```

  または

  ```
  <jrun:param id="result" type="javax.sql.RowSet"/>
  ```

- 結果セットをループ化します。

```
<jrun:foreach group='<%=result%>'>
<%= result.getObject("propertyname1") %><br>
<%= result.getObject("message") %><br>
</jrun:foreach>
```

次の表では、getmsg クエリ結果セットで最も一般的に使用されるメソッドについて説明しています。

メソッド	返されるタイプ	説明
next()	Boolean	TRUE の場合、結果セットで次の行を取得します。
getObject(int)	Object	インデックス番号によって列を取得します。
getObject(String)	Object	名前によって列を取得します。
get(int)	String	インデックス番号によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(int) の使用をお勧めします。
get(String)	String	名前によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(String) の使用をお勧めします。

前提条件 sendmsg および getmsg タグを使用するには、EJB および JMS オプションを含む完全な JRun インストールを実行する必要があります。ユーザ独自のメッセージ キューを指定する場合は、JRun サーバーの local.properties ファイルを編集して、次のエントリを加えます。

```
jms.queue.[your queue name].description=[description]
jms.queue.[your queue name].display-name=[displayname in JMC]
```

例 この JSP は、getmsg タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>getmsg Tag Example</title>
</head>
<body>
<%--
Queue1 という名前のメッセージ キューからすべてのメッセージを取得します。
<jrun:getmsg msgsrc="QueueConnectionFactory" id="result">
select * from Queue1
</jrun:getmsg>
<%--
```

結果セットをループ化し、メッセージ property1 および取得されたメッセージのメッセージ本文を表示します。

```
--%>
<jrun:param id="table" type="MessageTable"/>
<jrun:foreach group="<%= result %>">
   <%= result.getObject("propertyname1") %><br>
   <%= result.getObject("message") %><br>
</jrun:foreach>
</body>
</html>
```

sendmail

説明 マルチパートの電子メール メッセージを構築し、それを指定のメール サーバーに送信します。

sendmail タグ内でフロー制御タグを使用して、電子メール本文を動的に構築できます。sendmail タグを mailparam タグと併用すると、電子メールの添付ファイルが組み込めます。

sendmail タグでは、2 種類の構文がサポートされます。1 つめのタイプでは、ユーザがメール サーバー情報を指定できます。2 つめのタイプでは、Web アプリケーションで使用する JavaMail セッション オブジェクトを JRun 管理者があらかじめ定義しておく必要があります。詳細については、43 ページの「前提条件」セクションを参照してください。

構文 1 指定されたメール サーバー プロパティを使用して電子メール メッセージを送信します。

```
<jrun:sendmail
  host="host name"
  sender="e-mail address"
  recipient="e-mail address"
  [contentType="MIME type;charset=valid character set"]
  [port="port number"]
  [timeout="3000|timeout in milliseconds"]
  [cc="e-mail address"]
  [bcc="e-mail address"]
  [subject="message subject"]>
  オプション <jrun:mailparam/> のある電子メール メッセージ本文
</jrun:sendmail>
```

属性

host

必須。属性として java.lang.String を取ります。

メール サーバー ホスト名。

sender

必須。java.lang.String または javax.mail.internet.InternetAddress を取ります。

送信者の電子メール アドレス。

recipient

必須。java.lang.String、java.lang.String[]、java.util.Enumeration、java.util.Iterator、javax.mail.internet.InternetAddress、または javax.mail.internet.InternetAddress[] を取ります。

受信者の電子メール アドレス。文字列を指定すると、sendmail によって、文字列がカンマ区切りの電子メール アドレス一覧であるかどうかが確認されます。

contentType

オプション。java.lang.String を取ります。

電子メール メッセージ コンテンツの MIME タイプおよび文字セット。

port

オプション。java.lang.String、java.lang.Integer、または int を取ります。

メール サーバーのポート 番号。文字列を指定した場合は、整数値に解析されます。既定値は 25 です。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

メール サーバーのミリ秒単位のタイムアウト値。文字列を指定した場合は、整数値に解析されます。既定値は 3000 です。

cc

オプション。java.lang.String、java.lang.String[]、java.util.Enumeration、java.util.Iterator、javax.mail.internet.InternetAddress、または javax.mail.internet.InternetAddress[] を取ります。

カーボン コピー (cc) 受信者の電子メール アドレス。文字列を指定すると、sendmail によって、文字列がカンマ区切りの電子メール アドレス一覧であるかどうかが確認されます。

bcc

オプション。java.lang.String、java.lang.String[]、java.util.Enumeration、java.util.Iterator、javax.mail.internet.InternetAddress、または javax.mail.internet.InternetAddress[] を取ります。

ブラインド カーボン コピー (bcc) 受信者の電子メール アドレス。文字列を指定すると、sendmail によって、文字列がカンマ区切りの電子メール アドレス一覧であるかどうかが確認されます。

subject

オプション。java.lang.String を取ります。

電子メールの件名。

構文 2 指定された JavaMail セッション オブジェクトを使用して電子メール メッセージを送信します。

```
<jrun:sendmail
  session="JavaMail session object"
  sender="e-mail address"
  recipient="e-mail address"
  [contentType="mime type;charset=valid character set"]
  [cc="e-mail address"]
  [bcc="e-mail address"]
  [subject="message subject"]>
  オプション <jrun:mailparam/> のある電子メール メッセージ本文
</jrun:sendmail>
```

属性 **session**

必須。java.lang.String または javax.mail.Session を取ります。

JavaMail セッション オブジェクト。文字列を指定すると、"java:comp/env/mail/[session]" で JNDI 検索 (ルックアップ) を実行することにより、メール セッション オブジェクト が取得できると 仮定されます。セッション オブジェクト を既定の InitialContext から取得できない場合は、jndi タグを使用して別の InitialContext からセッションを検索します。詳細については、「jndi」53 ページセクションを参照してください。通常、JRun 管理者によって、JavaMail セッション オブジェクトがあらかじめインストールされています。

sender

必須。java.lang.String または javax.mail.internet.InternetAddress を取ります。

送信者の電子メール アドレス。

recipient

必須。java.lang.String、java.lang.String[]、java.util.Enumeration、java.util.Iterator、javax.mail.internet.InternetAddress、または javax.mail.internet.InternetAddress[] を取ります。

受信者の電子メール アドレス。文字列を指定すると、sendmail によって、文字列がカンマ区切りの電子メール アドレス一覧であるかどうかが確認されます。

contentType

オプション。java.lang.String を取ります。

電子メール メッセージ コンテンツの MIME タイプおよび文字セット。

cc

オプション。java.lang.String、java.lang.String[]、java.util.Enumeration、java.util.Iterator、javax.mail.internet.InternetAddress、または javax.mail.internet.InternetAddress[] を取ります。

カーボン コピー (cc) 受信者の電子メール アドレス。文字列を指定すると、sendmail によって、文字列がカンマ区切りの電子メール アドレス一覧であるかどうかが確認されます。

bcc

オプション。java.lang.String、java.lang.String[]、java.util.Enumeration、java.util.Iterator、javax.mail.internet.InternetAddress、または javax.mail.internet.InternetAddress[] を取ります。

ブラインド カーボン コピー (bcc) 受信者の電子メール アドレス。文字列を指定すると、sendmail によって、文字列がカンマ区切りの電子メール アドレス一覧であるかどうかが確認されます。

subject

オプション。java.lang.String を取ります。

電子メールの件名。

前提条件 sendmail タグおよび getmail タグで JavaMail セッション属性を使用するには、JRun サーバーの local.properties ファイルを編集して、次のエントリを加えます。

```
mail.[session name].description=[JavaMail session description]
mail.[session name].display-name=[JavaMail session display name in JMC]
mail.[session name].store.protocol=[pop3|imap (message store protocol]
mail.[session name].transport.protocol=smtp (transport protocol)
mail.[session name].host =[Mail server host name]
mail.[session name].user =[Default user name]
mail.[session name].smtp.host=[Default SMTP server host name]
mail.[session name].smtp.user=[Default SMTP user name]
mail.[session name].imap.host=[Default IMAP server host name]
mail.[session name].imap.user=[Default IMAP server user name]
mail.[session name].pop3.host=[Default POP3 server host name]
mail.[session name].pop3.user=[Default POP3 server user name]
mail.[session name].from=[Default sender email address]
mail.[session name].debug=[true|false (enable debug messages)]
```

例 この JSP は、sendmail タグを使用して電子メール メッセージを送信する 2 種類の方法を示しています。電子メールの添付ファイル送信の例については、「mailparam」44 ページセクションを参照してください。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>sendmail Tag Example</title></head><body>
<%--
(sendmail 構文 1) メール サーバー プロパティを指定してメールを送信します。
--%>
<jrun:sendmail host="company1.com"
sender="someone@somewhere.com"
recipient="someone@somewhere.com">
Hello world.
</jrun:sendmail>
<%--
(sendmail 構文 2) JavaMail セッション オブジェクトを使用してメールを送信します。
--%>
<jrun:sendmail session="session1"
sender="someone@somewhere.com"
recipient="someoneelse@somewhereIse.com">
Hello world.
</jrun:sendmail>
</body></html>
```

mailparam

説明 sendmail タグと併用して、電子メールメッセージにファイルを動的に添付したり、追加の電子メール ヘッダを指定します。

構文 1 1 つまたは複数の電子メール添付ファイル URL を指定します。

```
<jrun:mailparam
  attachurl="attachment URL"
/>
```

属性 **attachurl**

必須。java.lang.String を取ります。

MIME メッセージに組み込むドキュメント (電子メール添付ファイル) の URL。「/」で始まる絶対 URL にしたり、このタグの使用するページに対する相対 URL にすることもできます。

構文 2 追加の電子メール ヘッダを指定します。

```
<jrun:mailparam
  name="e-mail header name"
  value="e-mail header value"
/>
```

属性 **name**

必須。java.lang.String を取ります。

追加の電子メール ヘッダ名。

value

必須。java.lang.String を取ります。

追加の電子メール ヘッダ値。

例 この JSP は、sendmail タグ内での mailparam タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>mailparam Tag Example</title></head><body>
<--%
(mailparam 構文 1) sendmail タグ内の mailparam タグで電子メール添付ファイルを指定します。
--%>
<jrun:sendmail host="company1.com"
sender="someone@somewhere.com"
recipient="someone@somewhere.com">
Hello world.
<jrun:mailparam attachurl="allaire.gif" />
</jrun:sendmail>
</body></html>
```

getmail

説明 指定されたメール サーバー上の指定されたメール フォルダから、電子メール メッセージの取得や削除を行います。

このタグでは、SQL SELECT または DELETE で囲まれた文字列を使用して、getmail タグで返される電子メール フォルダ名、検索条件、メッセージ取得順序、電子メール ヘッダ / 内容を識別します。getmail タグ内にフロー制御タグを含めると、SELECT または DELETE ステートメントを動的に構築できます。

次の表は、getmail SELECT または DELETE ステートメントの WHERE 節で使用できる有効な検索条件を示します。メッセージの取得後に、これらの同じ検索条件を使用して、返された電子メール メッセージにアクセスして表示できます。

検索条件	演算子	データ タイプ / 形式
Sender	=、contains	String literal
Recipient	=、contains	String literal
Cc	=、contains	String literal
Bcc	=、contains	String literal
MessageID	=、contains	String literal
MessageNumber	=、>=、>、<=、<、!=	Integer
Subject	=、contains	String literal
SentDate	=、>=、>、<=、<、!=	MM/dd/yyyy hh:mm:ss
ReceivedDate	=、>=、>、<=、<、!=	MM/dd/yyyy hh:mm:ss
Size	=、>=、>、<=、<、!=	Integer
Flag	=、!=	次のいずれかとなります。 • ANSWERED (返信済み) • DELETED (削除済み) • DRAFT (下書き) • FLAGGED (フラグ) • RECENT (最近) • SEEN (既読) • USER (ユーザ)
Body	contains	String literal

たとえば、次に示すように、かっこを使用して論理演算子 and、or、および not を検索条件と組み合わせて使用できます。

```
SELECT * FROM InBox
WHERE (Sender contains 'macromedia.com' or
  Sender contains 'sun.com')
  and Size < 1000
```

getmail タグでは、2 種類の構文がサポートされます。1 つめのタイプでは、ユーザがメール サーバー情報を指定できます。2 つめのタイプでは、Web アプリケーションで使用する JavaMail セッション オブジェクトを JRun 管理者があらかじめ定義しておく必要があります。詳細については、「前提条件」セクションを参照してください。

構文 1 指定されたメール サーバー プロパティを使用して電子メール メッセージを取得します。

```
<getmail
  host="host name"
  username="user name"
  password="password"
  id="variable name"
  protocol="pop3|imap"
  [port="port number"]
  [timeout="timeout in milliseconds"]
  [scope="page|request|session|application"]>
  SELECT {メール ヘッダのカンマ区切り一覧または *}
  FROM {メール フォルダ名}
  WHERE {JavaMail API で使用できるオプションの検索条件}
</getmail>
```

属性 **host**

必須。java.lang.String を取ります。

メール サーバー ホスト名。

username

必須。java.lang.String を取ります。

電子メール メッセージ取得に必要なユーザ名。

password

必須。java.lang.String を取ります。

電子メール メッセージ取得に必要なパスワード。

id

必須。java.lang.String を取ります。

取得されたメッセージのスクリプト変数名。

port

オプション。java.lang.String、java.lang.Integer、または int を取ります。

メール サーバーのポート番号。文字列を指定した場合は、整数値に解析されます。

timeout

オプション。java.lang.String、java.lang.Integer、または int を取ります。

メール サーバーのミリ秒単位のタイムアウト値。文字列を指定した場合は、整数値に解析されます。既定値は 3000 です。

scope

オプション。java.lang.String または int を取ります。

このタグによって返されるオブジェクト の既定の JSP スコープ。有効文字列値は、page、request、session、および application です。既定値は page です。

protocol

必須。java.lang.String を取ります。

getmail タグで使用されるメール プロトコル。有効値は imap および pop3 です。

構文 2 指定された JavaMail セッション オブジェクトを使用して電子メール メッセージを取得します。

```
<getmail
  session="JavaMail session name"
  username="user name"
  password="password"
  id="variable name"
  protocol="pop3|imap">
  [scope="page|request|session|application"]
  SELECT {メール ヘッダのカンマ区切り一覧または *}
  FROM {メール フォルダ名}
  WHERE {JavaMail API で使用できるオプションの検索条件}
</getmail>
```

属性 **session**

必須。java.lang.String または javax.mail.Session を取ります。

JavaMail セッション オブジェクト 。文字列を指定すると、"java:comp/env/mail/[session]" で JNDI 検索 (ルックアップ) を行う ことにより 、メール セッション オブジェクト が取得できると 仮定されます。セッション オブジェクト を既定の InitialContext から取得できない場合は、jndi タグを使用して別の InitialContext からセッションを検索します。詳細については、「jndi」53 ページセクションを参照してください。通常、JRun 管理者によって、JavaMail セッション オブジェクトがあらかじめインストールされています。

username

必須。java.lang.String を取ります。

電子メール メッセージ取得に必要なユーザ名。

password

必須。java.lang.String を取ります。

電子メール メッセージ取得に必要なパスワード。

id

必須。java.lang.String を取ります。

取得するメッセージのスクリプト変数名。

protocol

必須。java.lang.String を取ります。

getmail タグで使用されるメール プロトコル。有効値は imap および pop3 です。

scope

オプション。java.lang.String または int を取ります。

このタグによって返されるオブジェクト の既定の JSP スコープ。有効文字列値は、page、request、session、および application です。既定値は page です。

スクリプト変数

id を指定した場合、getmail タグによって返される結果セットは pageContext オブジェクトに allaire.taglib.EMailTable のインスタンスとして格納されます。返されるオブジェクトのスコープは、タグの scope 属性によって決まり、その既定値は page です。

メモ

getmail タグによって返される allaire.taglib.EMailTable の列は、電子メール メッセージ ヘッダと元のメッセージの結合を最後の列として示します。この列には、必ず Message という名前が付いています。

allaire.taglib.EMailTable クラスにより、allaire.taglib.QueryTable クラスが拡張されます。QueryTable には javax.sql.RowSet インターフェイスが実装されています。このインターフェイスによって java.sql.ResultSet が拡張されます。sql、storedproc、getmsg、getmail、および jndi タグを使用する前に、javax.sql.RowSet および java.sql.ResultSet について把握しておくことをお勧めします。詳細については、次の URL にある JDBC API のマニュアルを参照してください。

http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html

これは、getmsg タグとスクリプト変数を併用する場合の共通のシナリオです。この例では getObject が使用されていますが、RowSet には異なるオブジェクト タイプを使用するほかの多くの getter メソッドがあります。

- param タグで EMailTable または RowSet にタイプ変換します。
  ```
  <jrun:param id="result" type="EMailTable"/>
  ```
  または
  ```
  <jrun:param id="result" type="javax.sql.RowSet"/>
  ```
- 結果セットをループ化します。
  ```
  <jrun:foreach group='<%=result%>'>
  <%= result.getObject("From") %><br>
  <%= result.getObject("Subject") %><br>
  <%= result.getObject("Body") %><br>
  </jrun:foreach>
  ```

次の表では、getmail 結果セットで最も一般的に使用されるメソッドについて説明しています。

メソッド	返されるタイプ	説明
next()	Boolean	TRUE の場合、結果セットで次の行を取得します。
getObject(int)	Object	インデックス番号によって列を取得します。
getObject(String)	Object	名前によって列を取得します。
get(int)	String	インデックス番号によって列を取得します。この QueryTable メソッド は下位互換性のために用意されているので、同じ 機能を 持つ getObject(int) の使用をお勧めします。
get(String)	String	名前によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(String) の使用をお勧めします。

前提条件 sendmail および getmail タグで JavaMail session 属性を使用するには、JRun サーバーの local.properties ファイルを編集して、次のエントリを加えます。

```
mail.[session name].description=[JavaMail session description]
mail.[session name].display-name=[JavaMail session display name in JMC]
mail.[session name].store.protocol=[pop3|imap (message store protocol]
mail.[session name].transport.protocol=smtp (transport protocol)
mail.[session name].host =[Mail server host name]
mail.[session name].user =[Default user name]
mail.[session name].smtp.host=[Default SMTP server host name]
mail.[session name].smtp.user=[Default SMTP user name]
mail.[session name].imap.host=[Default IMAP server host name]
mail.[session name].imap.user=[Default IMAP server user name]
mail.[session name].pop3.host=[Default POP3 server host name]
mail.[session name].pop3.user=[Default POP3 server user name]
mail.[session name].from=[Default sender email address]
mail.[session name].debug=[true|false (enable debug messages)]
```

例 この JSP は、getmail を使用して電子メール メッセージを取り込む 2 種類の方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>getmail Tag Example</title></head><body>
<%--
(getmail 構文 1) 電子メール サーバー プロパティを使用して電子メール メッセージを取
        得します。
--%>
<jrun:getmail host="server.somewhere.com" username="test"
password="test" protocol="imap" id="result">
select subject, from, body from Inbox
```

```
</jrun:getmail>
<%--
(getmail 構文 2) JavaMail セッション オブジェクトを使用して電子メール メッセージ
        を取得します。
--%>
<jrun:getmail session="session1" id="e-mails" protocol="pop3">
select * from Inbox where subject contains 'jrun'
</jrun:getmail>
<%--
param タグを使用して変数を宣言し、getmail タグによって返される EmailTable
オブジェクトを javax.sql.RowSet にタイプ変換します。結果セットをループ化し、
getObject メソッドを使用して電子メール メッセージの件名、送信元、および
本文セクションを取得します。
--%>
<jrun:param id='result' type='javax.sql.RowSet'/>
<jrun:foreach group='<%= result %>'>
<%= result.getObject("subject") %><br>
<%= result.getObject("from") %><br>
<%= result.getObject("body") %><br>
</jrun:foreach>
</body></html>
```

servlet

説明 servletparam からサーブレット属性を収集し、RequestDispatcher インターフェイスを使用して、指定されたサーブレットを呼び出します。呼び出されたサーブレットの出力は、呼び出した JSP に埋め込まれます。

構文

```
<jrun:servlet
  code="servlet name">
  optional <servletparam/>
</jrun:servlet>
```

属性 **code**

必須。java.lang.String を取ります。

ターゲットのサーブレット名。

例 この JSP は、servlet タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>servlet Tag Example</title></head><body>
<%--
サーブレットを呼び出します。
--%>
<jrun:servlet code="SnoopServlet">
</jrun:servlet>
</body>
</html>
```

servletparam

説明 servlet タグと併用してサーブレット要求属性を指定します。

構文
```
<jrun:servletparam
  name="servlet attribute name"
  value="servlet attribute value"
/>
```

属性

name

必須。java.lang.String を取ります。

ターゲットのサーブレット属性名。

value

必須。java.lang.Object を取ります。

ターゲットのサーブレット属性値。

例 この JSP は、servlet タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>servletparam Tag Example</title></head><body>
<%--
servlet タグ内の servletparam タグでサーブレット属性を指定します。
--%>
<jrun:servlet code="SnoopServlet">
<jrun:servletparam name="key" value="<%= new Hashtable() %>"/>
</jrun:servlet>
</body>
</html>
```

jndi

説明 JSP から J2EE サーバー環境の分散型オブジェクトにアクセスできるようになります。

このタグでは、lookup (JNDI 検索 (ルックアップ))、list (一覧)、attribute (属性)、および search (検索) アクションがサポートされます。

構文

```
<jrun:jndi
  action="lookup|list|search|attribute"
  name="lookup/search name"
  attributes="search attributes" (検索のみ)
  [provider="service provider class name"]
  [url="service provider URL"]
  id="variable name"
  [scope="page|request|session|application"]
/>
```

属性

action

必須。java.lang.String を取ります。

JNDI アクション。値は、lookup、list、attribute、および search です。各 action の機能は次のとおりです。

- lookup の場合、このタグは JNDI サーバーからのオブジェクトを返します。
- list の場合、このタグは接頭辞が一致するオブジェクトの一覧を返します。
- attribute の場合、このタグは LDAP などの指定されたディレクトリ サービスから属性の一覧を返します。
- search の場合、指定ディレクトリ サービスから名前にバインドされたオブジェクト の列挙を返します。

name

必須。java.lang.String または javax.naming を取ります。

JNDI 検索 (ルックアップ)/検索名。javax.naming.Name のインスタンスも使用できます。

attributes (検索アクションのみ)

必須。java.util.Dictionary、java.util.Map、または javax.naming.Attributes です。

検索対象のキーと値ペアのセット。たとえば、Hashtable、Hashmap、または javax.naming.Attributes のインスタンスのフォームのキーと値のペアを指定できます。

provider

オプション。java.lang.String を取ります。

JNDI 検索 (ルックアップ)/ディレクトリ サービス プロバイダ クラス。

url

オプション。java.lang.String を取ります。

この属性は、JNDI 検索 (ルックアップ)/ディレクトリ サービス プロバイダ URL を表します。

id

必須。java.lang.String を取ります。

jndi 結果オブジェクトのスクリプト変数名。

scope

オプション。java.lang.String または int を取ります。

このタグによって返されるオブジェクト の既定の JSP スコープ。有効文字列値は、page、request、session、および application です。既定値は page です。

スクリプト変数

jndi タグによって返されるオブジェクトのタイプは、action によって異なります。返されるオブジェクトのスコープは、タグの scope によって決まり、その既定値は page です。param タグを使用してこのオブジェクトのスクリプト変数を宣言すると、定義されたスコープ内でその変数を暗黙オブジェクトのように使用できます。次の表は、各 action、アクションによって返されるオブジェクト、および返されるオブジェクトの列名とそのデータ タイプの一覧です。

JNDI アクション	返されるオブジェクト	テーブル列 : データ タイプ
lookup	java.lang.Object	なし
list	allaire.taglib.NameTable	Name: String ClassName: String isRelative: Boolean
search	allaire.taglib.DirSearchTable	Name: String ClassName: String isRelative: Boolean Attribute: allaire.taglib.AttributeTable
attribute	allaire.taglib.AttributeTable	ID: String Attribute: javax.naming.directory.Attribute

allaire.taglib.NameTable、DirSearchTable、および AttributeTable クラスにより 、allaire.taglib.QueryTable クラスが拡張されます。QueryTable には javax.sql.RowSet インターフェイスが実装されています。このインターフェイスによって java.sql.ResultSet が拡張されます。sql、storedproc、getmsg、getmail、および jndi タグを使用する前に、javax.sql.RowSet および java.sql.ResultSet について把握しておくことをお勧めします。詳細については、次の URL にある JDBC API のマニュアルを参照してください。

http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html

これは、一覧アクションで jndi タグのスクリプト変数を使用する場合の共通のシナリオです。この例では getObject が使用されていますが、RowSet には異なるオブジェクト タイプを使用するほかの多くの getter メソッドがあります。

- param タグで NameTable または RowSet にタイプ変換します。
  ```
  <jrun:param id="result" type="NameTable"/>
  ```
  または
  ```
  <jrun:param id="result" type="javax.sql.RowSet"/>
  ```
- 結果セットをループ化します。
  ```
  <jrun:foreach group='<%=result%>'>
  <%= result.getObject("Name") %><br>
  <%= result.getObject("ClassName") %><br>
  <%= result.getObject("isRelative") %><br>
  </jrun:foreach>
  ```

次の表では、jndi list、search、および attribute 結果セットで最も一般的に使用されるメソッドについて説明しています。

メソッド	返されるタイプ	説明
next()	Boolean	TRUE の場合、結果セットで次の行を取得します。
getObject(int)	Object	インデックス番号によって列を取得します。
getObject(String)	Object	名前によって列を取得します。
get(int)	String	インデックス番号によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(int) の使用をお勧めします。
get(String)	String	名前によって列を取得します。この QueryTable メソッドは下位互換性のために用意されているので、同じ機能を持つ getObject(String) の使用をお勧めします。

前提条件 Web アプリケーションで jndi タグを使用する前に、JNDI ディレクトリ サービスを使用可能にする必要があります。

例 この JSP は、jndi タグの使用方法および結果データの表示方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>jndi Tag Example</title></head><body>
<%--
JNDI 検索（ルックアップ）アクションを実行します。
--%>
<jrun:jndi action="lookup" name="java:comp/UserTransaction" id="txn"/>
<%--
```

```
JNDI 一覧アクションを実行します。
--%>
<jrun:jndi action="list" name="java:comp/env/jdbc" id="result"/>

<%--
結果セットをループ化し、名前値を表示します。
--%>
<jrun:param id="result" type="NameTable"/>
<jrun:foreach group='<%= result %>'>
<%= result.getObject("Name") %><br>
<%= result.getObject("ClassName") %><br>
<%= result.getObject("isRelative") %><br>
</jrun:foreach>
</body></html>
```

transaction

説明 JSP に分散型トランザクション ロジックを書き込むことができます。複数のコンピュータで複数のデータベース テーブルを更新する場合など、データの完全性を維持する必要がある場合に、分散型トランザクションを使用します。いずれかのデータベース サーバーで障害が発生した場合は、その他のデータベース サーバーに加えられた変更をトランザクションによってロール バックする必要があります。メッセージ キューへのメッセージの送信処理も、分散型トランザクションにより行われるため、トランザクションに障害が発生した場合はメッセージがキャンセルされます。一般に transaction タグは、sql、sendmsg、および getmsg タグと併用されます。これらのタグの場合は、transaction タグ内にあるかどうかが確認され、それに応じてロジックが使用されます。

構文

```
<jrun:transaction
  [timeout="time in seconds"]>
  <jrun:sql>
  SQL statement
  </jrun:sql>
  more <jrun:sql>, <jrun:sendmsg> or <jrun:getmsg> tags...
</jrun:transaction>
```

属性 **timeout**

オプション。java.lang.String、java.lang.Integer、または int を取ります。

トランザクションに関連付けられているミリ秒単位のタイムアウト値。この値が 0 の場合、トランザクション サービスでは既定値を復元します。

前提条件 transaction タグでは、*JRun のルート ディレクトリ*/samples/taglib ディレクトリにある jrunbeans.jar ファイルのステートレス セッション Bean を使用します。Bean の公開記述子によって、すべての Bean メソッドに対するセキュリティの役割として txntag が指定されます。JRun インストール時に、EJB および JMS オプションを含む完全なインストールを実行する必要があります。transaction タグを使用する前に、*JRun のルート ディレクトリ*/samples/taglib ディレクトリの Readme ファイルの説明どおりに Bean を公開し、Web アプリケーションの web.xml ファイルと JRun users.properties ファイルを変更する必要があります。

例 この JSP は、2 つの sql タグと 1 つの sendmsg が含まれている transaction タグを示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>transaction Tag Example</title></head><body>
<jrun:transaction>
<jrun:sql datasrc="taglib">
update Table1 set amount=amount+10 where id=2
</jrun:sql>
<jrun:sql datasrc="taglib2">
update Table1 set amount=amount-10 where id=2
</jrun:sql>
<jrun:sendmsg msgsrc="QueueConnectionFactory" queue="queue1">
<jrun:msgparam name="foo1" value="bar1"/>
<jrun:msgparam name="foo2" value="bar2"/>
This is a test.
</jrun:sendmsg>
</jrun:transaction>
</body></html>
```

param

説明 JSP スクリプトレットおよび数式内で使用するためにスクリプト変数を宣言します。

JSP スコープ (page、request、session、または application) を指定したり、オプションで既定値を指定できます。param タグで設定された変数は、pageContext オブジェクトから変数を取り込むことによってほかのカスタム タグからもアクセスできます。param タグではスクリプト変数のリセットは行いません。

メモ
ほかのタグ内に param タグをネストする場合、param タグはネストされたコンテキスト内だけで使用できます。

構文

```
<jrun:param
  id="variable name"
  [scope="page|request|session|application"]
  [type="java.lang.Object|class name"]
  [default="default value"]
/>
```

属性

id
必須。java.lang.String を取ります。
スクリプト変数名。

scope
オプション。java.lang.String を取ります。
スクリプト変数の JSP スコープ。有効な値は、page、session、request、および application です。既定値は page です。

type
オプション。java.lang.String を取ります。
スクリプト変数オブジェクト タイプ。

default
オプション。java.lang.Object を取ります。
スクリプト変数の既定値。

スクリプト変数 id を指定して、scope が page の場合は、type 属性で指定されたタイプの、次の特性を持つ変数が作成されます。

- 名前　TagData.getAttribute(“id")
- タイプ　TagData.getAttribute(“type")
- 宣言　true
- スコープ　AT_END

例 この JSP は、param タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>param Tag Example</title></head><body>
<%--
整数オブジェクトを宣言し、セッションのスコープに格納します。
--%>
<jrun:param id="x" type="java.lang.Integer"
default='<%= new Intger(1) %>' scope="session"/>
<%= session.getAttribute("x") %>
<%--
日付オブジェクトを宣言し、ページのスコープに格納します。
--%>
<jrun:param id="y" type="java.util.Date"
default="<%= new Date() %>" scope="page"/>
<%= y %>
</body></html>
```

foreach

説明 group 属性で定義されたオブジェクトのコレクションをループ化します。

foreach タグを使用すると、列挙タイプ (java.util.Enumeration、javax.naming.NamingEnumeration、java.sql.ResultSet、javax.sql.RowSet、java.util.Iterator、または java.lang.Object[]) が自動的に認識されるため、それぞれのコレクション構造で列挙メソッドを覚える必要がなくなります。コレクションで次のタイプのいずれかが使用される場合に、item および type 属性が使用されます。

- java.util.Enumeration
- javax.naming.NamingEnumeration
- java.util.Iterator
- java.lang.Object[]

スクリプト変数が作成され、コレクションから返されたオブジェクトが保持されます。

構文 1 group 属性を使用して指定されたコレクションからのオブジェクトを返します。

```
<jrun:foreach
  group="group name"
  item="variable name"
  [type="java.lang.Object|class name"]>
  ...
</jrun:foreach>
```

属性

group

必須。java.util.Enumeration、javax.naming.NamingEnumeration、java.util.Iterator、allaire.taglib.Table、java.lang.Object[]、または java.lang.String を取ります。

コレクション名。文字列を指定すると、pageContext.getAttribute() を呼び出すことで、列挙オブジェクトを見つけることができると仮定されます。

item

特定の group タイプの場合は必須。java.lang.String を取ります。

スクリプト変数名。この属性は、グループ属性が java.util.Enumeration、javax.naming.NamingEnumeration、java.util.Iterator、または java.lang.Object[] である場合に必要となります。

type

オプション。java.lang.String を取ります。

item を指定した場合、この属性は、item ループのスクリプト変数の作成に使用されるオブジェクト タイプを示します。既定値は java.lang.Object です。

構文 2 列挙のデータに直接アクセスします。

```
<jrun:foreach
  group="group name">
  ...
</jrun:foreach>
```

属性 **group**

必須。java.sql.ResultSet、javax.sql.RowSet、または java.lang.String を取ります。

コレクション名。文字列を指定すると、pageContext.getAttribute() を呼び出すことで、列挙オブジェクトを見つけることができると仮定されます。

スクリプト変数 item を指定した場合は、その値が変数名として使用されます。

スクリプト変数の情報は次のようになります。

- **名前** TagData.getAttribute("item")
- **タイプ** TagData.getAttribute("type")
- **宣言** true
- **スコープ** NESTED

例 この JSP は、2 種類の foreach 構文の使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>foreach Tag Example</title></head><body>
<%--
(foreach 構文 1) コレクションからオブジェクトが返されます。
--%>
<jrun:foreach item="c" type="Cookie"
        group="<%= request.getCookies() %>">
<%= c.getname() %>
</jrun:foreach>
<%--
(foreach 構文 2) 列挙のデータに直接アクセスします。
--%>
<jrun:sql datasrc="dsn1" id="rs">
SELECT * FROM Table1
</jrun:sql>

<jrun:param id="rs" type="QueryTable"/>
<jrun:foreach group="<%= rs %>">
... <%= rs.get("column_name") %> ...
</jrun:foreach>
</body></html>
```

if

説明 expr 属性で指定された式を評価し、この式の結果に従って囲まれたコード ブロックを条件分岐して実行します。

構文

```
<jrun:if
  [expr="true|boolean expression]
  ...">
</jrun:if>
```

属性 **expr**

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

評価する式。有効な文字列値は、true、false、TRUE、および FALSE です。既定値は true です。

例 この JSP は、if タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>if Tag Example</title></head><body>

<jrun:if expr="<%= "true".equals(request.getParameter("foo") %>">
foo = true<br>
</jrun:if>
</body></html>
```

switch

説明 case タグ内のコードの条件ブロックを実行します。

このタグは、式が true であると最初に評価されたケース ブロックを実行します。switch タグ内の case タグのそれぞれにオプションで独自の評価式を組み込めます。

構文

```
<jrun:switch>
  one or more <case> tags
</jrun:switch>
```

例 この JSP は、switch タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>switch Tag Example</title></head><body>

<jrun:switch>
<jrun:case expr="true">
....
</jrun:case>
<jrun:case expr="true">
....
</jrun:case>
<jrun:case>
....
</jrun:case>
</jrun:switch>
</body></html>
```

case

説明 switch タグ内で使用され、評価式の結果に従ってカプセル化されたコードを条件分岐して実行します。

expr 属性はオプションです。既定値は true です。case タグは、switch タグ内にあるものとし、これ以外の場合は例外となります。switch タグは、式が true であると最初に評価された case ブロックを実行します。詳細については、「switch」セクションを参照してください。

構文

```
<jrun:case
  [expr="true|boolean expression"]>
</jrun:case>
```

属性

expr

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

評価する式。有効な文字列値は、true、false、TRUE、および FALSE です。既定値は true です。

例 この JSP は、case タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>case Tag Example</title></head><body>
<jrun:switch>
<jrun:case expr="true">
....
</jrun:case>
<jrun:case expr="true">
....
</jrun:case>
<jrun:case>
....
</jrun:case>
</jrun:switch>
</body></html>
```

form

説明 クライアント側で JavaScript フォームの妥当性を確認して、HTML form タグを拡張します。このタグは、既存の HTML 4.0 form タグ属性にも対応しています。

構文
```
<jrun:form
  name="HTML form name"
  action="URL"
  [onSubmit="JavaScript function name"]
  [other HTML 4.0 attributes]>
  optional <input/>|<select>...</select>
</jrun:form>
```

属性

name

必須。java.lang.String を取ります。

HTML フォーム名。

action

オプション。java.lang.String を取ります。

HTML フォーム アクション URL。有効な URL でなければなりません。

onSubmit

オプション。属性として java.lang.String を取ります。

JavaScript 関数名。送信ボタンをクリックすると、この JavaScript 関数が実行されます。

その他の HTML 4.0 属性

指定したその他の HTML 4.0 フォーム属性はすべて、生成された form タグの属性に追加されます。

例 この JSP は、form タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>form Tag Example</title></head><body>
<jrun:form name="form1" action="form.jsp">
<input type="submit" value="submit">
</jrun:form>
</body></html>
```

input

説明 form タグの中で使用し、ラジオ ボタン、チェックボックス、テキスト ボックスを挿入します。特定のコントロール タイプの入力の妥当性を確認します。

このタグは、HTML INPUT タグと同様に JavaScript の onClick イベントをサポートします。

構文 onValidate 値を指定した場合は、ユーザ定義の JavaScript の妥当性確認を使用します。それ以外の場合は、組み込みの妥当性確認を使用します。

```
<jrun:input
  name="name"
  [type="form control type"]
  [value="value"]
  [required="true|false"]
  [onError="JavaScript function name"]
  [other HTML 4.0 input attributes]
/>
```

属性 **name**

必須。java.lang.String を取ります。

ユーザ インターフェイス コントロール名。

type

オプション。java.lang.String を取ります。

フォーム コントロール タイプ。有効タイプは、checkbox、radio、password、creditcard、date、eurodate、float、integer、ssc、phone、time、zipcode、および text です。既定値は text です。

value

オプション。java.lang.String を取ります。

フィールド値。

required

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

この属性を指定すると、チェックボックスの選択、ラジオ ボタンの選択、パスワードの入力、入力フィールド ボックスへの文字列の入力が必須になります。既定値は false です。

onError

オプション。java.lang.String を取ります。

JavaScript 関数名。この関数は、クライアント側でフォームの妥当性確認ができない場合に実行されます。

その他の HTML 4.0 入力属性

指定したその他の HTML 4.0 入力属性はすべて、生成された input タグの属性に追加されます。

例 この JSP は、input タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>input Tag Example</title></head><body>
<%--
組み込みの電話フォームの妥当性確認を使用します。
--%>
<jrun:form name="form1" action="input.jsp">
<jrun:input name="input1" type="phone" required="true" />
<input type="submit" value="validate">
</jrun:form>

<%--
カスタム入力フィールドの妥当性確認およびエラー メッセージを使用します。
--%>
<script language="javascript">
<!--
function customValidate(obj_value) {
    if (!(obj_value.toString() == "PROMOTION")) {
     return false;
  }
  return true;
}
function customErrorMsg(form_obj, input_obj, obj_value, error_msg) {
    alert("custom error message:Invalid Promotion Code:"+obj_value);
    return false;
}
//-->
</script>

<jrun:form method="post" name="form1" action="form4.jsp">
<jrun:input name="t1" required="true" onValidate="customValidate"
  onError="customErrorMsg" />
<jrun:input name="t2" type="password" required="true"/>
<input type="submit" value="Submit"/>
</jrun:form>
</body></html>
```

select

説明 ハッシュテーブルのようなキーと値のペアのテーブルが取得され、ドロップダウン リスト ボックスに自動的に割り当てられます。また、データベース クエリ結果も取得できます。この場合は value および display 文字列としてクエリ から 2 つの列を指定する必要があります。

構文 1 ハッシュテーブルの値を使用してドロップダウン リスト ボックスに割り当てます。

```
<jrun:select
  name="name"
  hashtable="hashtable name"
  [multiple=”true|false”]
  [size="1|integer"]
  [required="true|false"]
  [onError="JavaScript function name"]
  [selected="value"]
  [other HTML 4.0 select attributes]>
  optional HTML <option> tags
</jrun:select>
```

属性

name

必須。java.lang.String を取ります。

ユーザ インターフェイス コントロール名。

size

オプション。java.lang.String、java.lang.Integer、または int を取ります。

ドロップダウン リスト ボックスに表示される行数。既定値は 1 です。文字列を指定した場合は、整数値に解析されます。

hashtable

必須。java.util.Dictionary または java.util.Map のサブクラスを取ります。

キーおよび値のペアのテーブル。

multiple

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

この属性を指定した場合は、複数の値を選択できます。既定値は false です。

required

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

この属性を指定すると、チェックボックスの選択、ラジオ ボタンの選択、パスワードの入力、入力フィールド ボックスへの文字列の入力が必須になります。既定値は false です。

onError

オプション。java.lang.String を取ります。

JavaScript 関数名。この関数は、クライアント側でフォームの妥当性確認ができない場合に実行されます。

selected

オプション。java.lang.String (複数の場合はカンマ区切りリスト)、java.lang.String[]、java.util.Enumeration、java.lang.Iterator を取ります。

ドロップダウン リスト ボックスであらかじめ選択される値で、ハッシュテーブルのキー値です。

その他の HTML 4.0 選択属性

指定したその他の HTML 4.0 選択属性はすべて、生成された select タグの属性に追加されます。

構文 2 データベース テーブルの指定された列を使用してドロップダウン リストに割り当てます。

```
<jrun:select
  name="name"
  query="database query result"
  [multiple="true|false"]
  [value="value column"]
  [display="display column"]
  [size="1|integer"]
  [required="true|false"]
  [onError="JavaScript function name"]
  [selected="value"]
  [other HTML 4.0 select attributes]>
  optional HTML <option> tags
</jrun:select>
```

属性 **name**

必須。java.lang.String を取ります。

フォーム コントロール名。

query

必須。java.sql.ResultSet、javax.sql.RowSet、allaire.taglib.Table、または java.lang.String を取ります。

データベース クエリ結果。この属性は、操作を簡単にする目的で文字列値に対応しています。次を呼び出してクエリ オブジェクトを取得します。

```
pageContext.getAttribute(query);
```

multiple

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

この属性を指定した場合は、複数の値を選択できます。既定値は false です。

value

オプション。java.lang.String を取ります。

キー値。既定値はクエリ結果の最初の列です。

display

オプション。java.lang.String を取ります。

表示値。既定値は value 属性の値です。

size

オプション。java.lang.String、java.lang.Integer、または int を取ります。

ドロップダウン リスト ボックスに表示される行数。既定値は 1 です。文字列を指定した場合は、整数値に解析されます。

required

オプション。java.lang.String、java.lang.Boolean、または boolean を取ります。

この属性を指定すると、チェックボックスの選択、ラジオ ボタンの選択、パスワードの入力、入力フィールド ボックスへの文字列の入力が必須になります。既定値は false です。

onError

オプション。java.lang.String を取ります。

JavaScript 関数名。この関数は、クライアント側でフォームの妥当性確認ができない場合に実行されます。

selected

オプション。java.lang.String を取ります。

ドロップダウン リスト ボックスであらかじめ選択される値で、ハッシュテーブルのキー値です。

その他の HTML 4.0 選択属性

指定したその他の HTML 4.0 選択属性はすべて、生成された select タグの属性に追加されます。

例 この JSP は、select タグの使用方法を示しています。

```
<%@ page import="allaire.taglib.*" %>
<%@ taglib uri="jruntags" prefix="jrun" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title>select Tag Example</title></head><body>
<jrun:param id="rs" type="allaire.taglib.QueryTable"/>
<jrun:sql driver="sun.jdbc.odbc.JdbcOdbcDriver"
url="jdbc:odbc:DBNAME" id="q3">
select * from Table1
</jrun:sql>
```

```
<jrun:form name="form1" action="select.jsp">
<jrun:select name="box1" required="true" query="<%= rs %>"
value="ID" display="LastName">
</jrun:select>
<input type="submit" value="validate">
</jrun:form>
</body></html>
```